Winning Space Race with Data Science

Soraya Rodríguez
July, 26th 2025

Outline

- Executive Summary
- Introduction
- Section 1: Methodology
- Section 2: Insights drawn from EDA
- Section 3: Launch Sites Proximities Analysis
- Section 4: Build a Dashboard with Plotly
- Section 5: Predictive Analysis (Classification)
- Conclusions
- Appendix

Executive Summary

Methodologies Summary

- REST API data collection from SpaceX with requests
- Web scraping with BeautifulSoup
- Data wrangling to clean, merge, and process features with numpy and pandas
- Exploratory analysis using SQL and Seaborn visualization
- Interactive map visualizations with Folium
- Interactive dashboard with Plotly Dash
- Machine Learning models with Scikit-Learn: Logistic Regression, SVM, Random Forest, Decision Tree

Results Summary

- Reusability: the impact of Launch Site, Payload Mass, Orbit and Booster Type on Success Rate.
- The 4 models reached 83% accuracy.
- False Positives: after tuning, 1 of the model's precision improved keeping the same predicting accuracy.

Introduction

Project Background and Context

- This project focuses on predicting SpaceX Falcon 9 first-stage landings — essential for cutting mission costs through reusable boosters.
- Cost Perspective: False positives (predicting success when failure happens) are costly. Reducing such errors helps improve mission cost-efficiency.

Problems to Solve

- Can we predict landing success?
- What features are most important?
- How reliable is our prediction model?
- False positives: Is *accuracy* or *precision* a better measure for the model?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - GET request to SpaceX API and parsing
  - Web scraping historical record from the List of Falcon 9 and Falcon Heavy launches on Wikipedia
- Perform data wrangling
- Perform exploratory data analysis (EDA) using with SQL and Seaborn visualization
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models:
  - Preprocessing data with StandardScaler
  - Splitting data into Train and Test sets
  - Cross Validation with GridSearchCV
  - Logistic Regression, SVM, Decision Tree and K-Nearest Neighbours Classification models for prediction

Data Collection

Summary:

- Data Collection -SpaceX API

  GET request to the SpaceX URL -although a json static response was used for the project.

- Data Collection -Web scraping

  Data gathering: GET request to a static URL from the "List of Falcon 9 and Falcon Heavy launches" on Wikipedia.

- Data Wrangling

Data Collection – SpaceX API

- Data was gathered by making a GET request to the SpaceX URL. The data was transformed into a dataframe using pd.json_normalized()
- Feature selection and transformation: a subset of features, rows with multiple cores were removed, single values for cores and payload were extracted, date was converted to datetime data type and the dates of launches of interest were restricted.
- Upon applying set functions, a dataframe was created from a dictionary including just Falcon 9 launches.
- For data wrangling, null values were replaced by the mean for numerical features (save for 26 in LaunchPad)
- Data was saved as a CSV file using pd.to_csv()

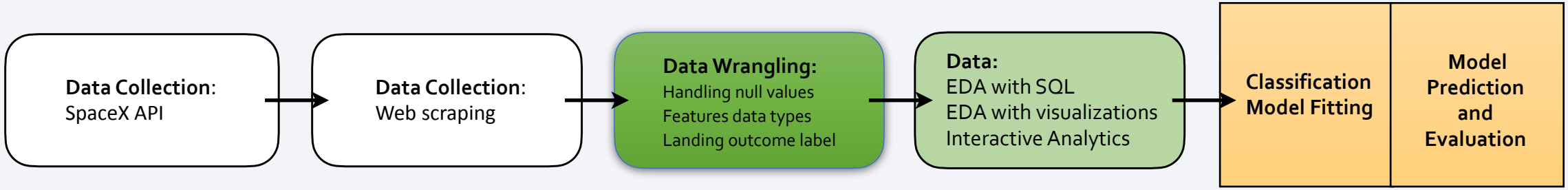
GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M1L1_jupyter-labs-spacex-data-collection-api.ipynb

Data Collection – Web scraping

- Data was gathered by making a GET request to a static URL from the List of Falcon 9 and Falcon Heavy launches on Wikipedia.
- Using a BeautifulSoup object, data was parsed to get the table.
- Upon applying set functions, data was gathered in columns and turned into a dataframe from a dictionary.
- Data was saved as a CSV file using pd.to_csv()

GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M1L2_jupyter-labs-webscraping.ipynb

Data Collection – Data Wrangling

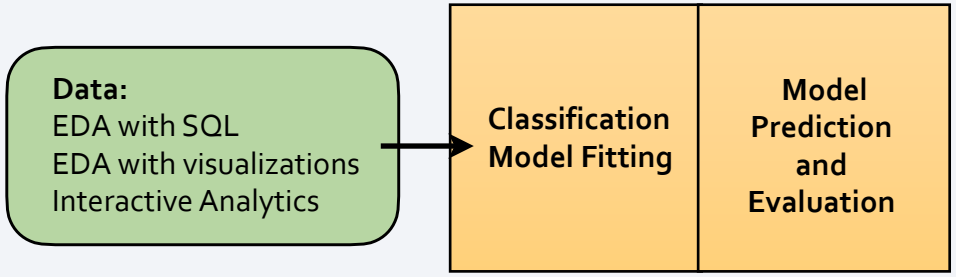


- Identifying proportion of null values and feature data types.
- Count of launches per site.
- Count and occurrence of orbits.
- Count and occurrence of mission outcomes.
- Landing Outcome label.

GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M1L2_jupyter-labs-webscraping.ipynb

EDA with Data Visualization I

- **Data Visualization: Plots and Graphs**



1. *FlightNumber* vs. *PayloadMass* scatter plot:

   To determine, as experience increases, whether the number of successful launches increases with respect to a heavier payload.

2. *FlightNumber* vs. *LaunchSite* scatter plot:

   To determine, as experience increases, whether the number of successful launches increases with respect to the site from which it was launched.

3. *PayloadMass* vs. *LaunchSite* scatter plot:

   To determine, as payloads get heavier, whether the number of successful launches increases with respect to the site from which it was launched.

4. *Orbit* vs. *Average Success Rate* bar plot:

   To identify the orbit type that has the highest success rates.

5. *FlightNumber* vs. *Orbit* scatter plot:

   To determine, as experience increases, whether the number of successful launches increases with respect to the orbit type.

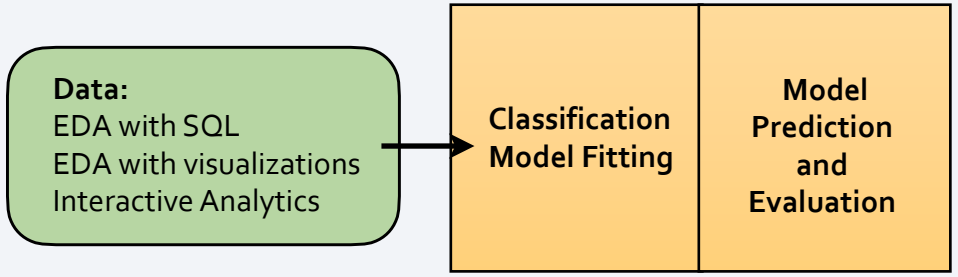
EDA with Data Visualization II

6. *PayloadMass* vs. *Orbit* scatter plot:

   To determine, as experience increases, whether the number of successful launches increases with respect to the Orbit type.

7. *Average Success Rate per Year* line plot:

   To determine the yearly trend and see whether success rate kept increasing from 2013 to 2020.



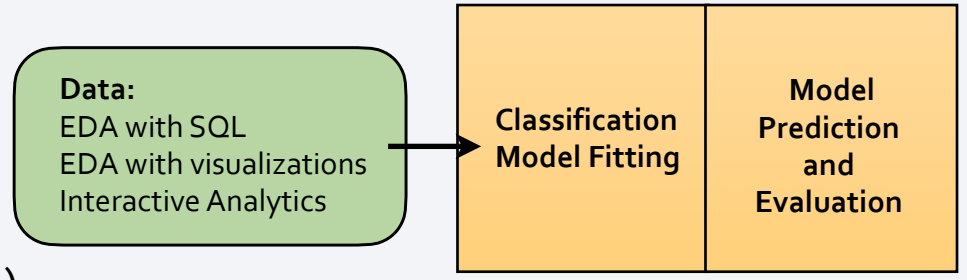
Data Visualization: Feature Engineering

- Feature Selection
- Transforming Categorical features to Numerical by creating dummies with pd.get_dummies()
- Cast all variables as 'float64' using OneHotEncoder
- Save data as a civ file using pd.to_csv

GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M2L2_edadataviz.ipynb

EDA with SQL

SQL queries:



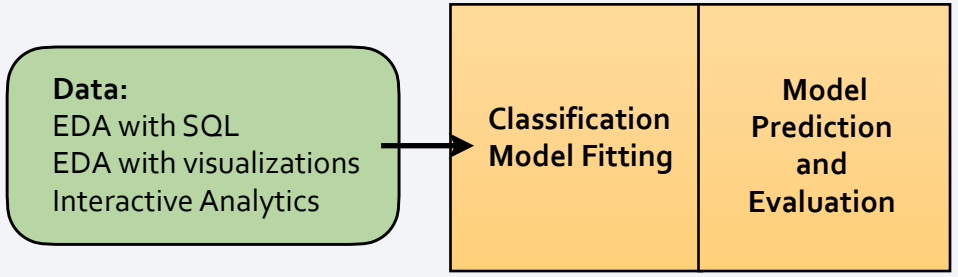
- Names of the unique launch sites in the space mission.
- 5 records where launch sites begin with the string 'CCA' display.
- Total payload mass carried by boosters launched by NASA (CRS).
- Average payload mass carried by booster version F9 v1.1
- First successful landing outcome in ground pad's date.
- Names of boosters with successful drone ship landing and payload mass between 4000 and 6000 kgs.
- Total number of successful and failure mission outcomes.
- Booster versions that have carried the maximum payload mass.
- Landing outcomes in drone ship failures, booster versions and launch_site for the months in year 2015 display.
- Count of landing outcomes between t 2010-06-04 and 2017-03-20 in descending order.

GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M2L1_jupyter-labs-eda-sql-coursera_sqllite%20.ipynb

Build an Interactive Map with Folium I

AIM 1: Locating all launch sites in the map

- Initialising map (site-map) with NASA Johnson Space Center at Houston, Texas as center.
- Using folium.Circle() and folium.map.Marker(), adding highlighted circle area and text label in the NASA Center.
- Using folium.Circle() and folium.map.Marker(), iterating through launch site coordinates to add all launch sites in the map with highlighted area and text label.



AIM 2: Locating successful launches from launch sites in the map

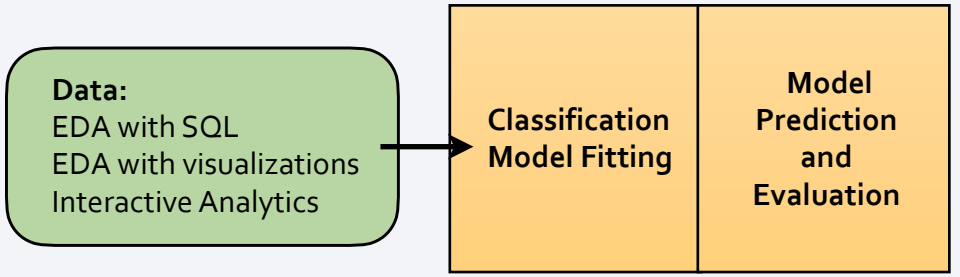
- Using of MarkerCluster(), folium.Marker() and color-coded column ("red" = unsuccessful/ "green"= successful) from "class" to show success and failure mission outcomes.
- Iterating through launch site coordinates and, using the color-coded column, adding marker cluster with an icon property to show if the launch was successful or unsuccessful per cluster.

GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M3L1_lab_jupyter_launch_site_location.ipynb

Build an Interactive Map with Folium II

AIM 3: In order to explore an analyse the proximities of launch sites

- Using predefined calculate_distance function, calculate distance to the closest railway from site KSCLC39A and to the closest coastline from site CCAFS SLC-40.
- Adding folium.Marker() with icon property and folium.Polyline() to mark distances from sites to the closest railway and coastline.

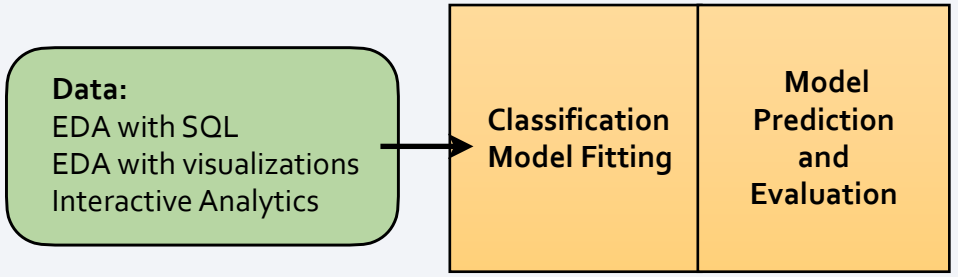


GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M3L1_lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash I

Dropdown list:

- Adding a dropdown list in the dash application layout.
- Aim: Enabling specific launch site selection and including an "all sites" option.



Pie Chart Function and callback:

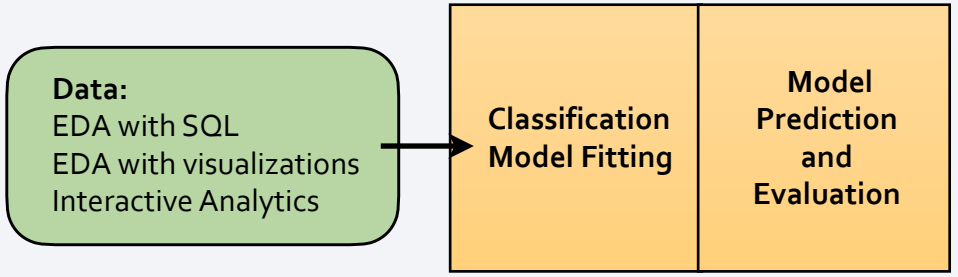
- def get_pie_chart(): creates pie chart that shows either the total successful launch count when "all sites" is selected or the successful launch count when each specific launch site is selected using the dropdown list.
- @app.callback(): defines call back function with the count output dependent on dropdown input.
- Aim: Comparing counts of success vs. failure mission outcome by site vs. all sites.

GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M3L2_spacex-dash-app.py

Build a Dashboard with Plotly Dash II

Payload Slicer

- Adding a slicer in the dash application layout.
- Aim: Enabling selection of payload range in Kg from 0 to 10k.



Get Scatter Function and callback

- def get_scatter(): creates a scatter plot that shows successful and unsuccessful launches by booster type dependent on dropdown list and payload range slicer selections.
- @app.callback(): define call back function with the launches output by booster type dependent on dropdown selection input and on payload slicer range input.
- Aim: Displaying payload per site impact on successful outcome missions by booster type.

GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M3L2_spacex-dash-app.py

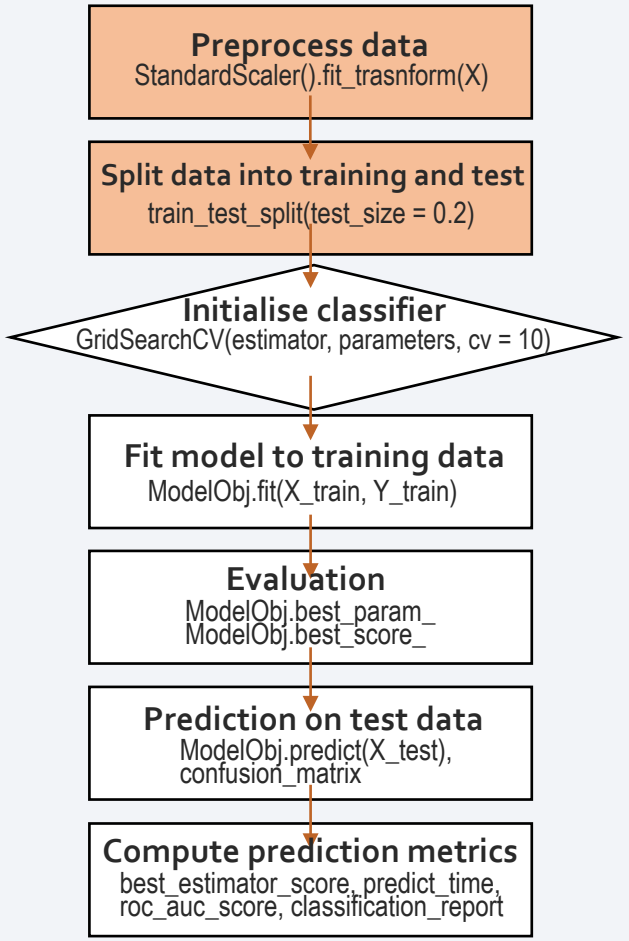
Predictive Analysis (Classification)

For the 4 models:

- Preprocessing data
- Splitting data into Training (80%) and Test (20%) sets

Per model:

- Use pre-defined parameter grids for the 4 different classifiers: Logistic Regression, SVM, Decision Tree, KNNs
- Initialise model within cross-validation pipe (cv=10)
- Fit model to Training data
- Output: Best Parameters and Accuracy Score
- Evaluate model on Test data
- Output: accuracy score and confusion matrix
- Added metrics: model_pred_time, roc_auc_score, classification_report



GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M4L2_SpaceX_Machine%20Learning%20Prediction_Part_5_TreeAuto_Threshold.ipynb

Section 2

Insights drawn from EDA

Results: EDA

📌 SQL queries EDA results:

Each query built context around what may affect the **reusability** of the first stage.

1. **Understanding Launch Locations and Outcomes**
   - Identified launch sites
   - Count successful vs. failed landings per site
2. **Understanding Payload, Booster, and Orbit Characteristics**
   - Payload mass relationship to landing success
   - Explored booster versions and orbits with respect to outcomes
3. **Discovering Conditions Related to Landing Success**
   - Correlate launch outcome (Landing _Outcome) with payload, booster version, orbit, etc.
   - Found conditions most/least associated with success

📌 Visualizations EDA results:

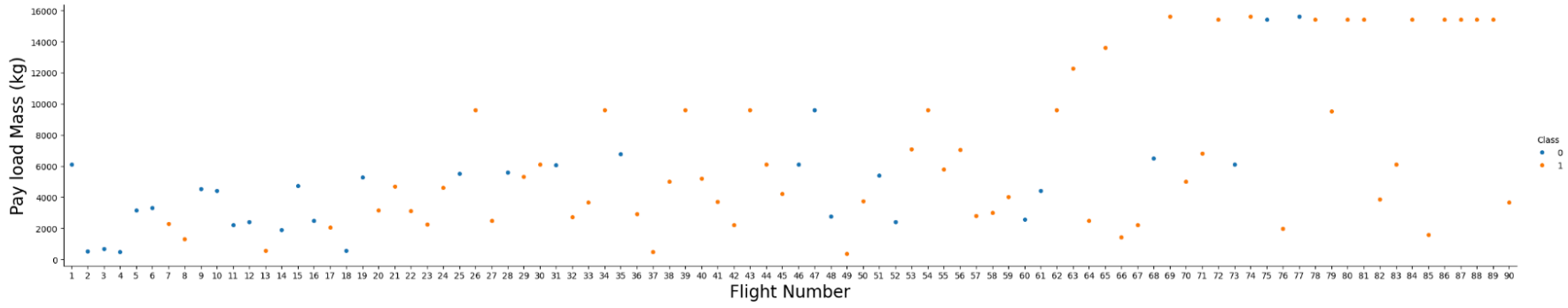
Visualizations served as counterpart to previous SQL querying.

1. **Further exploring relationships between key mission attributes**
   - Payload, Orbit, Launch Site, Fight Number.
2. **Identifying patterns or distributions that may influence landing success**
   - Correlations
3. **Helping prepare features for a machine learning model to predict successful landings**
   - Feature engineering

Flight Number vs. Payload Mass

- **Purpose:** To check whether payload over mission experience (as indicated by flight number) correlates with landing success.
- **Insight:** Later flight numbers show more successful landings with heavier payloads, suggesting SpaceX improved landing success with increasing payloads over time.

