IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

David Lepore
Aug 11, 2025

Outline

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

Executive Summary

- Data was collected directly from SpaceX REST API and by scraping the Wikipedia page using BeautifulSoup.
  - Unnecessary features and entries were removed while additional columns were created with further launch information.
- Data collected was processed/wrangled by converting landing outcomes to Classes, either 1 or 0 for good or bad launch outcomes.
- Exploratory data analysis (EDA) was performed using visualization and SQL
- Interactive visual analytics were performed using Folium and Plotly Dash
- Predictive analysis performed using classification models
  - Standardized data, split into training and test sets, used Grid Search to find the best hyperparameters for 4 models (Logistics Regression, SVM, Decision Tree, K Nearest Neighbor), evaluated using test sets to find best performing model

<u>Results</u>

- Due to our relatively small data set, all models performed similarly
  - Best model based on generalizability would be Decision Tree Classifier
  - For similar performance, but faster fit use K Nearest Neighbor
  - We correctly predicted if the Falcon 9 would land 100% of the time
  - We incorrectly predicted the Falcon 9 would NOT land 50% of the time

Introduction

- SpaceX is an American company specializing in affordable access to space
  - Developed the Falcon 9 reusable first stage
  - Advertises a Falcon 9 rocket at $62mil compared to others upwards of $165mil
- However, they do not always land, sometimes they crash, sometimes they are sacrificed
- If we can determine if the first stage will land, we can determine the cost of a launch and see if we will be competitive

Section 1

Methodology

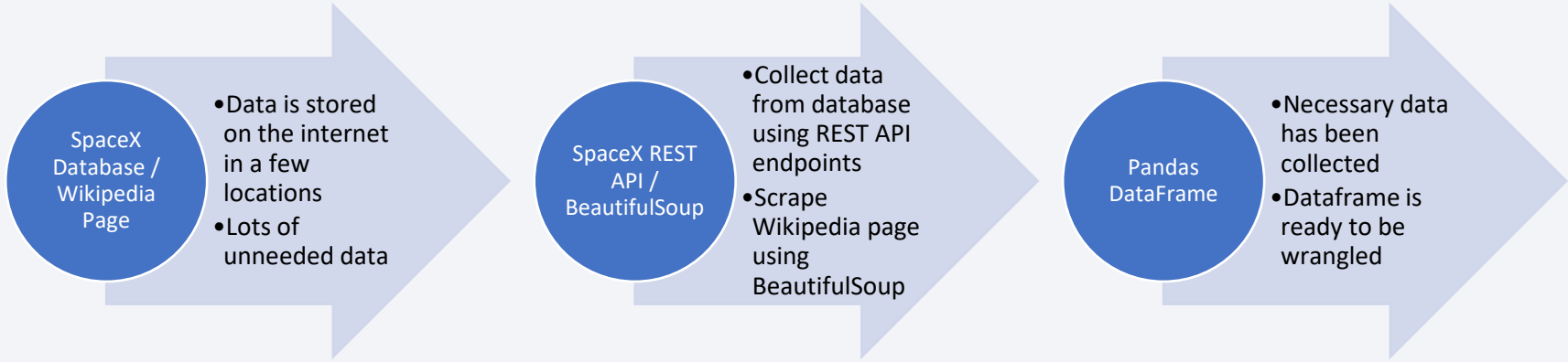
Methodology

Executive Summary

- Data collection methodology:
  - Directly from SpaceX REST API, Web Scraping the Wikipedia page using BeautifulSoup
- Perform data wrangling
  - Normalize our JSON from the REST API into a DataFrame, Convert landing outcomes to Classes, either 1 or 0 for good or bad launch outcomes
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Standardized data, split into training and test sets, used Grid Search to find the best hyperparameters for 4 models (Logistics Regression, SVM, Decision Tree, K Nearest Neighbor), evaluated using test sets to find best performing model

Data Collection

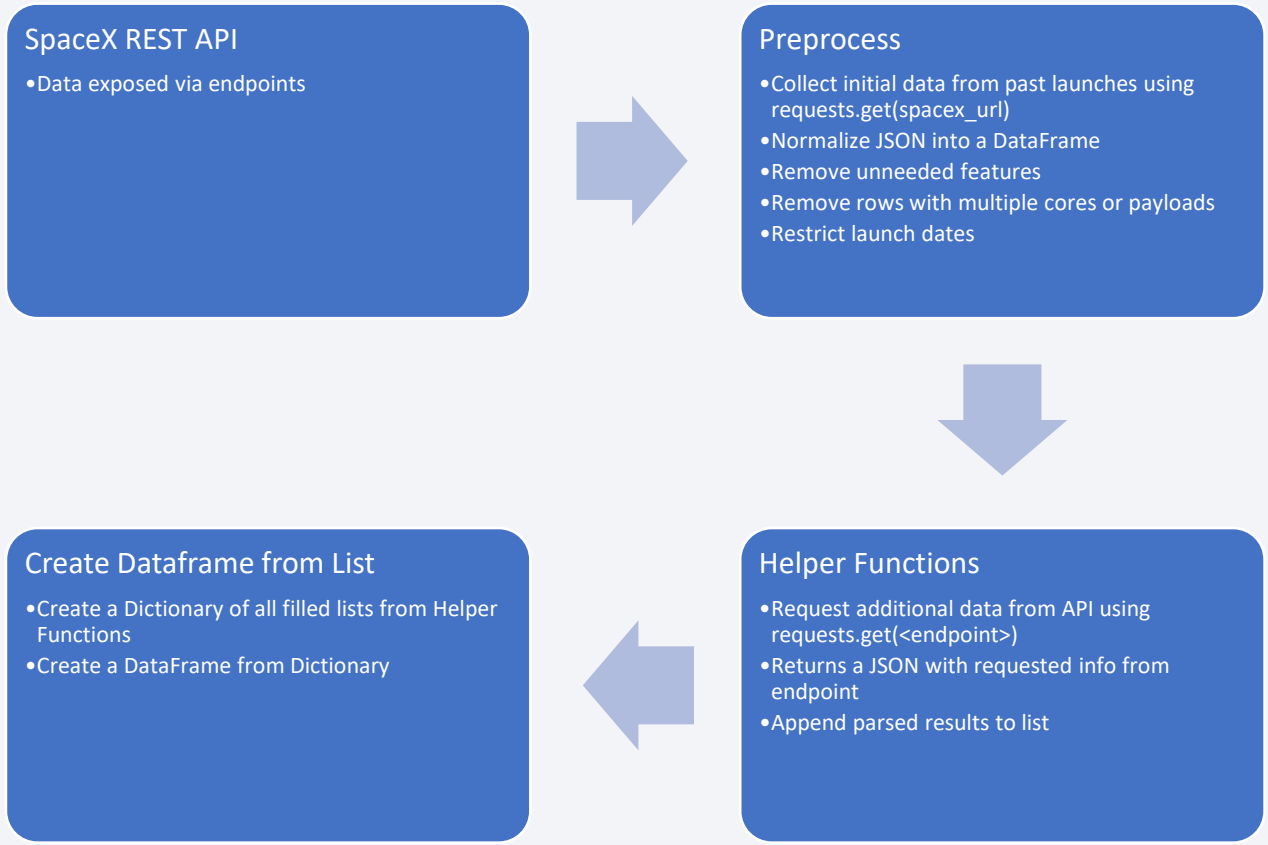
- Data was collected directly from SpaceX REST API and by scraping the Wikipedia page using BeautifulSoup.
  - From SpaceX REST API core, we pulled the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, whether the core is reused, whether legs were used, the landing pad used, the block of the core which is a number used to separate version of cores, the number of times this specific core has been reused, and the serial of the core.
  - Using different SpaceX REST API endpoints, we gathered BoosterVersion, LaunchSite, and PayloadMass
  - Similar information was scraped from the Wikipedia page using BeautifulSoup: Flight No., Launch Site, Payload, Payload Mass, Orbit, Customer, Launch Outcome, Version Booster, Booster Landing, Date, Time
- Unnecessary features and entries were removed while additional columns were created with further launch information.



Data Collection – SpaceX API

- SpaceX REST API will give us data about launches via endpoints
- Created functions to query each endpoint using requests.get
- Appended relevant info parsed from returned JSON to a list
- Lists will be used to form a launch_dict dictionary
- Dictionary will be used to create our DataFrame we will wrangle

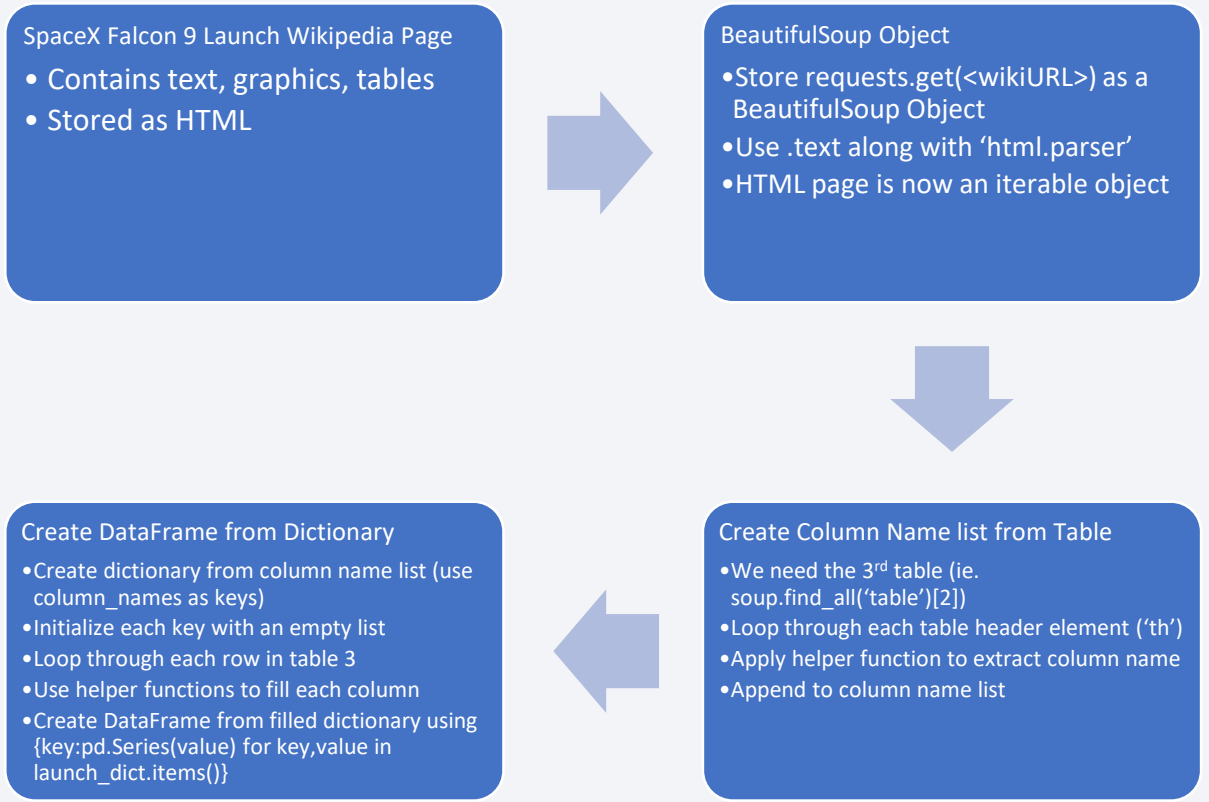
- Github: https://github.com/Davidlepore/CourseraTestingRepoPublic/blob/main/Capstone/jupyter-labs-spacex-data-collection-api-lab1.ipynb