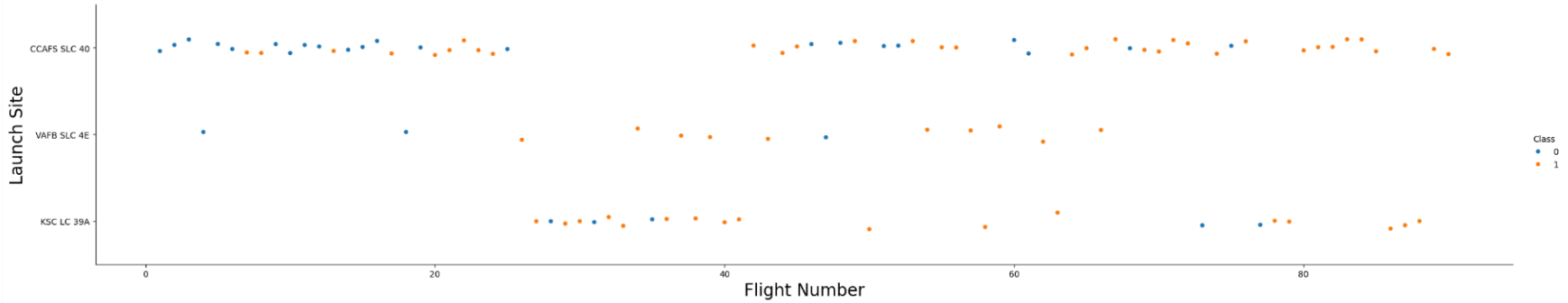
```
]: sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
   plt.xlabel("Flight Number",fontsize=20)
   plt.ylabel("Pay load Mass (kg)",fontsize=20)
   plt.show()
```



Flight Number vs. Launch Site

- **Purpose:** To check whether the launch site over mission experience (as indicated by flight number) correlates with landing success.
- **Insight:** Later flight numbers per launch site show more successful landings, suggesting SpaceX improved landing success over time regardless of the site.

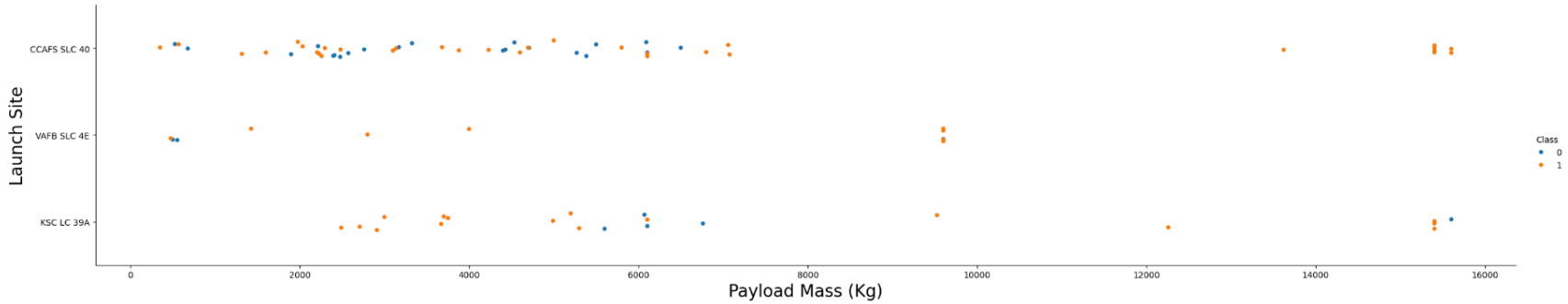
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the
sns.catplot( x = "FlightNumber", y= "LaunchSite", hue = "Class", data = df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Payload vs. Launch Site

- **Purpose:** To check whether the launch site as payload increased correlates with landing success. This plot supports a SQL query.
- **Insight:** The plot shows that, with the exception of a case, high payload missions were successful regardless of the site. However, most data is placed below 6000 kg and, in this case, the pattern shows that the relationship between payload and successful landing is different depending on the launch site and should be further looked into in interactive analytics.

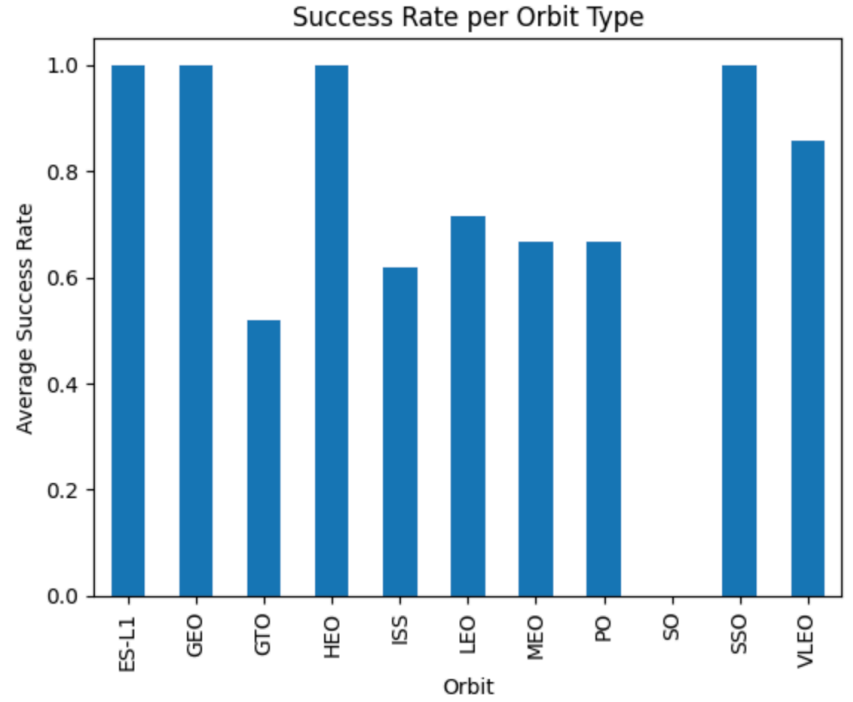
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be
sns.catplot( x = "PayloadMass", y= "LaunchSite", hue = "Class", data = df, aspect = 5)
plt.xlabel("Payload Mass (Kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Success Rate vs. Orbit Type

- **Purpose:** To analyze which orbit types (LEO, GTO, etc.) are associated with successful landings.
- **Insight:** Most successes are clustered in Es-L1, GEO, HEO, SSO and followed by VLEO. The fact that there are 3 geosynchronous orbits must have implications in successful first-stage recovery.

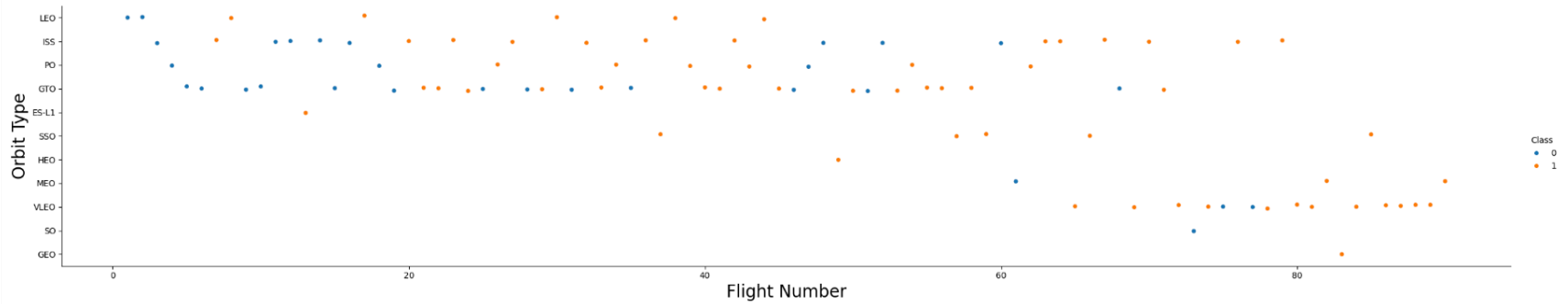
```
# HINT use groupby method on Orbit column and get the mean of Class column
orbit = df.groupby("Orbit")["Class"].mean()
orbit.plot(kind = "bar")
plt.xlabel("Orbit")
plt.ylabel("Average Success Rate")
plt.title("Success Rate per Orbit Type")
plt.show()
```



Flight Number vs. Orbit Type

- **Purpose:** To check whether the orbit type over mission experience (as indicated by flight number) correlates with landing success.
- **Insight:** Most successes are clustered in LEO and ISS orbits, suggesting lower-altitude or simpler missions may be more successful for first-stage recovery with experience. With later data, VLEO seems to follow the same pattern.