Data Collection - Scraping

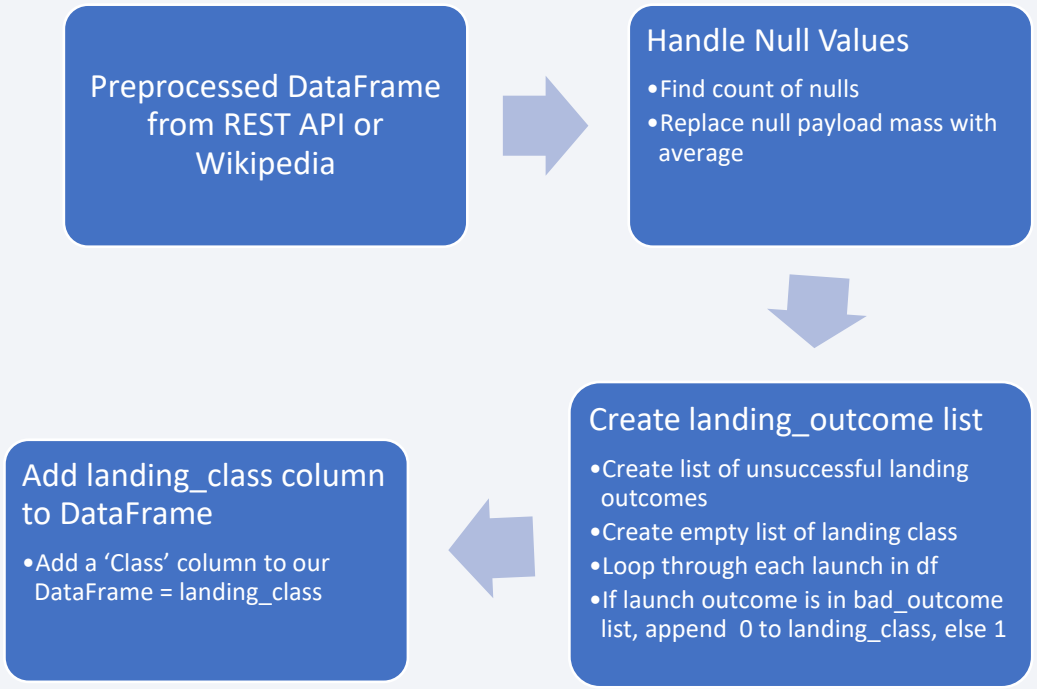
- SpaceX Falcon 9 launches are also stored on Wikipedia in HTML format
- Defined helper functions to scrape the web page (ie. return date and time, booster version, landing status, payload mass, and column headers)
- Created a BeautifulSoup object from the HTML page
- Created a dictionary of lists to fill (this will become our final DataFrame)
- Parsed the tables found in the BeautifulSoup object
- Necessary info was put in our dictionary using helper functions
- Converted dictionary into DataFrame to wrangle

- GitHub: https://github.com/Davidlepore/CourseraTestingRepoPublic/blob/main/Capstone/jupyter-labs-webscraping-lab1-part2.ipynb



Data Wrangling

- Data was processed in a number of ways
  - Dealt with null values
    - LandingPad null values will be kept as they represent when landing pads were used
    - PayloadMass null values were replaced with the average payload mass
  - Added landing_class column based on the 8 possible launch outcomes
    - 0 if unsuccessful landing, 1 if successful landing
  - 5 possible unsuccessful landings
    - None None
    - False ASDS
    - False Ocean
    - None ASDS
    - False RTLS
  - 3 possible successful landings
    - True ASDS
    - True RTLS
    - True Ocean



- GitHub: https://github.com/Davidlepore/CourseraTestingRepoPublic/blob/main/Capstone/labs-jupyter-spacex-Data%20wrangling-lab2.ipynb

EDA with Data Visualization

- Created 7 different charts, 5 scatter with Class as the color, 1 bar, and 1 line to show the trend
- FlightNumber was plotted against PayloadMass, LaunchSite, and Orbit in scatters
  - Used to determine the distribution of successful launch parameters (were later launches more successful)
- PayloadMass was plotted against LaunchSite and Orbit in a scatter
  - Used to determine if payloads over a certain mass were more likely to land successfully
- Orbit was plotted against Success Rate in a bar chart
  - Used to determine which sites were ideal to launch from based on past successes
- Year was plotted against Success Rate in a line chart to determine the trend

- GitHub: https://github.com/Davidlepore/CourseraTestingRepoPublic/blob/main/Capstone/jupyter-labs-eda-dataviz-lab3.ipynb

EDA with SQL

- Determined the names of the 4 unique launch sites
- Explored a handful of entries from sites beginning with ‘CCA’
- Found the total payload mass carried by boosters launched by NASA (CRS) to be 45596 Kg
- Found the average payload mass carried by booster version F9 v1.1 to be 2534.667 Kg
- The first successful landing outcome in ground pad was achieved 2015-12-22
- Determined the 4 names of boosters that successfully landed in Drone ship with a payload between 4000 and 6000 Kg
- Found there to be a total of 100 successful mission outcomes, and 1 failure
- Found all 12 booster versions that carried the maximum payload mass
- Found Month, Booster Version, Landing Outcome, and Launch Site for launches in 2015 with a Failure (drone ship) landing outcome, 2 records found in month 01 and 04
- Ranked the occurrence of landing outcomes between 2010-06-04 and 2017-03-20 in descending order, found No Attempt, Success (drone ship), and Failure (drone ship) to be the leaders
- GitHub: https://github.com/Davidlepore/CourseraTestingRepoPublic/blob/main/Capstone/jupyter-labs-eda-sql-coursera_sqllite-lab3-SQLPractice.ipynb

Build an Interactive Map with Folium

- Created circles with text markers to indicate the 4 unique launch sites and their names
  - Allows for visual identification of each site, aids those unfamiliar with US geography
- Added MarkerClusters to each launch site to indicate successful and failed launches
  - Allows for visual determination of success rate and launch count, which sites had more successful launches than others
- Added measuring lines/marks to determine distance from CCAFS SLC-40 to the nearest coast, town, highway, and railroad
  - Allows for visual determination of construction requirements (how far must they be from cities and coastline, do they need a private railroad or highway for transport, etc.)
  - Additional lines/markers can be added for the remaining sites to see if there are baseline requirements for a site to be constructed

- GitHub: https://github.com/Davidlepore/CourseraTestingRepoPublic/blob/main/Capstone/lab_jupyter_launch_site_location-lab4.ipynb
- To view maps: https://nbviewer.org/github/Davidlepore/CourseraTestingRepoPublic/blob/main/Capstone/lab_jupyter_launch_site_location-lab4.ipynb

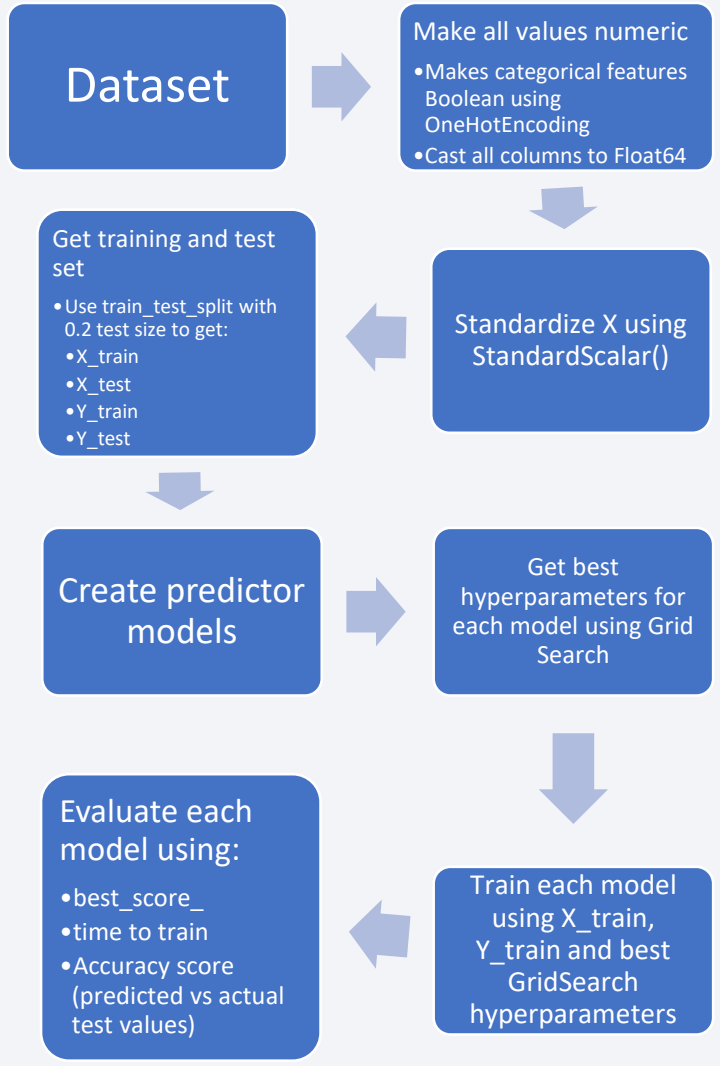
Build a Dashboard with Plotly Dash

- Dashboard includes 2 Plots, Site Selection Dropdown, and Payload Range Slider
- 1st Plot: Pie chart showing successful launches per site
  - All Sites selected by default, shows portion of total successful launches by site
  - A specific launch site can be selected via the Site Selection Dropdown to show site's successful vs failed launches only
- 2nd Plot: Scatter chart showing Payload Mass vs Class (success/fail)
  - Launches are colored based on their Booster Version
  - Chart will filter to show only launches from selected site based on Site Selection Dropdown
  - Chart can be further filtered by payload range using the Payload Range Slider
- Charts allow us to further explore each launch site's performance
  - Can easily view specific launch information in an easy understand format
  - Can find what payload mass and booster version are ideal for each site
  - Can visually predict the likelihood of a successful launch given the site, booster version, and payload mass

- GitHub Notebook (download and run locally/cloud to show dashboard): https://github.com/Davidlepore/CourseraTestingRepoPublic/blob/main/Capstone/jupyter-labs-spacex-dash-app-lab4-part2.ipynb
- Python script: https://github.com/Davidlepore/CourseraTestingRepoPublic/blob/main/Capstone/spacex-dash-app.py

Predictive Analysis (Classification)

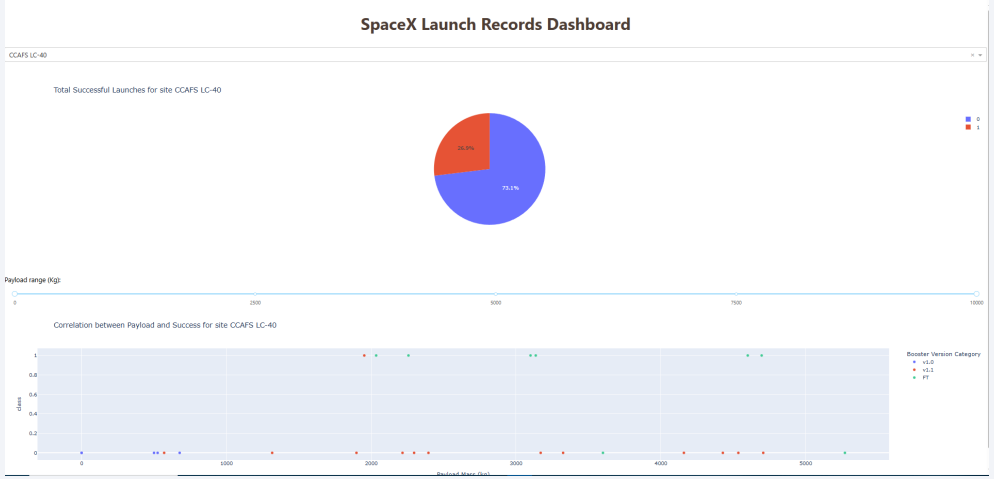
- To prepare the data set:
  - Performed One Hot Encoding on the categorical features, cast all columns to float64
  - Created X (input) dataframe from feature columns
  - Created Y (output/class) pandas series from target column
  - Standardized X using StandardScaler()
  - Created training and test sets using train_test_split with 20% as our test size (0.2)
- Created 4 predictor models: Logistic Regression, Support Vector Machine, Decision Tree, K Nearest Neighbor
- Optimized models using Grid Search to find the best hyperparameters for each model
  - Trained each model using the best hyperparameters from Grid Search and the same training set
- Best model was select based on generalizability (using best_score_ of the model), accuracy on unseen data (score using test sets), and run time
- The confusion matrix for each model was also plotted for visual comparison
- GitHub: https://github.com/Davidlepore/CourseraTestingRepoPublic/blob/main/Capstone/SpaceX_Machine%20Learning%20Prediction_Lab5.ipynb

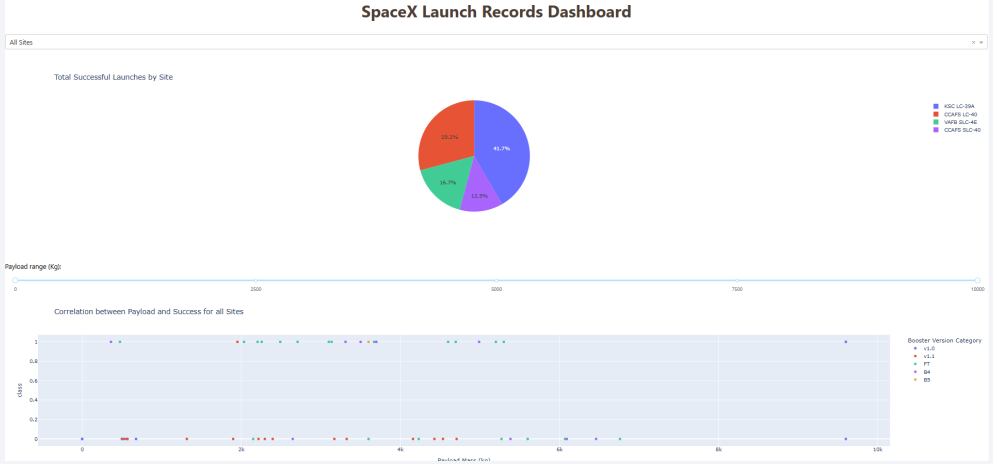


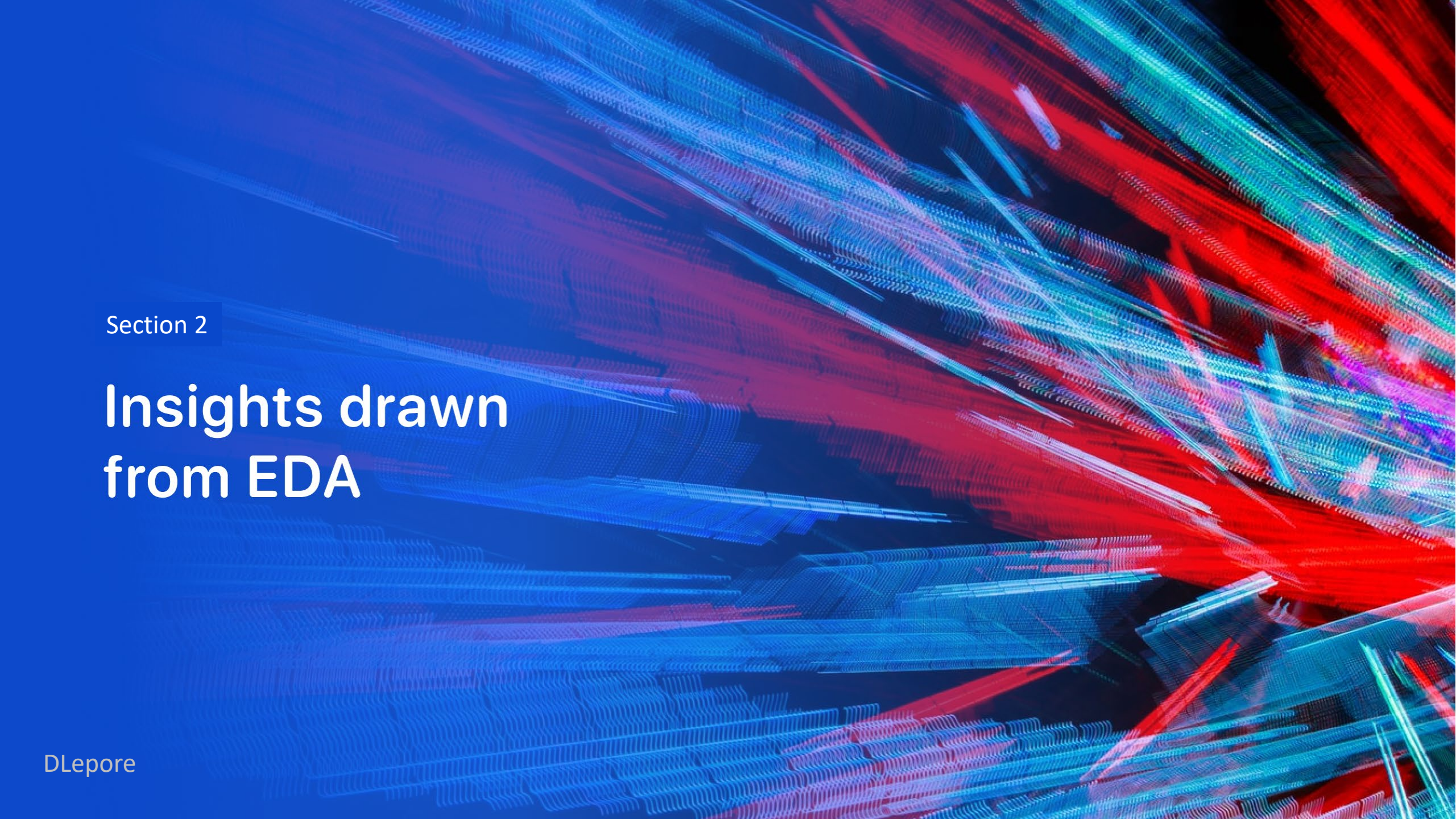
Results

- Exploratory data analysis results
  - As flight number increases, the chances of a successful launch increase regardless of payload mass, launch site, or orbit
  - Payload Mass and Orbit could be used to exclusively determine if a launch will be successful or not (numerous orbits had 100% success rate)
    - Certain orbits also had 100% success rate over a certain payload mass
  - As the years progress, we see a generally upward trend in successful launches
- Predictive analysis results
  - All of our models were able to correctly predict the outcome of the launch with 83% accuracy
  - The shortcoming for each model was in correctly predicting if it would NOT land (incorrectly predicted the launch would land 3 times)
  - Decision Tree Classifier gave the highest best_score_ which may prove it is more generalized, but it took the longest to train
  - K Nearest Neighbor had the second highest best_score_ while being the second fastest to train
  - Either Decision Tree or KNN would be a good choice