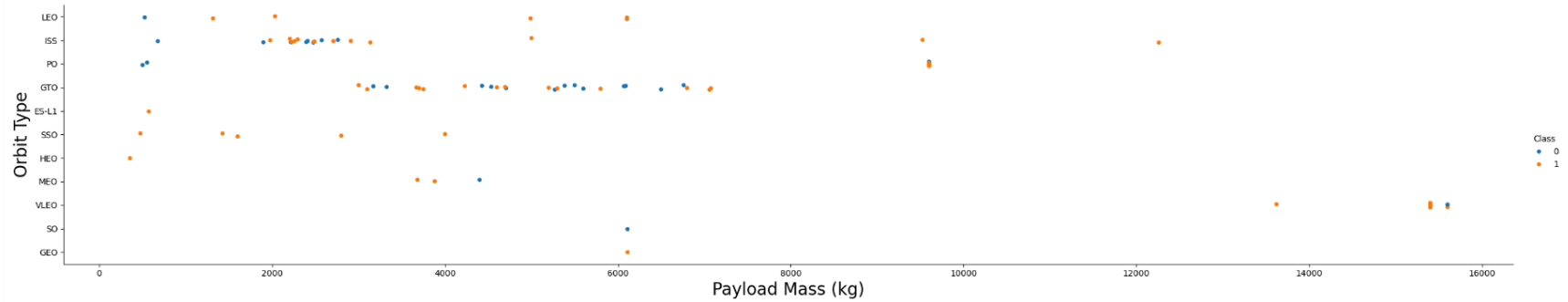
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class v
sns.catplot( x = "FlightNumber", y= "Orbit", hue = "Class", data = df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit Type",fontsize=20)
plt.show()
```



Payload vs. Orbit Type

- **Purpose:** To check whether the orbit type as payload increases in missions (correlates with landing success.
- **Insight:** Again, the plot shows that high payload missions were successful regardless of the orbit type. However, most data is placed below 6000 kg and, in this case, the pattern shows that the relationship between orbit and successful landing is different depending on the orbit. This is clearly shown for GTO where both successes and failures show not to be related with heavier payload.

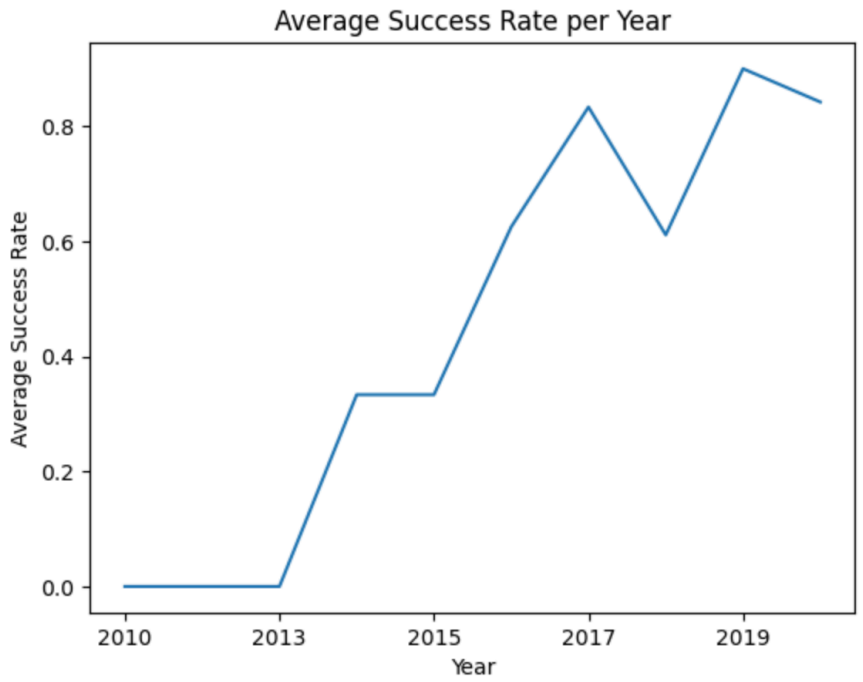
```
# Plot a scatter point chart with x axis to be Payload Mass and y axis to be the Orbit, and hue to be the class v
sns.catplot( x = "PayloadMass", y= "Orbit", hue = "Class", data = df, aspect = 5)
plt.xlabel("Payload Mass (kg)",fontsize=20)
plt.ylabel("Orbit Type",fontsize=20)
plt.show()
```



Launch Success Yearly Trend

- **Purpose:** To observe landing trends over time.
- **Insight:** Clear upward trend in landing success from 2013 to 2020, showing consistent progress in reusability technology.

```
# Plot a line chart with x axis to be the extracted year and y axis to be t
success_rate = df.groupby("Date")["Class"].mean()
success_rate.plot()
plt.title("Average Success Rate per Year")
plt.xlabel("Year")
plt.ylabel("Average Success Rate")
plt.show()
```



Results: EDA

SQL queries EDA results:

Each query built context around what may affect the **reusability** of the first stage.

1. **Understanding Launch Locations and Outcomes**
   - Identified launch sites
   - Count successful vs. failed landings per site
2. **Understanding Payload, Booster, and Orbit Characteristics**
   - Payload mass relationship to landing success
   - Explored booster versions and orbits with respect to outcomes
3. **Discovering Conditions Related to Landing Success**
   - Correlate launch outcome (Landing _Outcome) with payload, booster version, orbit, etc.
   - Found conditions most/least associated with success

Visualizations EDA results:

Visualizations served as counterpart to previous SQL querying.

1. **Further exploring relationships between key mission attributes**
   - Payload, Orbit, Launch Site, Fight Number.
2. **Identifying patterns or distributions that may influence landing success**
   - Correlations
3. **Helping prepare features for a machine learning model to predict successful landings**
   - Feature engineering

All Launch Site Names

- **Purpose**: Identifies the different locations SpaceX uses for launches.
- **Explanation**: Finds locations that will be used in interactive analytics and helps further analysis on whether location influences landing success.

```
%%sql
select distinct "Launch_Site"
from SPACEXTABLE;
```

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

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

Launch Site Names Begin with 'CCA'

- **Purpose:** Views sample data for Cape Canaveral Air Force Station (indicated by %CCA%).
- **Explanation:** Offers a focused preview of launches from a specific site to observe patterns.

```
%%sql
select *
from SPACEXTABLE
where "Launch_Site" like "%CCA%"
limit 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (p
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 (p
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

Total Payload Mass

```
%%sql
select "Customer", SUM("PAYLOAD_MASS__KG_") as "total_mass_carried"
from SPACEXTABLE
where "Customer" == "NASA (CRS)"
group by "Customer";
```

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

Customer	total_mass_carried
NASA (CRS)	45596

- **Purpose:** Computes total payload carried for NASA's CRS missions.
- **Insight:** Highlights how much payload NASA entrusted to SpaceX, this is important as payload is critical for missions and revenue generating.

Average Payload Mass by F9 v1.1

```
%%sql
select "Booster_Version", AVG("PAYLOAD_MASS__KG_") as "average_payload_mass"
from SPACEXTABLE
where "Booster_Version" == "F9 v1.1"
group by "Booster_Version";
```

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

Booster_Version	average_payload_mass
F9 v1.1	2928.4

- **Purpose:** Finds the payload mass in kg the F9 v1.1 booster version carries on average.
- **Insight:** It could link the F9 v1.1 booster version to payload efficiency and relate it to landing reliability. This also helps further interactive analysis.

First Successful Ground Landing Date

```
%%sql
select MIN("Date") as "first_successful_groundpad_landing"
from SPACEXTABLE
where "Landing_Outcome" == "Success (ground pad)";
```

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

first_successful_groundpad_landing
2015-12-22

- **Purpose:** Identifies the earliest ground pad landing success.
- **Insight:** Marks a milestone in SpaceX's reusability history.

Successful Drone Ship Landing with Payload between 4000 and 6000

- **Purpose:** Lists booster types that succeeded in drone ship landings with medium-heavy payloads.
- **Insight:** Important for understanding which boosters are reliable for missions future missions (FT). It also helps to show that technical improvements in newer versions help landings.

```
%%sql
select "Booster_Version" as "booster_name"
from SPACEXTABLE
where ("Landing_Outcome" == "Success (drone ship)")
    and ("PAYLOAD_MASS__KG_" between 4000 and 6000);
```

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

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

Total Number of Successful and Failure Mission Outcomes

- **Purpose:** Summarizes how often each mission outcome occurred.
- **Insight:** Provides a success/failure rate snapshot for all missions.

```
%%sql
select "Mission_Outcome", count(*) as number
from SPACEXTABLE
group by "Mission_Outcome";
```

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

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

Boosters Carried Maximum Payload

- **Purpose**: Identifies the type of boosters that launched the heaviest payload.
- **Explanation**: Useful for matching high-capacity boosters with successful landings and improving technology of booster with heavy payloads.

```
%%sql
select "Booster_Version"
from SPACEXTABLE
where "PAYLOAD_MASS__KG_" == (select MAX("PAYLOAD_MASS__KG_")
                              from SPACEXTABLE);
```

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

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

2015 Launch Records

```
%%sql
select substr(Date, 6,2) as "month", "Landing_Outcome", "Booster_Version", "Launch_Site"
from SPACEXTABLE
where "Landing_Outcome" == "Failure (drone ship)"
    and substr(Date,0,5)='2015' ;
```

```
 * 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

- **Purpose:** Details failed drone landings in 2015 by month and booster.
- **Explanation:** Helps pinpoint time periods and booster version associated with landing failures -both of them on drone ship and not on ground pad. It also draws attention to the site for further interactive analysis.

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

- **Purpose:** Gives a full count of landing outcomes within a defined period.
- **Explanation:** It measures landing success trends over 7 years. It also shows no failures in ground pad landings: ground pad landings then show high success rates, while drone ship landings are more mixed supporting the idea that drone landings are riskier.