- Interactive analytics demo in screenshots







Section 2

Insights drawn from EDA

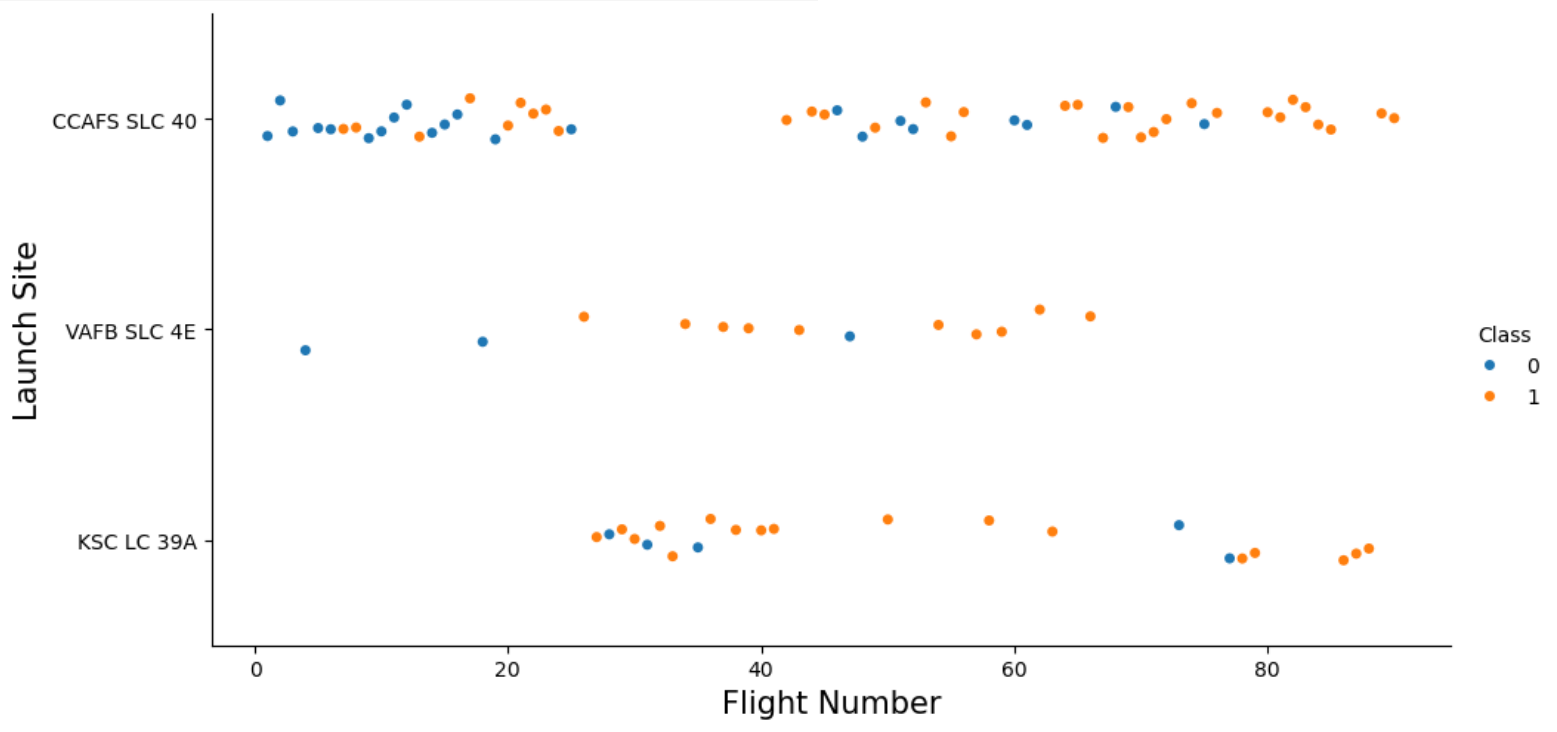
DLepore

Flight Number vs. Launch Site

- Scatter Plot of 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 value
sns.catplot(x="FlightNumber", y="LaunchSite", data=df,hue='Class', aspect=2)
plt.xlabel("Flight Number",fontsize=15)
plt.ylabel("Launch Site",fontsize=15)
plt.show()
```

- We can see that as flight number increases, success rate increases
- However, this is not a clear enough predictor alone

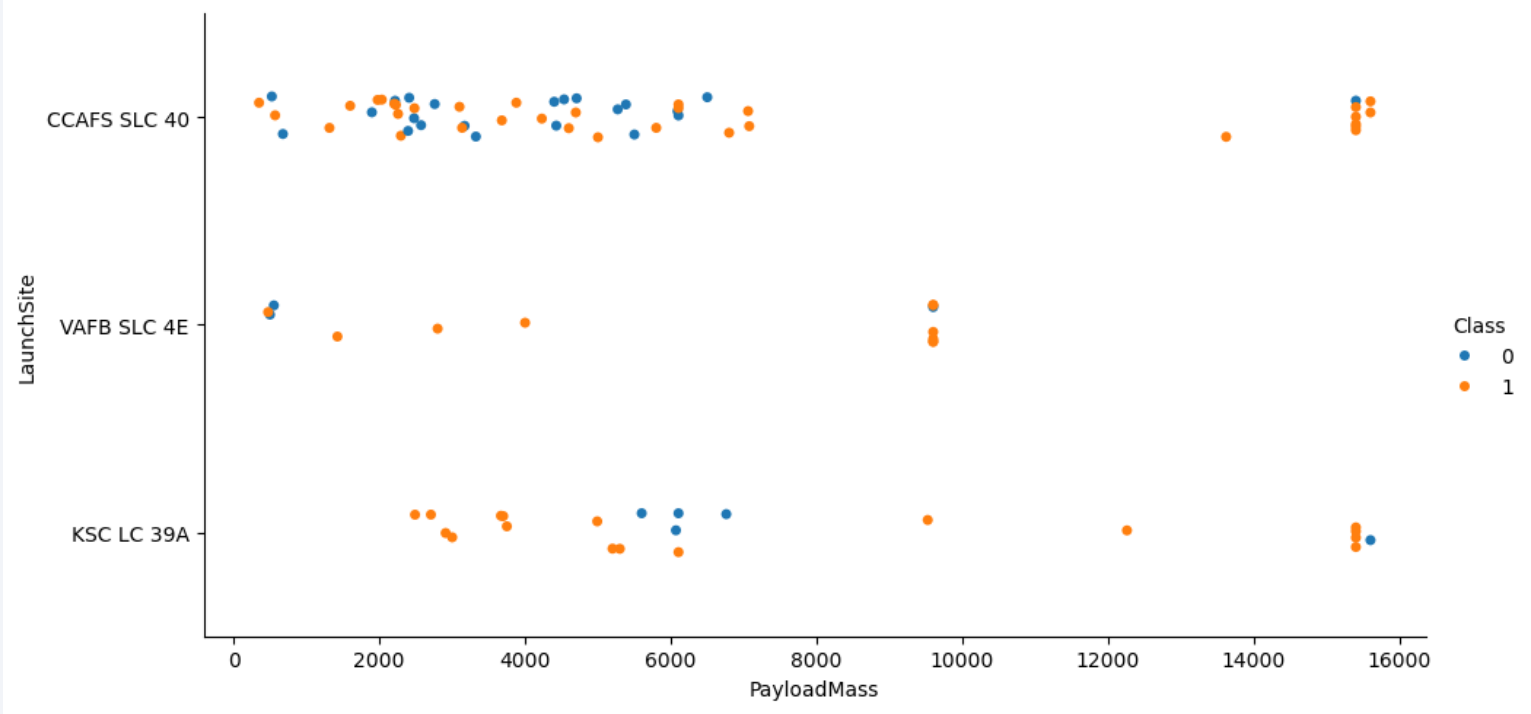


Payload vs. Launch Site

- Scatter Plot of Payload vs. Launch Site

```
# 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 the class value
sns.catplot(x="PayloadMass", y="LaunchSite", data=df,hue='Class', aspect=2)
```

- We can see that there are some launch sites that do not launch heavier payloads
- We can also see that for other launch sites, heavier payloads typically mean a higher success rate

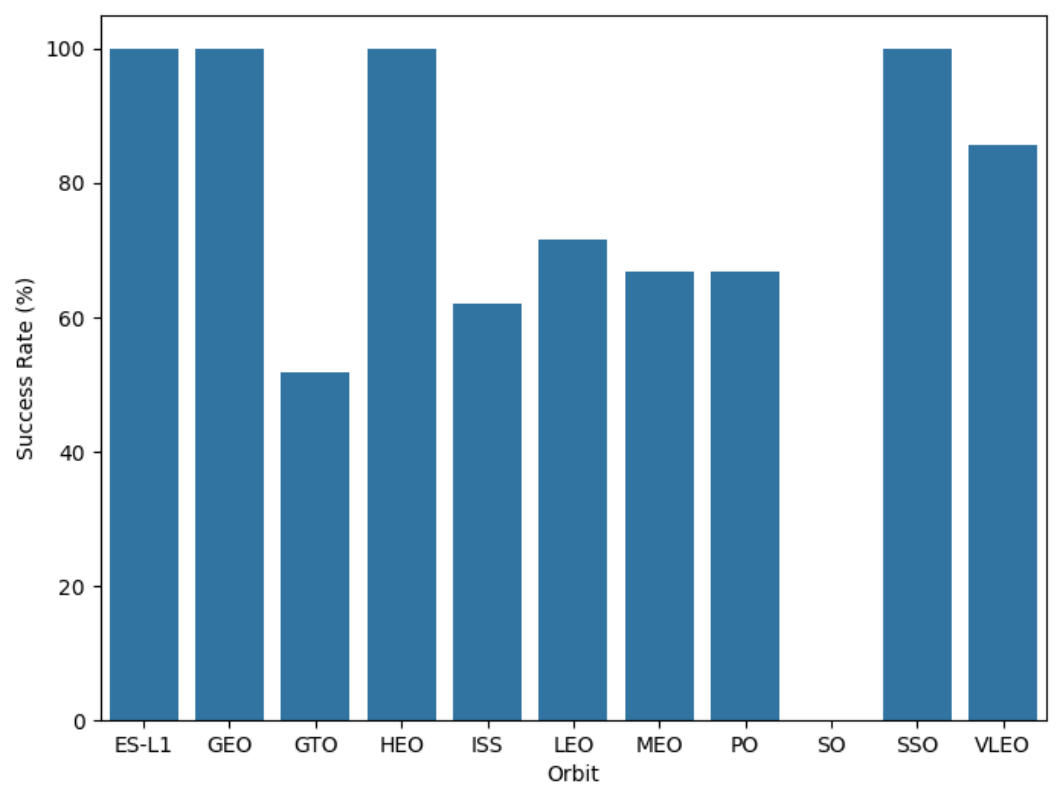


Success Rate vs. Orbit Type

- Bar Chart of success rate of each orbit type

```
success_df = df.groupby('Orbit')['Class'].mean().mul(100).reset_index()
plt.figure(figsize=(8,6))
sns.barplot(x='Orbit',y='Class',data=success_df)
plt.ylabel('Success Rate (%)')
plt.show()
```

- We can see there are numerous Orbits that have a 100% success rate
- There is also one that has a 0% success rate
- The rest hover around 50-90% success rate

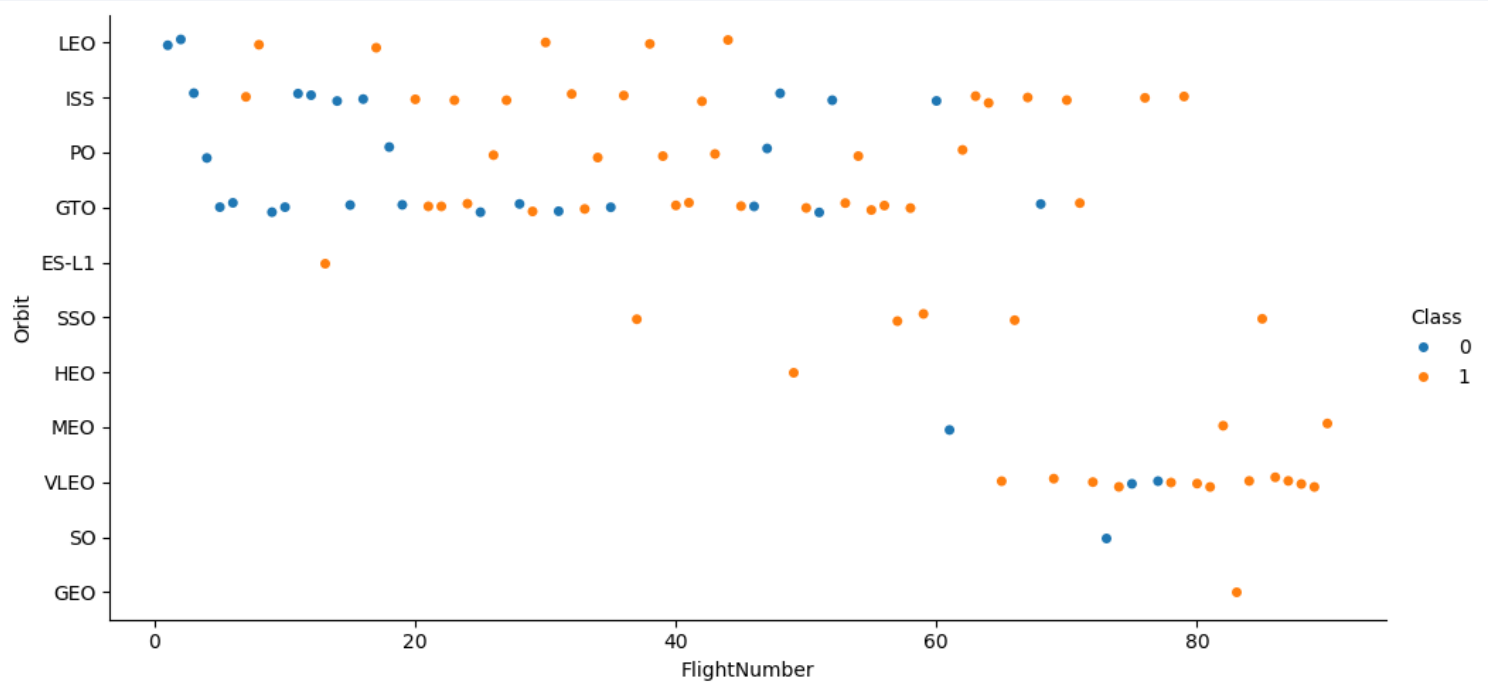


Flight Number vs. Orbit Type

- Scatter Plot of Flight number vs. Orbit type

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(x="FlightNumber", y="Orbit", data=df,hue='Class', aspect=2)
```

- We can see that for certain orbits, success rate increases to 100% as flight number increases
- However, it is harder to determine the cutoff for some orbits as success and failure are more common throughout
- Some orbits have only had a few launches
- We can also see how some Orbits are used more or less as flight number increases

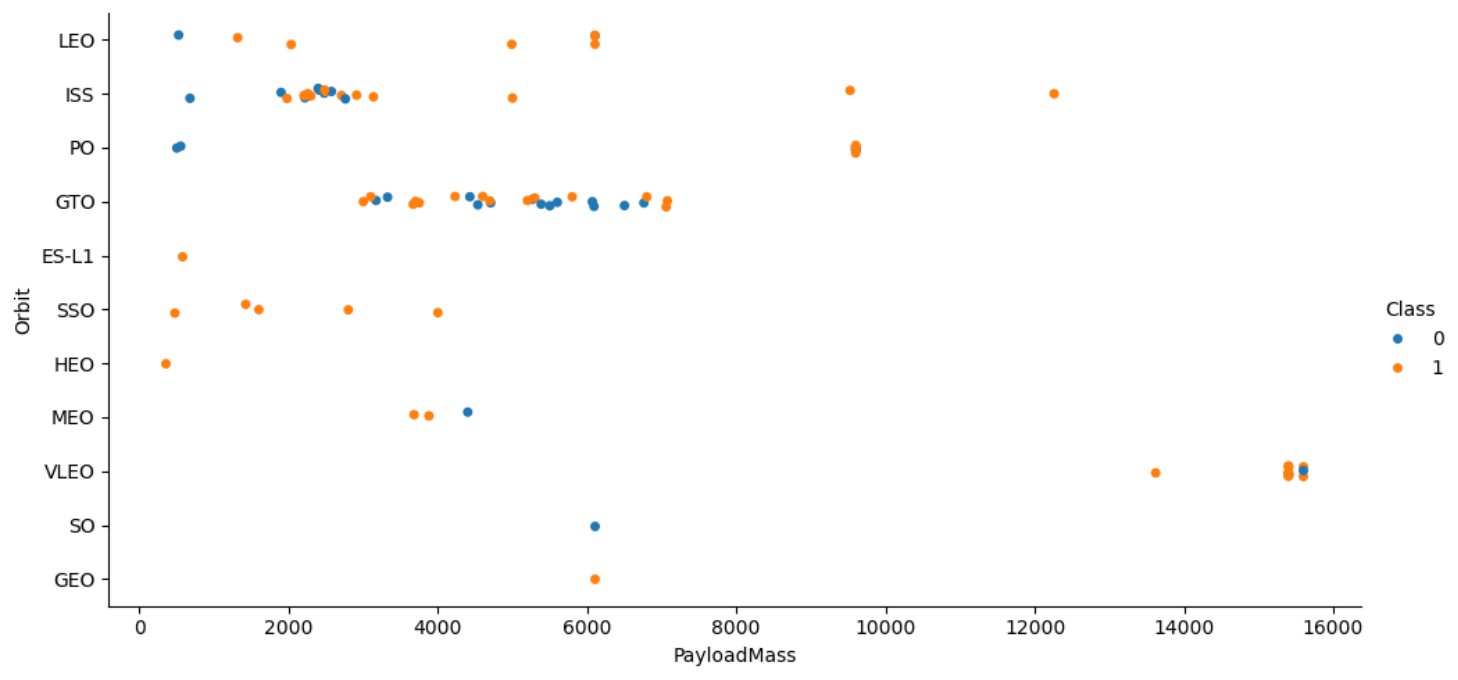


Payload vs. Orbit Type

- Scatter Plot of payload vs. orbit type

```
# 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 value
sns.catplot(x="PayloadMass", y="Orbit", data=df,hue='Class', aspect=2)
```

- We can clearly see for certain orbits, heavier payloads yield successful launches
- However, for orbits GTO, it is harder to distinguish
- For VLEO and MEO, we can see that lighter payloads yield successful launches

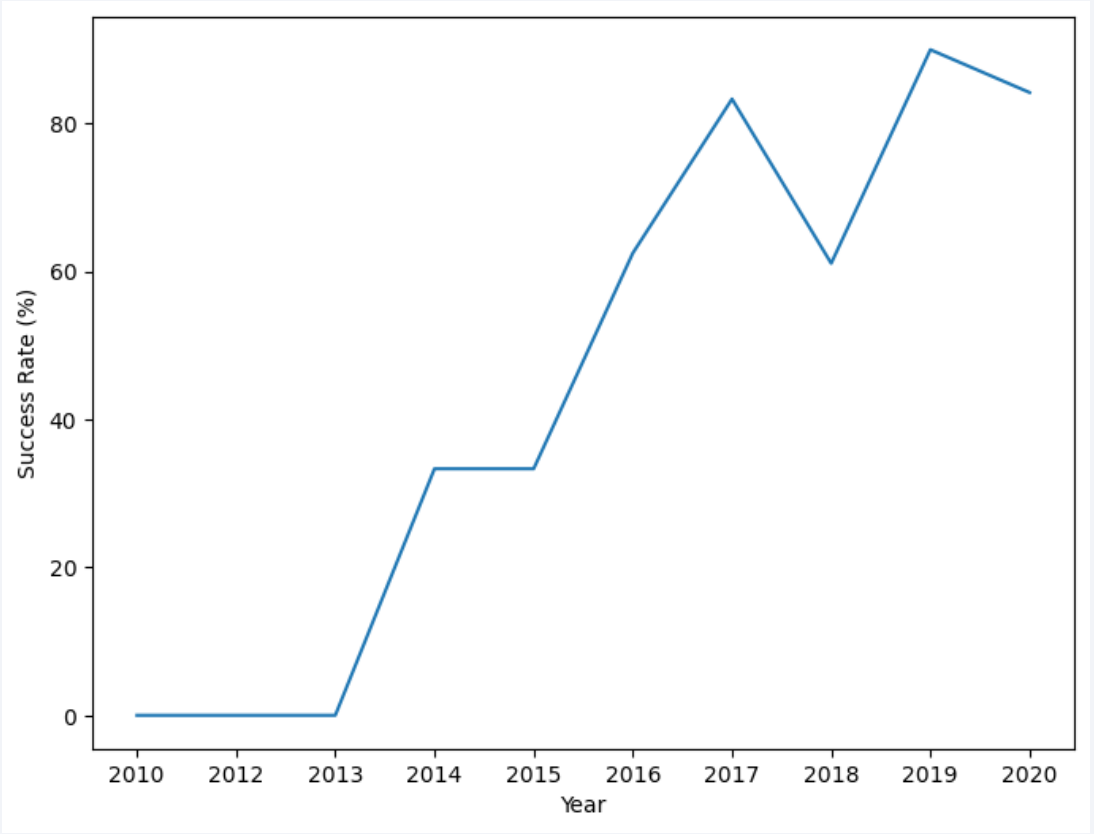


Launch Success Yearly Trend

- Line chart of yearly success rate

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
success_df = df.groupby('Date')['Class'].mean().mul(100).reset_index()
plt.figure(figsize=(8,6))
sns.lineplot(x='Date',y='Class',data=success_df)
plt.xlabel('Year')
plt.ylabel('Success Rate (%)')
plt.show()
```

- We can clearly see that as the years progress, the success rate generally increases
- Exceptions in 2017 and 2019



All Launch Site Names

- Find the names of the unique launch sites