```
%%sql
select "Landing_Outcome",COUNT("Landing_Outcome") as "count"
from SPACEXTBL
where "Date" between '2010-06-04' and '2017-03-20'
group by "Landing_Outcome"
order by COUNT("Landing_Outcome") desc;
```

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

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

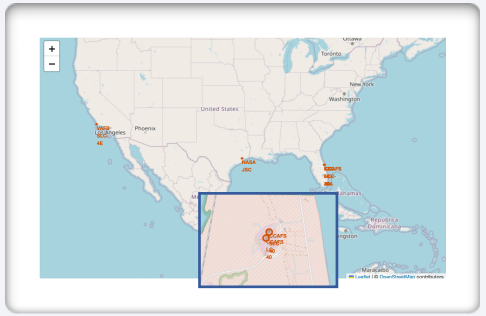
Section 3

Launch Sites Proximities Analysis

Results: Interactive Analytics Demo

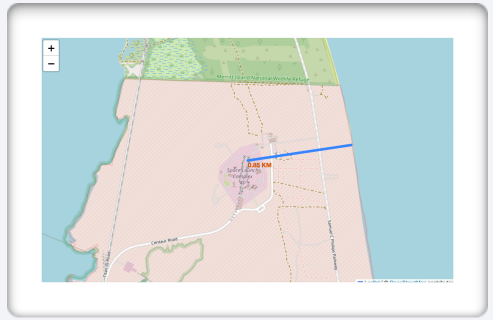
- Folium: Impact of Launch Site location on successful missions

Markers:



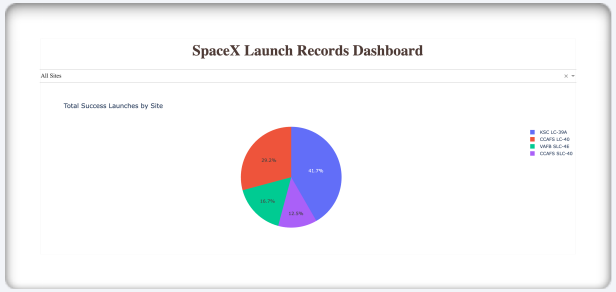
Clusters:

Proximities:

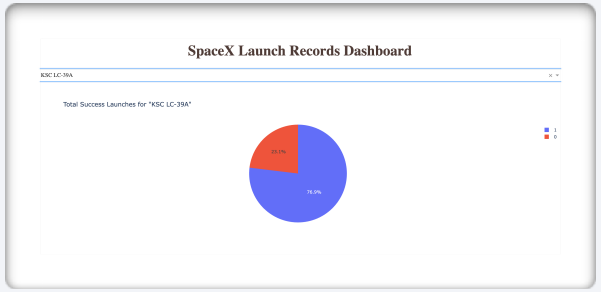


- Plotly Dash: Impact of payload on successful missions per booster and per site

Success rate for all sites:



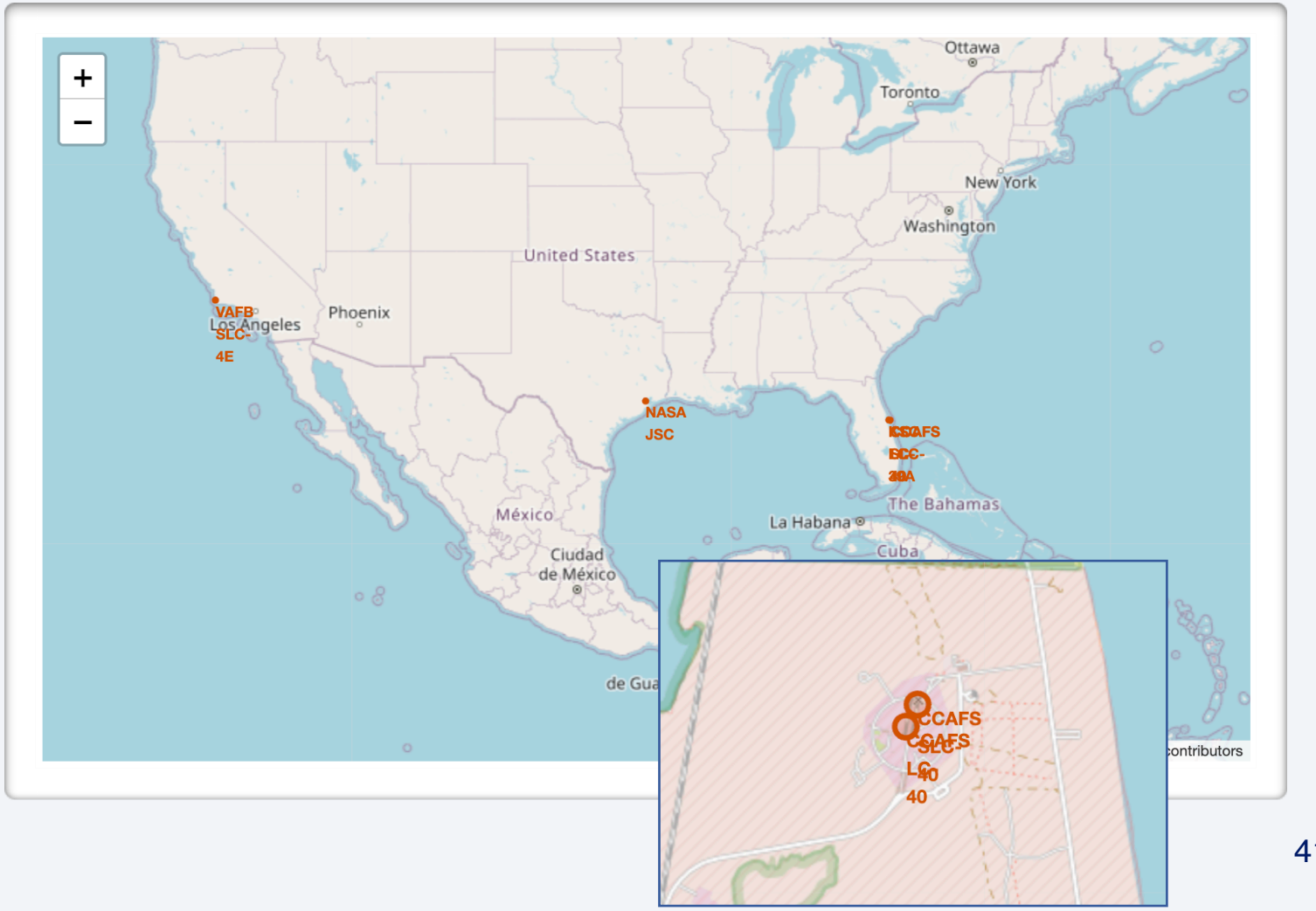
Success rate for KSC LC-39A:



Payload range (3k-6k kg) vs. Success by booster:

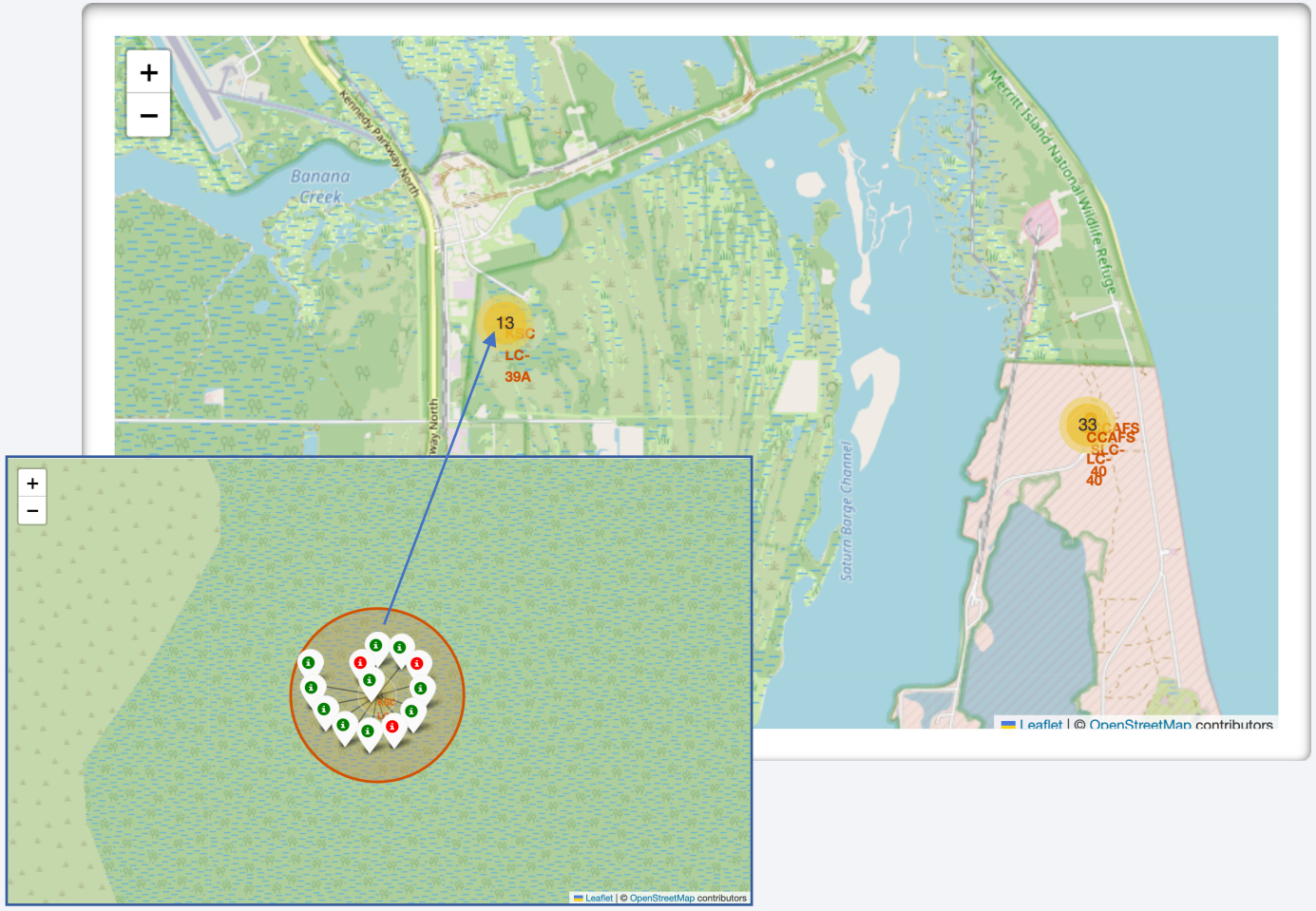
Folium: Launch Sites location Markers

- **Purpose**: Place launch site locations on the map.
- **Explanation**: Easy identification for further interaction. Launch sites of interests are all located near the ocean.



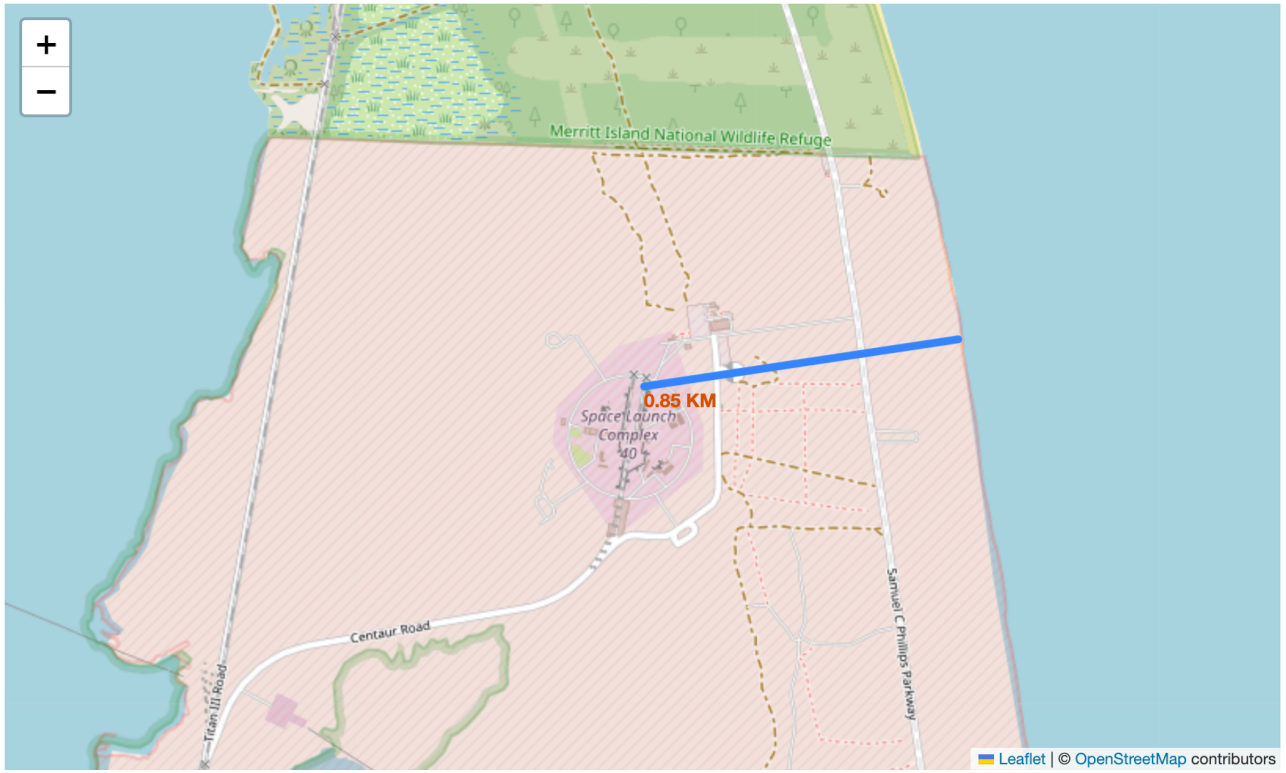
Folium: Color-labeled Launch Outcomes

- **Purpose:** Show successful vs. unsuccessful landings per location.
- **Explanation**: Zooming in on a color-coded cluster to show successful vs. unsuccessful landings allows for faster future launching decision-making which can have an impact in reusability and thus, future costs.



Folium: Coastline distance from CCAFS SLC-40

- **Purpose**: Mark the distance between site CCAFS SLC-40 and the closest coastline.
- **Explanation:** Calculating the distance to the ocean could help decisions to improve ground infrastructure or choose the type of mission.



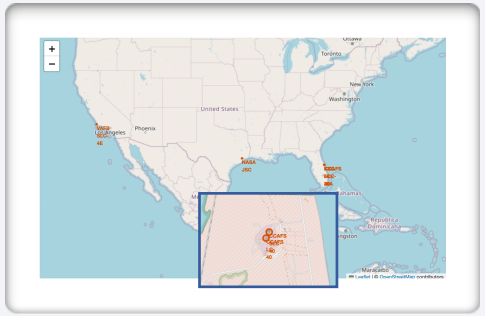
Section 4

Build a Dashboard with Plotly Dash

Results: Interactive Analytics Demo

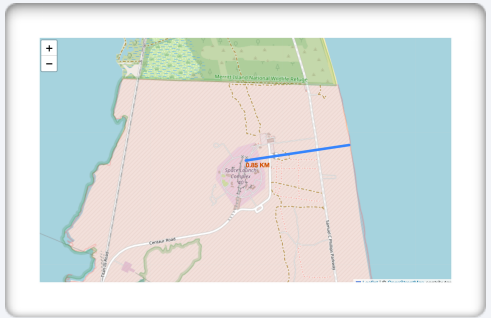
- Folium: Impact of Launch Site location on successful missions

Markers:



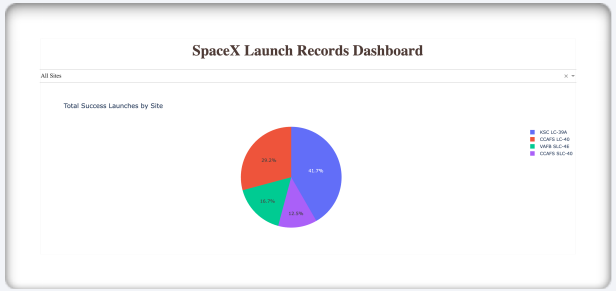
Clusters:

Proximities:

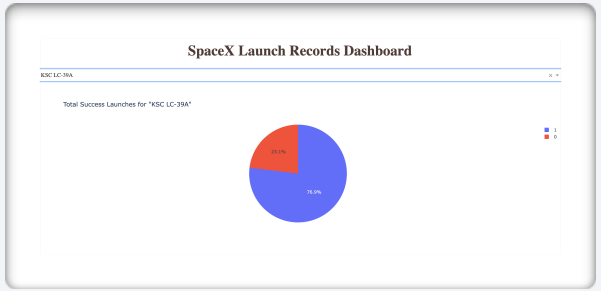


- Plotly Dash: Impact of payload on successful missions per booster and per site

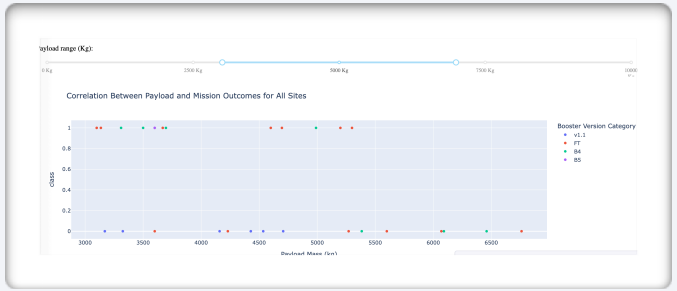
Success rate for all sites:



Success rate for KSC LC-39A:

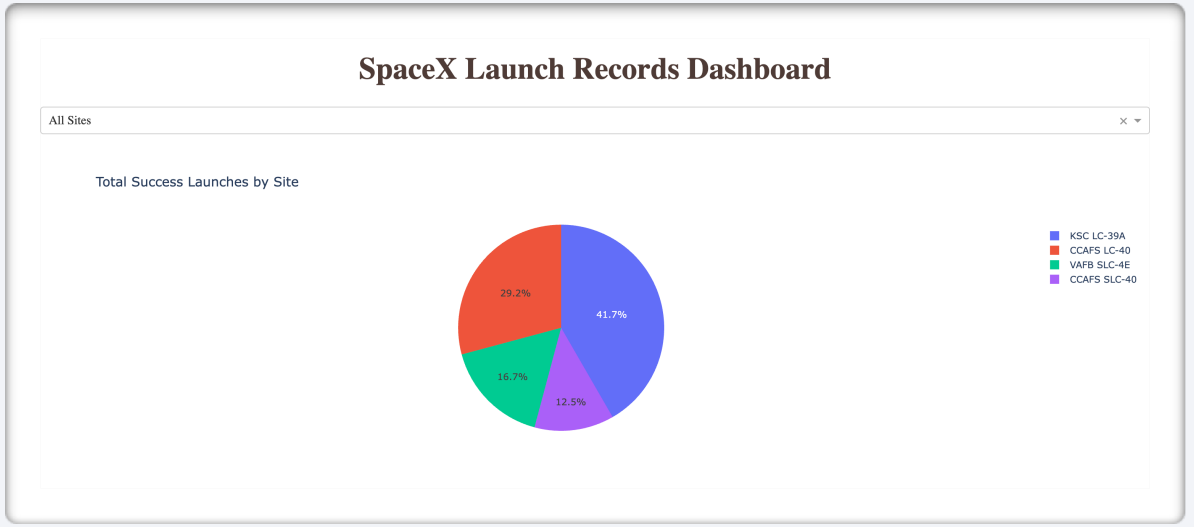


Payload range (3k-6k kg) vs. Success by booster:



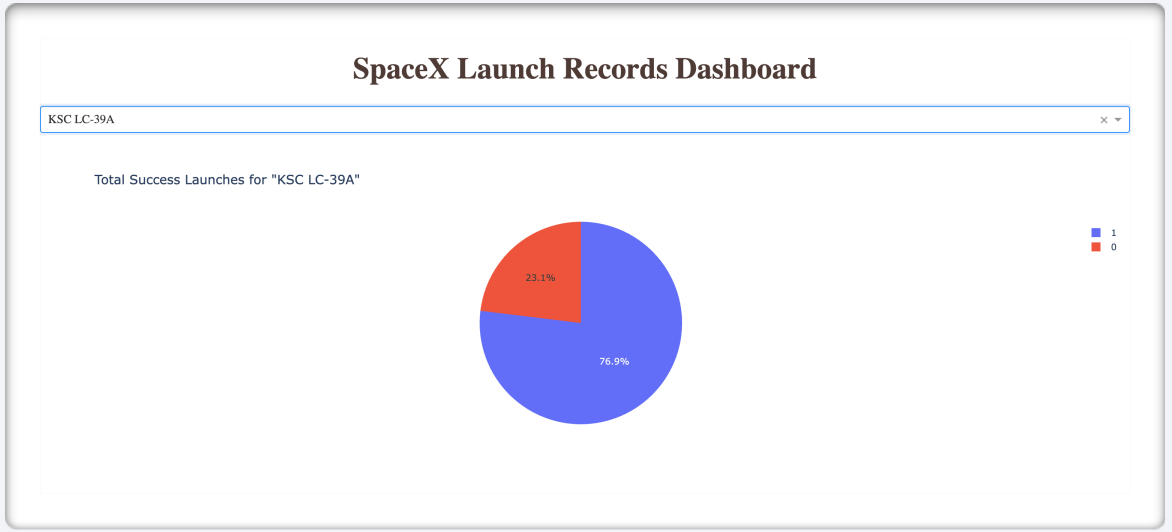
Plotly Dash: Launch Success for All Sites

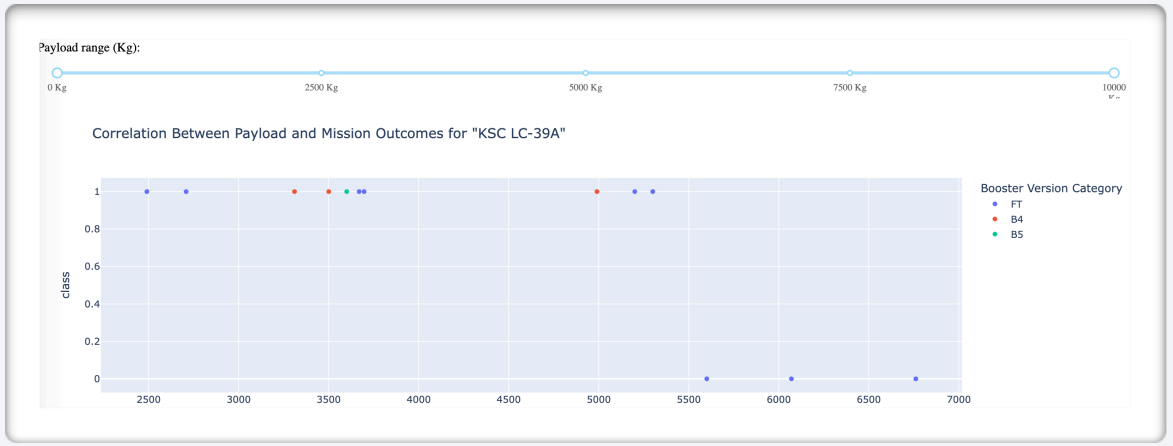
- **Purpose**: Show launch success rate for all sites
- **Findings**:
  - KSC LC-39A shows the highest success rate (look at cluster in Folium.map)
  - Most common booster versions between 0-10K kg linked to success are FT and B4.
  - Only 16.7% of success rate comes from the West coast, from site VAFB SLC-4E (look at site location in Folium.map) -supporting the idea of East coast launching leveraging the Earths' rotational velocity and help reduce costs.



Plotly Dash: Launch Success for KSC LC 39A

- **Purpose**: Show launch success rate for site KSC LC-39A
- **Findings**:
  - KSC LC-39A shows a 76% success rate.
  - Most common booster versions between 0-10000 kg linked to success are FT and B4.
  - Failure of the FT is linked to payload mass above 5500 kg: considering the Folium.cluster, here is an explanation that could account for the 3 red markers in the cluster.
  - No B4 booster data available above 5500 kg





Plotly Dash: Payload vs. Launch Outcome

- **Purpose:** Show launch success rate for all sites with Payload range 3k-6k kg to cover average payload mass carried by a NASA mission and 4k to 6k range of the SQL query

- **Findings**:
  - For payload higher than the average mass carried by a NASA mission (2900 kg) and slightly lower than 4000 kg, boosters FT and B4 are linked to successful missions. B4 does not show a single failed mission.
  - For payload between 4000 and 6000 kg (given by the SQL query), the FT booster is clearly linked to successful missions when payload is higher than 4500 and below 5500 kg. There is 1 successful account of the B4 booster.
  - Any launches for payload beyond 5500 kg lead to a failed mission regardless of the booster.

Section 5

Predictive Analysis (Classification)

Results: Predictive Analysis

- **Purpose 1**: Train 4 different classifiers (Logistic Reg, SVM, Decision Tree (with Max_Features = 'Auto'), KNNs) to predict success vs. failure of mission outcomes.

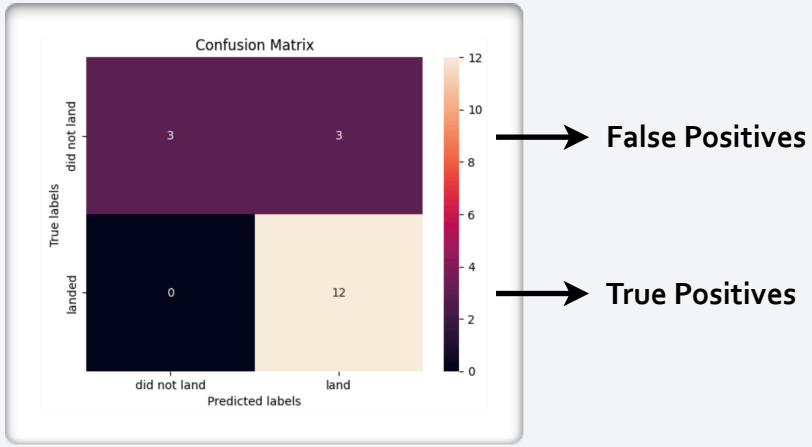
- **Results:** The 4 models rendered almost the same results.

  Very similar accuracy in training and same accuracy in test data (83%). The gap in accuracy between training and test is fairly normal, not necessarily a sign of overfitting.

  Added metrics: predict_time was very similar for the 4 classifiers; ROC-AUC score was the same (0.75) for the 4 classifiers.

  The 4 models provided the same confusion matrix with 12 True Positives and 3 False Positives.

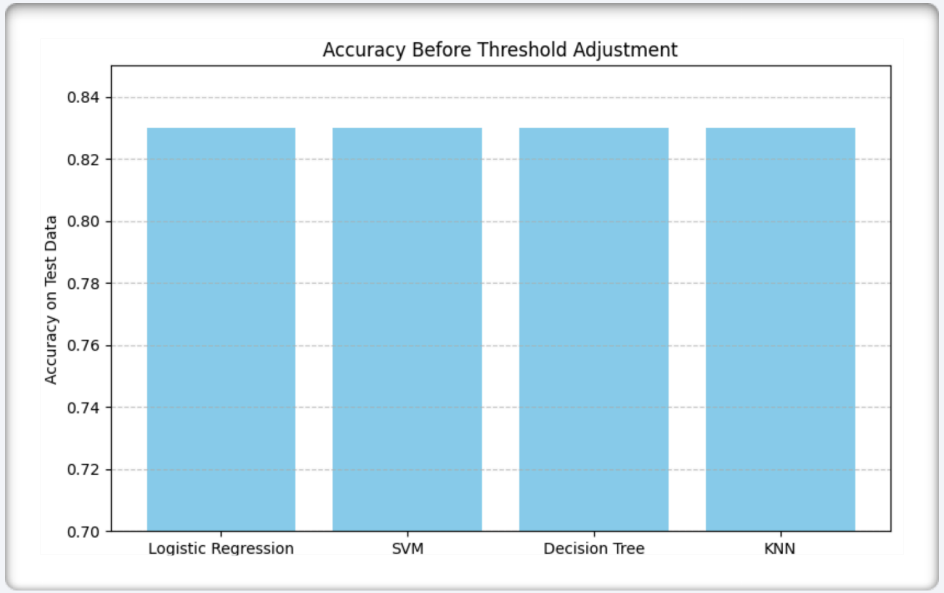
- **Purpose 2**: Tuning 1 of the classifiers (Decision Tree with Max_Features = 'Log2') to focus on **increasing precision** = reducing False Positives



	Accuracy Training	Accuracy Test	Time	ROC-AUC score	True Positives	False Positives	Precision
Logistic Reg	0.846	0.833	0.0009	0.75	12	**3**	**0.80**
SVM	0.848	0.833	0.0	0.75	12	**3**	**0.80**
Decision Tree (Auto)	0.843	0.833	0.0	0.75	12	**3**	**0.80**
Decision Tree (Log2)	0.875	**0.77**	0.0020	0.75	10	**2**	**0.83**
KNN	0.848	0.833	0.001	0.75	12	**3**	**0.80**

Classification Accuracy

- The 4 classifiers accuracy on Training data were very similar:
  - Logistic Regression: **85%**
  - SVM: **85%**
  - Decision Tree: **84%**
  - KNNs: **85%**
- The 4 classifiers showed the same accuracy on Test data:
  - Logistic Regression: **83%**
  - SVM: **83%**
  - Decision Tree: **83%**
  - KNNs: **83%**
- The dataset is small; some difference in accuracy between Training and Test sets is expected, not necessarily a sign overfitting.
- ☑ Classification reports on Test data show that there is no difference among the 4 different classifiers.



	precision	recall	f1-score	support
0	1.00	0.50	0.67	6
1	0.80	1.00	0.89	12
accuracy			0.83	18
macro avg	0.90	0.75	0.78	18
weighted avg	0.87	0.83	0.81	18

GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M4L2_SpaceX_Machine%20Learning%20Prediction_Part_5_TreeAuto_Threshold.ipynb

Confusion Matrix

- The 4 classifiers Confusion Matrix displayed the same results:

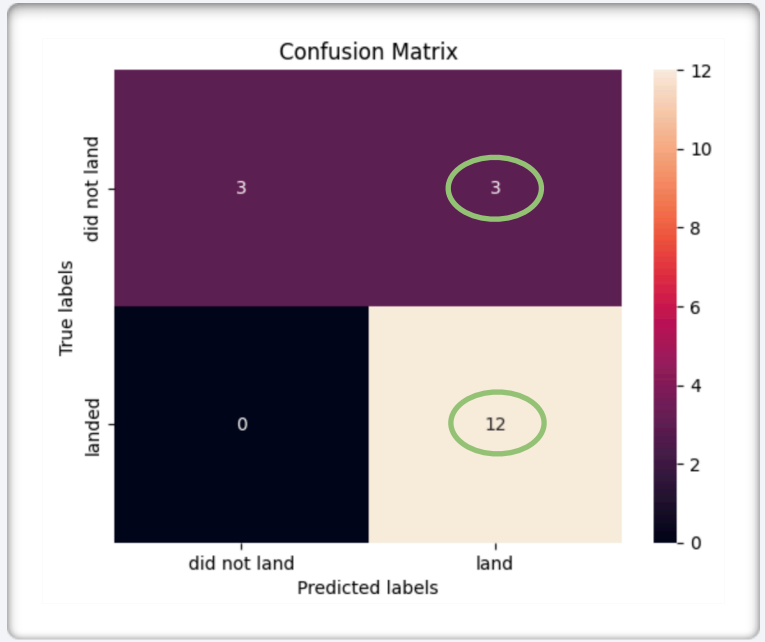
  True Positives = Predicted to land and landed: **12**

  False Positives = Predicted to land but not landed: **3**

  True Negatives = Predicted not to land and not landed: **3**

  False Negatives: = Predicted not to land but landed: **0**

☑ With the same accuracy on Test sets and the same number of False Positives, no model is better than the other.



Suggested follow-up work:

- Predicting success when the mission fails (False Positive) is costly; predicting failure when it would succeed (False Negative) costs a missed opportunity -but is less risky.
- Consider tuning parameters to find a model with lower overall accuracy but higher precision to minimise financial risk.

	precision	recall	f1-score	support
0	1.00	0.50	0.67	6
1	0.80	1.00	0.89	12
accuracy			0.83	18
macro avg	0.90	0.75	0.78	18
weighted avg	0.87	0.83	0.81	18

GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M4L2_SpaceX_Machine%20Learning%20Prediction_Part_5_TreeAuto_Threshold.ipynb