```
%sql SELECT DISTINCT("Launch_Site") FROM SPACEXTABLE
```

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

- We can see there are 4 unique launch sites (database must use different data that API)

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

```
%sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" like 'CCA%' LIMIT 5
```

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

- We see numerous launches to LEO from SpaceX and NASA from CCAFS LC-40 launch site

Total Payload Mass

- Calculate the total payload carried by boosters from NASA (CRS)

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass (Kg)" FROM SPACEXTABLE WHERE Customer = "NASA (CRS)"
```

Total Payload Mass (Kg)
45596

- Total Payload Mass carried by boosters from NASA is 45596 Kg

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Avg Payload Mass (Kg)" FROM SPACEXTABLE WHERE "Booster_Version" like 'F9 v1.1%'
```

Avg Payload Mass (Kg)
2534.6666666666665

- Average payload mass carried by booster version F9 v1.1 is 2534.667 Kg

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)'
```

MIN(Date)
2015-12-22

- First successful landing outcome on ground pad was on 2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
%sql SELECT DISTINCT("Booster_Version") FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000
```

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

- We can see there are 4 boosters that have successfully landed on drone ship with payload mass between 4000-6000 Kg

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT(*) AS "Outcome Count" FROM SPACEXTABLE GROUP BY "Mission_Outcome"
```

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

- We can see that our data set contains mostly successful missions, with only 1 failure
- Mission outcome is not the same as landing/launch outcome

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

```
%sql SELECT DISTINCT("Booster_Version") FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE) ORDER BY "Booster_Version" ASC
```

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

- We can see there are 12 boosters that have carried the maximum payload mass

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
%sql SELECT substr(DATE, 6, 2) AS "Month", "Landing_Outcome", "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE substr(Date, 0, 5) = '2015' AND "Landing_Outcome" = 'Failure (drone ship)'
```

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

- We can see there were only 2 records found that matched our criteria

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

- 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(*) AS "Outcome Count" FROM SPACEXTABLE WHERE Date > '2010-06-04' AND Date < '2017-03-20' GROUP BY "Landing_Outcome" ORDER BY "Outcome Count" DESC
```

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

- We can see the most common outcome between the dates is No attempt [at landing]
- Second most common landing outcome is on a drop ship whether failure or success

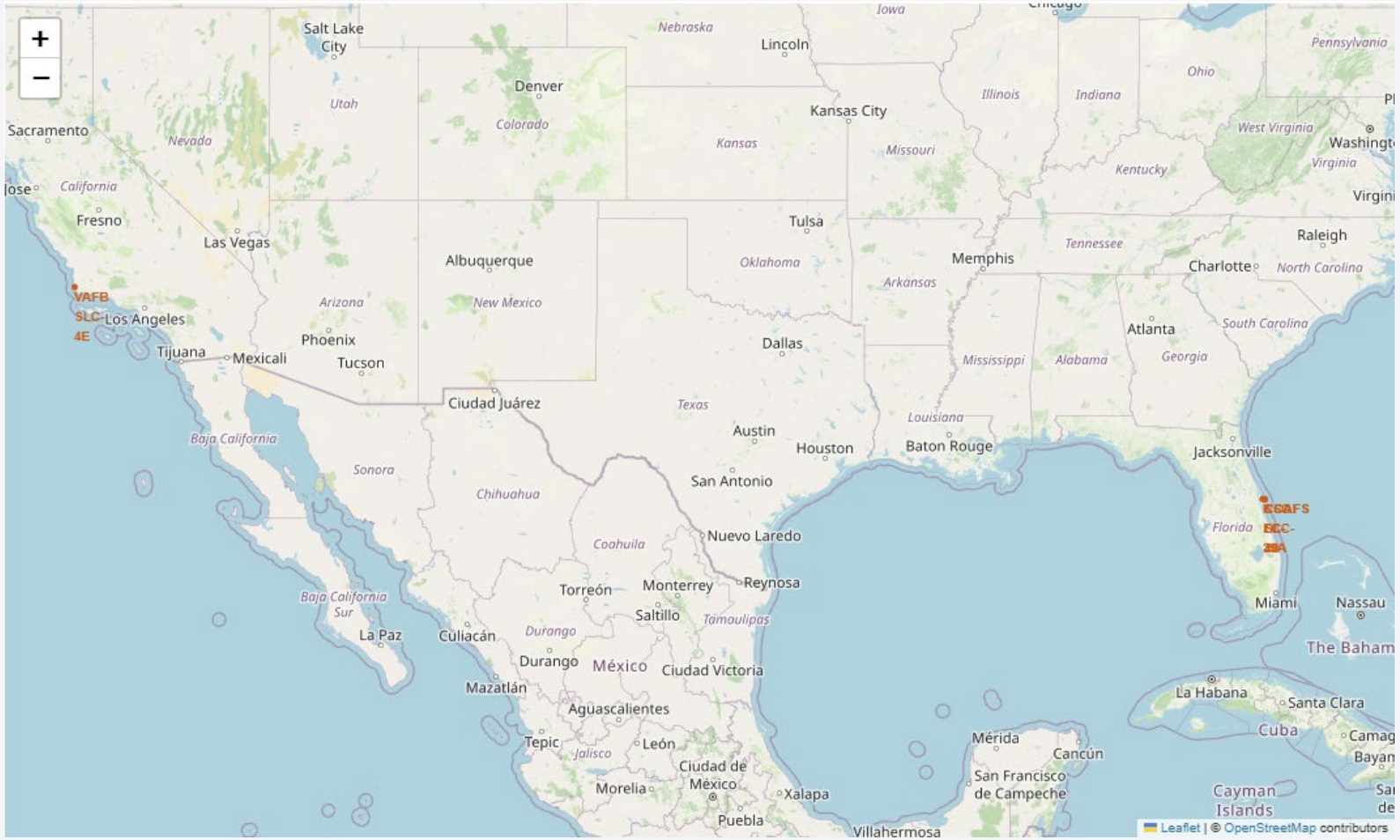
Section 3

Launch Sites Proximities Analysis

DLepore

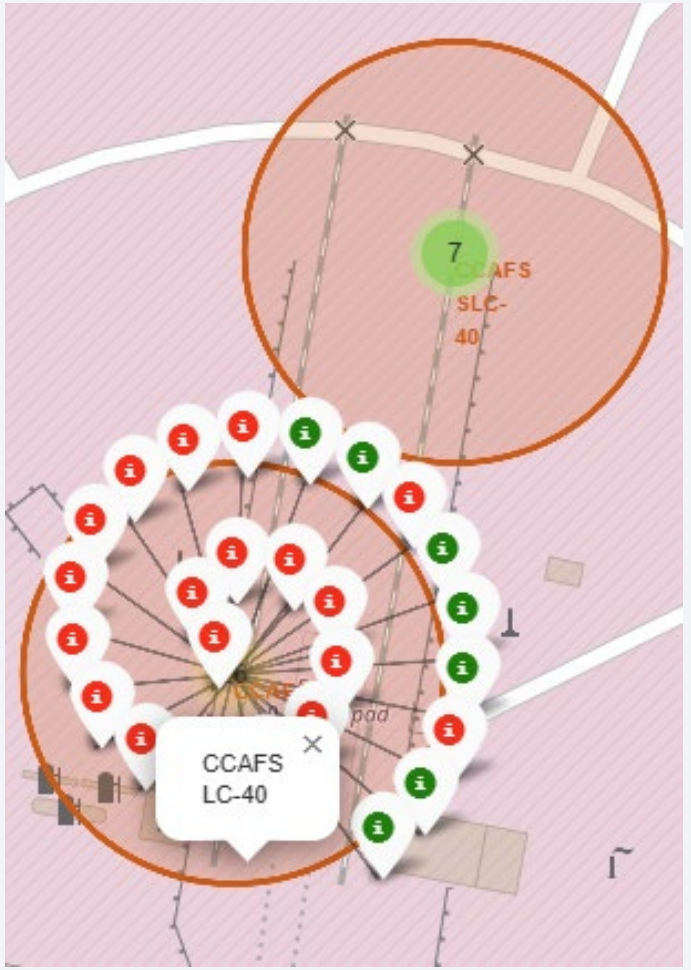
Folium Map – Launch Site Markers

- We have added circle markers at the latitude and longitude of each launch site
- We have also added the launch site names
- As you zoom, they will spread out and not overlap
- We can see that all sites are located extremely close to the coast
- They are also in the southern area of the US, closer to the equator



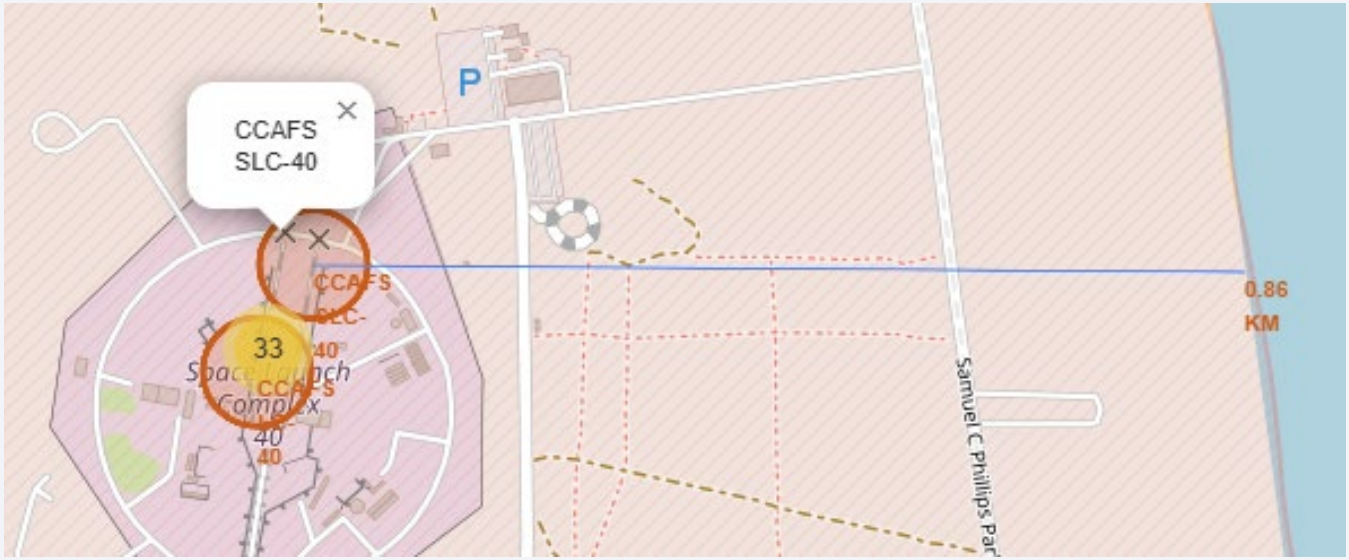
Folium Map – Success/Fail Launch Markers

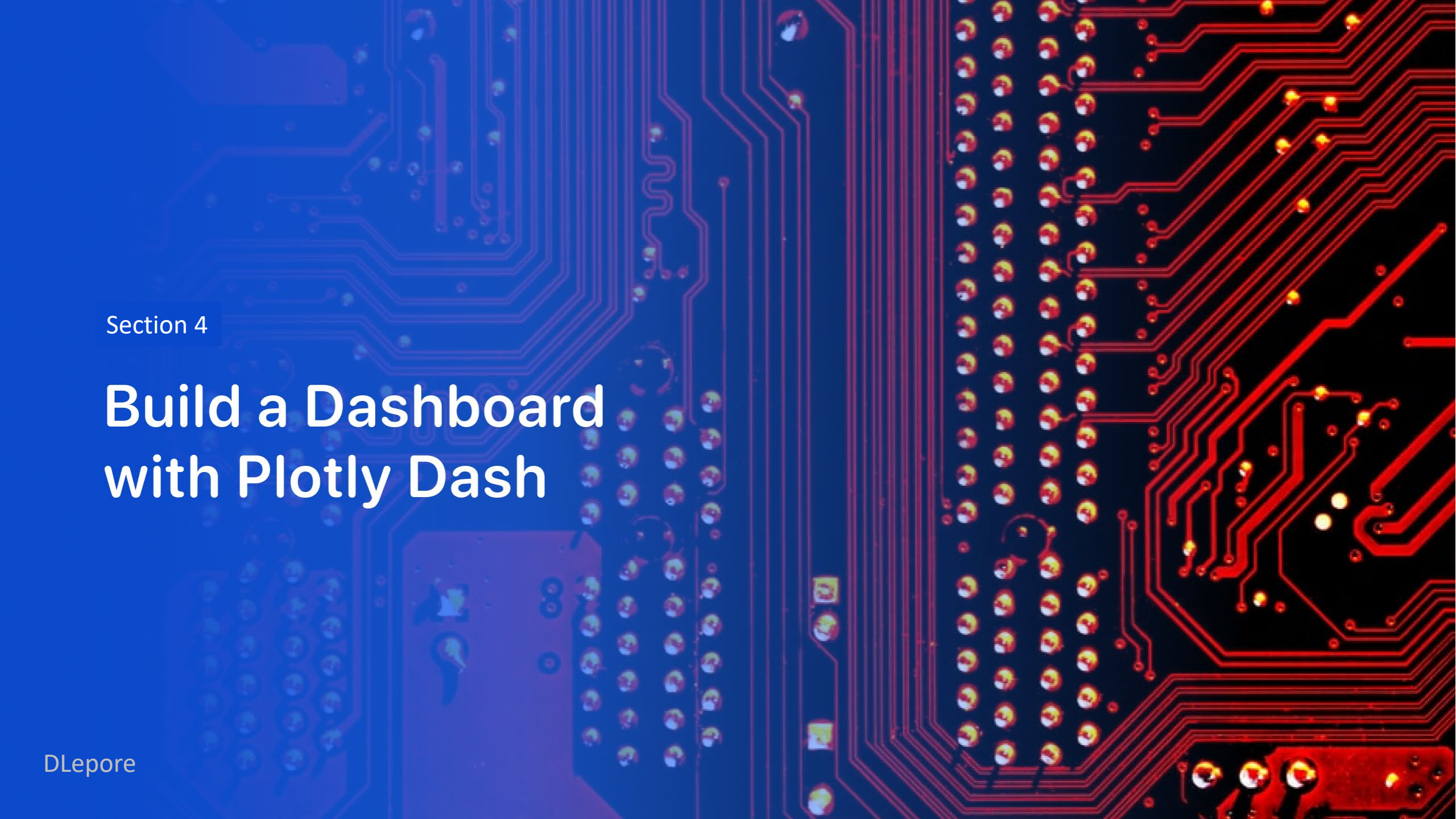
- Added marker clusters at each launch site to indicate successful and failed launches
- Zooming in and clicking on the launch site will reveal all the launches with coloring (red = fail, green = success)
- We can easily identify which launch sites have a high success rate by looking at the marker cluster colors



Folium Map – Coast Proximity

- We can calculate the distance from between any two coordinates
- We can then create a marker with the distance noted
- Finally we can create a polyline to show the distance on our map
- We can see while zoomed out that many launch sites are required to be extremely close to the coast
- CCAFS SCL-40 for example is not even 1KM away from the water



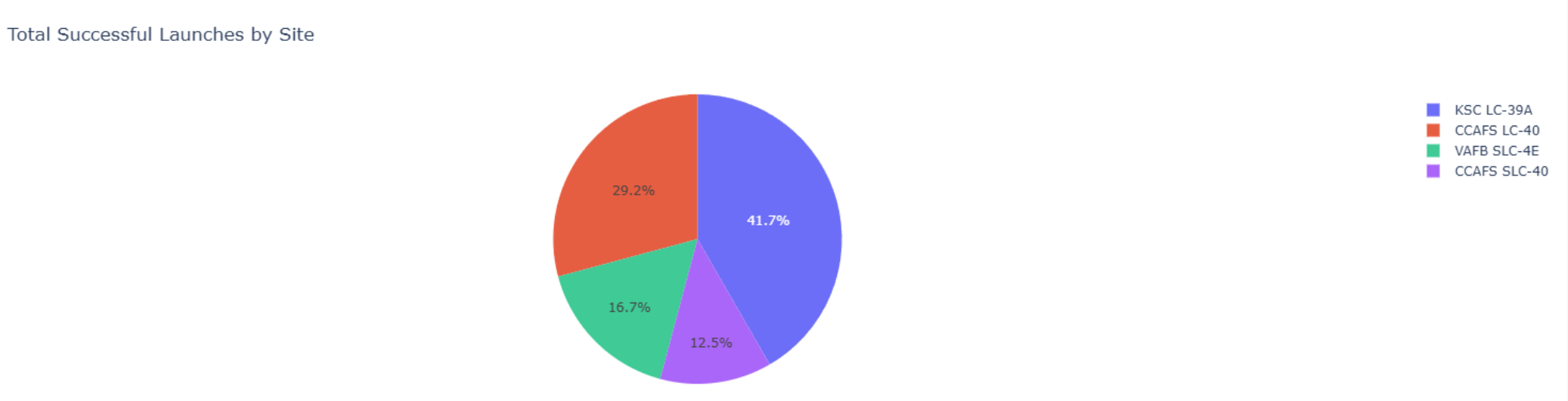


Section 4

Build a Dashboard with Plotly Dash

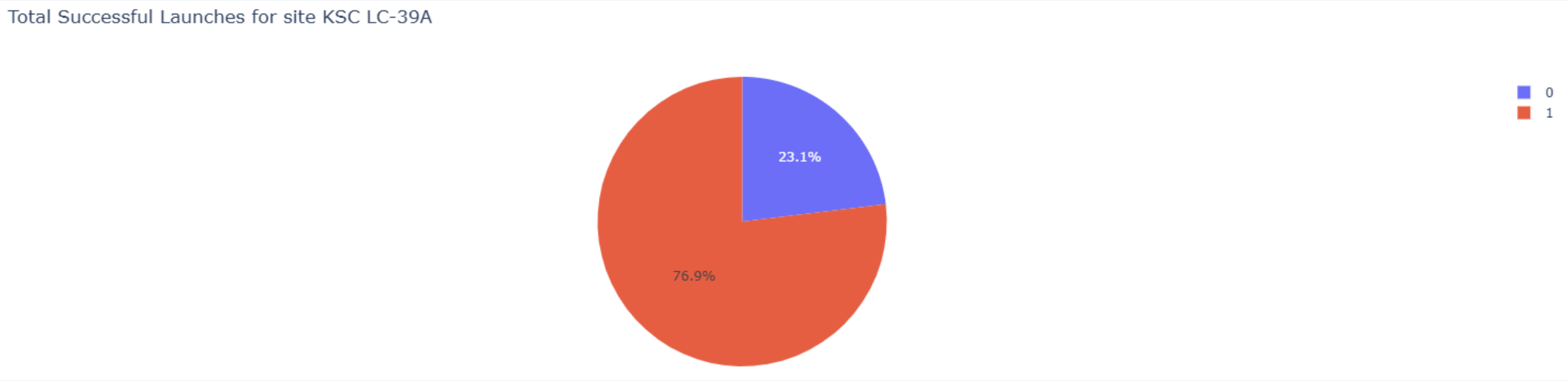
DLepore

Dash – Launch Site Success Count Pie Chart



- We can see that the most successful launch site it KSC LC-29A by far
  - 41.7% of successful launches happened here

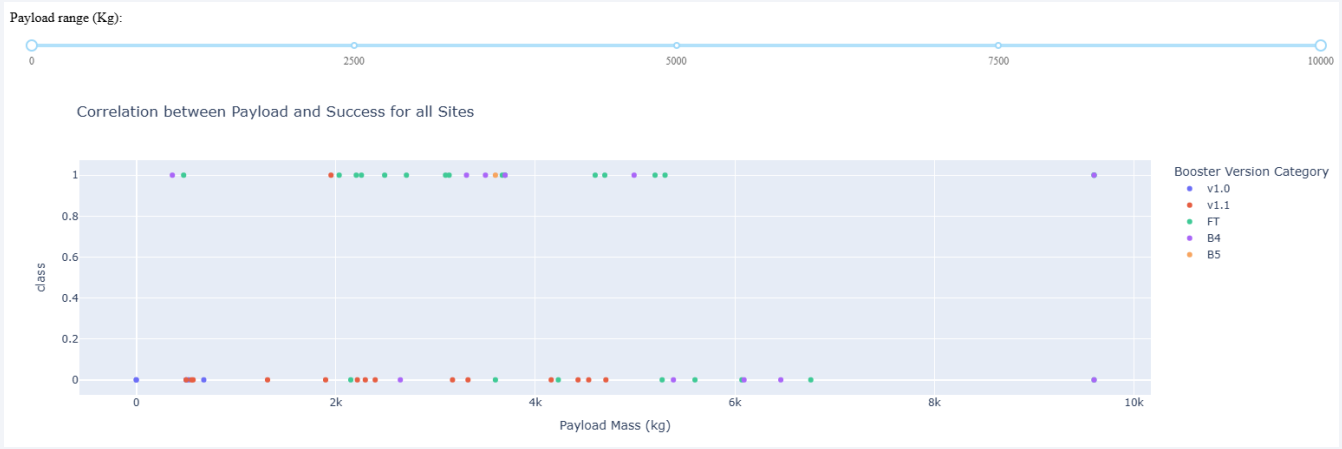
Dash – Most Successful Launch Site Pie Chart