Precision vs. Accuracy I

- As suggested by an entry in the fora, replacing parameters_depth 'auto' to 'log2' in the decision tree, increased precision (83%) but decreased accuracy on Test data (70%):

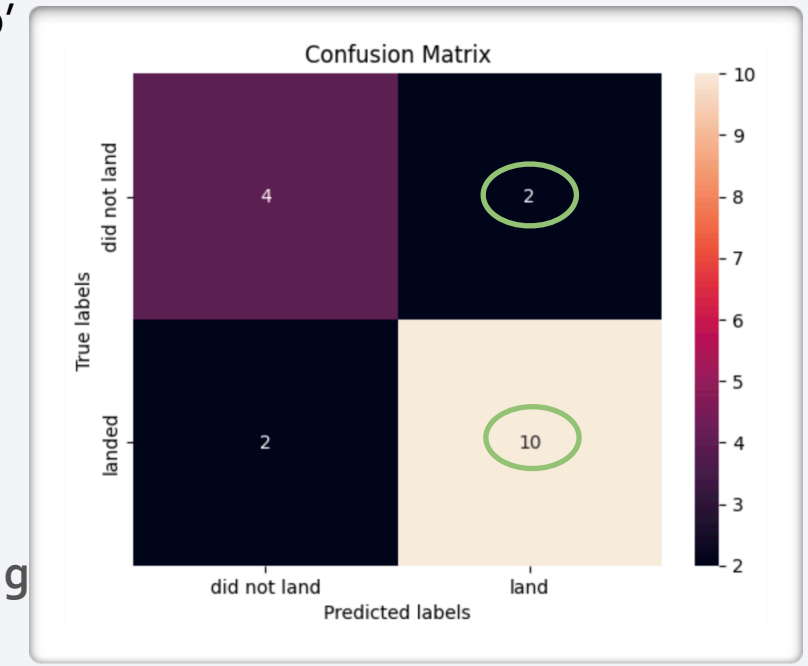
  True Positives = Predicted to land and landed: **10**

  False Positives = Predicted to land but not landed: **2**

  True Negatives = Predicted not to land and not landed: **4**

  False Negatives: = Predicted not to land but landed: **2**

☑ Even though a model with fewer False Positives (but more False Negatives) could be greenlighted, the difference between accuracy on Training (88%) vs. Test (78%) sets suggests that **this model is memorising data and not generalising well.**



Suggested follow-up work:

- More data: 18 samples are too few.
- Tune the probability threshold in Logistic Regression or SVM classifiers by increasing it to 0.7/ 0.8 on Test sets in order to improve precision without overfitting.

	precision	recall	f1-score	support
0	0.67	0.67	0.67	6
1	0.83	0.83	0.83	12
accuracy			0.78	18
macro avg	0.75	0.75	0.75	18
weighted avg	0.78	0.78	0.78	18

GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M4L1_SpaceX_Machine%20Learning%20Prediction_Part_5_TreeLog2.ipynb

Precision vs. Accuracy II

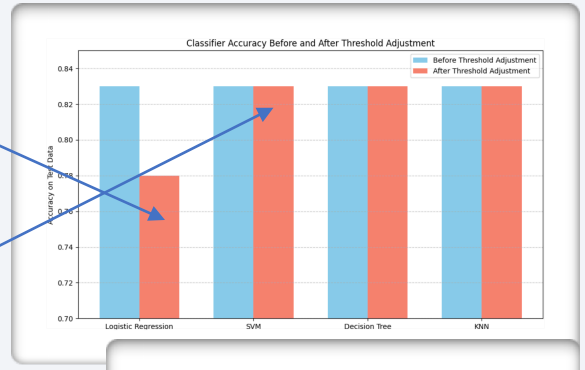
- Increasing the Logistic Regression classifier's probability threshold (0.7) on Test data decreased the number of False Positives but showed lower Test accuracy (78%) compared to it with original settings (83%)
- Increasing the SVM classifier's probability threshold (0.8) on Test data decreased the number of False Positives while keeping the original Test accuracy (83%).

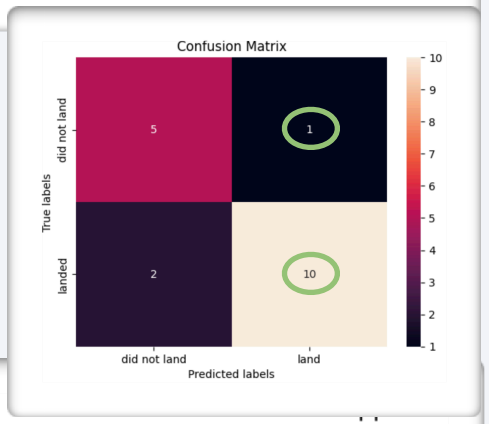
  True Positives = Predicted to land and landed: **10**

  False Positives = Predicted to land but not landed: **1**

  True Negatives = Predicted not to land and not landed: **5**

  False Negatives: = Predicted not to land but landed: **2**





Insights:

- A model with higher precision (at the expense of recall) while keeping the original accuracy on Test data can be achieved by tuning the probability threshold.
- After evaluation and tuning, the SVM classifier with 83% accuracy is the best model to predict successful landing and thus, reusability while aligned with a business goal.

	precision			
0	0.71	0.83	0.77	6
1	0.91	0.83	0.87	12
accuracy			0.83	18
macro avg	0.81	0.83	0.82	18
weighted avg	0.84	0.83	0.84	18

GitHub URL: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M4L2_SpaceX_Machine%20Learning%20Prediction_Part_5_TreeAuto_Threshold.ipynb

Conclusions

- From EDA, some features are more likely to correlate with landing success and consequential reusability:
  - ☑ Flight Number: Later flight numbers show increased success — indicating mission experience and learning curve matter.
  - ☑ Booster Version: Newer versions (e.g., F9 FT, B4/5) are more successful, suggesting technological improvements contribute to better outcomes.
  - ☑ Launch Site: Sites like KSC LC 39A and CCAFS SLC 40 have higher success rates, implying infrastructure or mission profiles at these sites favor recovery. Sites from which launching eastward is required entail less fuel would be needed and might imply better chances of recovery. This would also make sense for orbital mechanics, geostationary and low Earth orbits.
  - ☑ Landing Pad: Ground pad landings are more often successful than drone ship landings, indicating landing method/location strongly affects success.
  - ☑ Year (Date): Successes clearly increase over time — SpaceX's iterative improvement is a major driver.