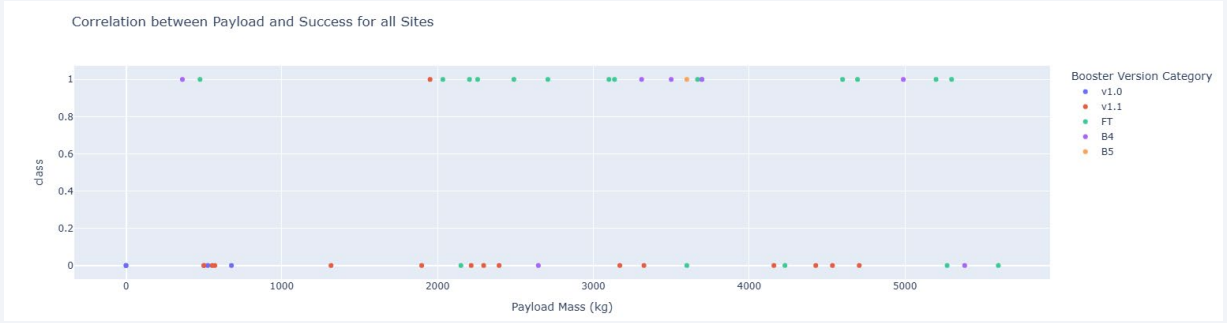
- Note: Red = 1, Success, Blue = 0, Fail
- We can see that roughly 77% of launches from KSC LC-39A were successful

Dash – Payload Mass vs Outcome Scatter Chart

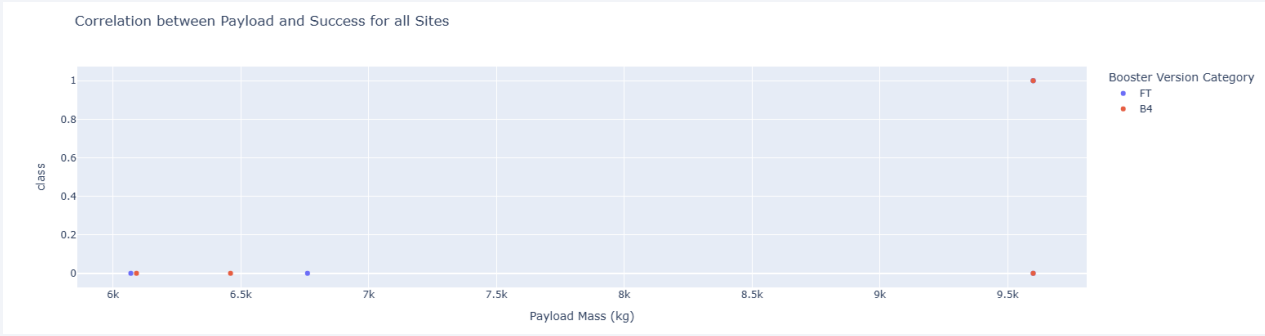
- Full Range – We can see that booster version FT appears to have the highest success rate



- If we adjust the range to 0-6000 on the slider, we can see range 0-5500 Kg gives the most successful launches



- Conversely, if we adjust range to 6000-10000 Kg, we see an overwhelming amount of failed launches



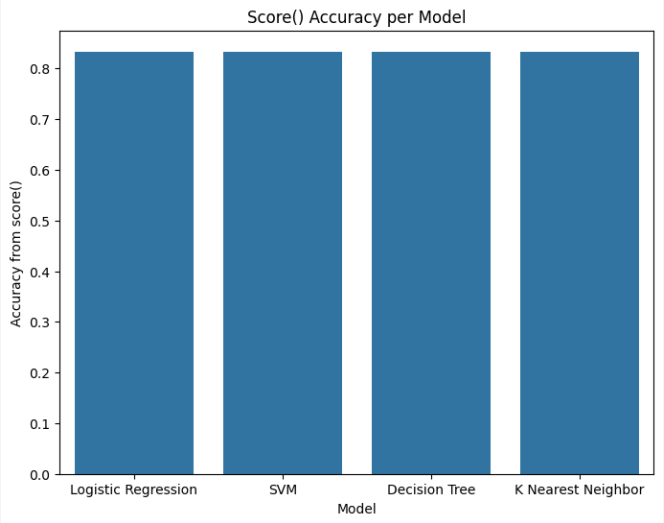
Section 5

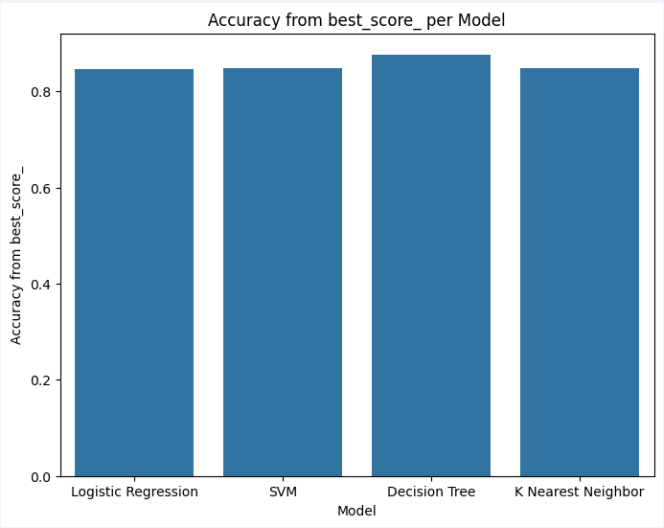
Predictive Analysis (Classification)

DLepore

Classification Accuracy

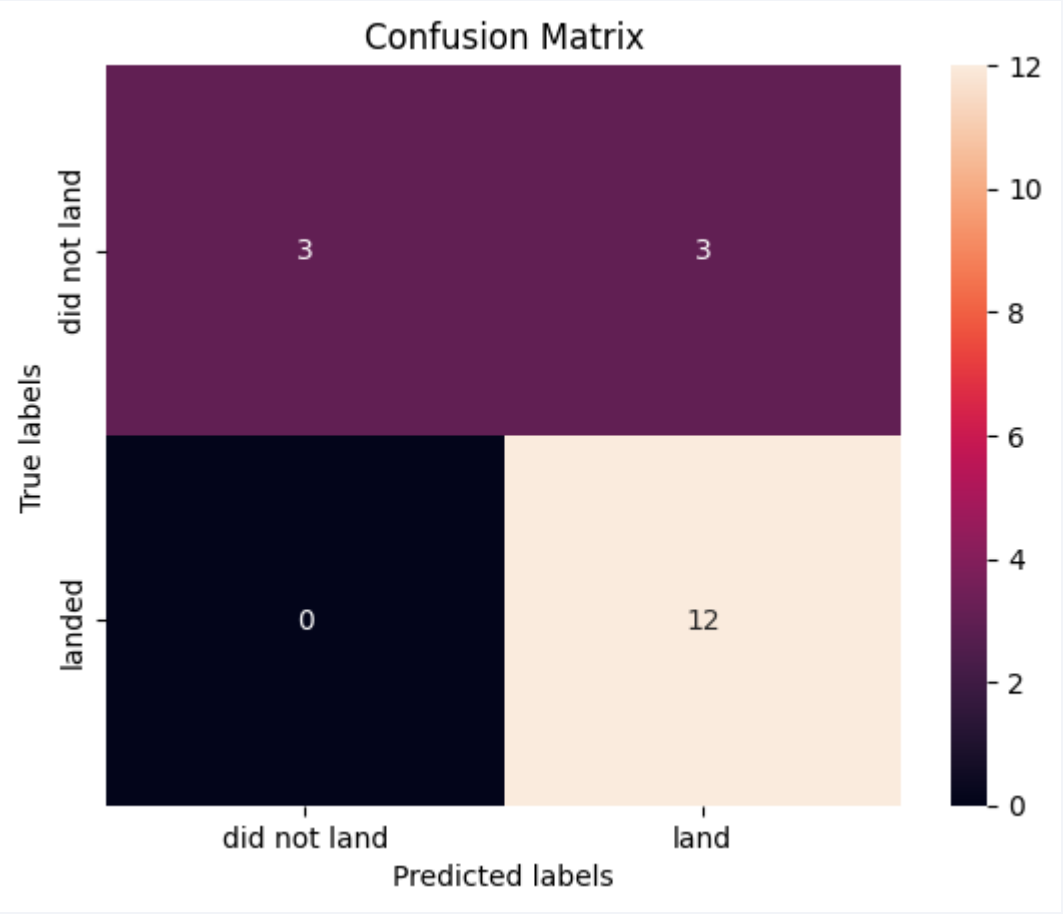
- Due to our small data set, the models perform almost identically (all have 83.33% according to score() function)
- However, we can differentiate them by their best_score_ (generalizability) and noticed training time
  - Decision Tree has highest best_score_
  - While K Nearest Neighbor has 2nd highest but trained faster
- Decision Tree would be the best to use if training time/cost is not a concern





Confusion Matrix

- We can see that from 18 test sets, we correctly predicted 15 outcomes
- We incorrectly predicted 3 launches would land when they in fact didn’t (false positive)
- We did not have any false negatives!



Conclusions

- Using Decision Tree Classifier, we can correctly predict the outcome 83% of the time
- To further improve our model:
  - We can look more closely at our features and see if removing lowly correlated features improves our accuracy
  - We can retrain our model when we have more launch data
- For the highest rate of success, the launch should include the following parameters
  - KSC LC-39A Launch Site
  - Payload mass of less than 5500 Kg or greater than 7000 Kg
  - Go into ES-L1, GEO, HEO, or SSO orbits

Appendix

- All notebooks and python code can be found on the following repo: https://github.com/Davidlepore/CourseraTestingRepoPublic/tree/main/Capstone

Thank you!

DLepore