  - ☑ Orbit Type: Different Orbits showed up (LEO, ISS, GTO, GEO..) associated with more successful landings, possibly due to lower complexity, or altitude or being geosynchronous.
- From EDA, some other features are less likely or weakly correlated with landing success -or their relationship is more complex as they interact with other features:
  - ☑ Payload Mass: Success spans a wide payload range. There were fewer cases of extremely heavy payloads and showed successes. With lower and medium payloads, where most of the data lies, successes were linked to the booster version as payload increased within the range.
  - ☑ Customer: While large clients like NASA fly often, there's no clear link between customer and success. It's more about mission type than client.
  - ☑ Mission Outcome: Strongly related to success/failure, but it is a label rather than a predictive feature (not useful for prediction input).

Conclusions

- From predictive analytics, further tuning was required upon evaluation of the models as they originally were, with preset parameters:
  - ☑ The 4 models with given parameters rendered the same prediction accuracy (83%) and the same precision (80%). Their confusion matrices also showed the same proportion of True Positives (12) and False Positives (3).
  - ☑ Even though more metrics were added (classification reports, prediction time and ROC-AUC scores) looking for further distinction, these results made choosing the best model impossible.
  - ☑ Tuning the Decision Tree max_features from the given "auto" to "log 2" rendered a model with higher precision (83%) but it affected generalisation as it lowered its predicting accuracy to 78%. The difference between accuracy on Training (88%) vs. accuracy on Test (78%) suggested that the model was memorising data and thus, overfitting.
  - ☑ Adjusting the probability threshold from 0.5 to 0.8 on Test data in the SVM classifier rendered a model with higher precision (91%.) while keeping the original accuracy on Test data (83%); that is, the number of False Positives decreased -although it did so at the expense of recall (71%)
  - ☑ Given that this is a case in which False Positives are very costly -while False negatives are not ideal but they only cost an opportunity, minimising the financial risk should be the goal. Taking this into account and given that the highest prediction accuracy reached by any of the models was 83%, **the SVM classifier with an 0.8 probability threshold, matching the 83% prediction accuracy, is the best model as it rendered the highest precision, 91%.**

Appendix

There are 2 Module 4 notebooks, M4L1 and M4L2 uploaded on GitHub as per request.

- M4L1: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M4L1_SpaceX_Machine%20Learning%20Prediction_Part_5_TreeLog2.ipynb
  - Includes all the required tasks.
  - Added metrics per model (classification report, prediction_time and ROC-AUC score)
  - For the Decision Tree classifier, the original 'max_features: "Auto"' in given parameters was changed to 'max_features: "Log2"' as suggested in the fora.
- M4L2: https://github.com/RoSoMu/IBMCapstoneProject/blob/main/M4L2_SpaceX_Machine%20Learning%20Prediction_Part_5_TreeAuto_Threshold.ipynb
  - Includes all the required tasks.
  - Added metrics per model (classification report, prediction_time and ROC-AUC score)
  - For the Decision Tree classifier, the original max_features: "Auto".
  - Upon the Logistic Regression classifier predictions and matrix were rendered, an extra cell was added to work on adjusting the threshold to compare results.
  - Upon the SVM classifier predictions and matrix were rendered, an extra cell was added to work on adjusting the threshold to compare results.
  - Requested bar plots for accuracy were also added at the end of this notebook


Thank you!