



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Danny Zhou>
<7/26/2025>



Outline

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

Executive Summary

- Acquired and enriched SpaceX launch data via API and web scraping
- Performed SQL-based EDA to extract insights (e.g., launch sites, booster types, orbit success)
- Visualized trends in launch success using static + interactive dashboards
- Developed machine learning models to predict mission success
- Decision Tree achieved the highest cross-validation accuracy (~88%) for training set

Introduction

- SpaceX's growing launch data provides opportunities for exploration and prediction
- Goals:
 - Understand what factors influence launch success
 - Predict success using classification models
 - Build a dashboard for exploration

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
 - Utilized SpaceX API and BeautifulSoup
- Perform data wrangling
 - Cleaned and turned class outcomes into binary results
- 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 and split the data to find the best parameters for various models

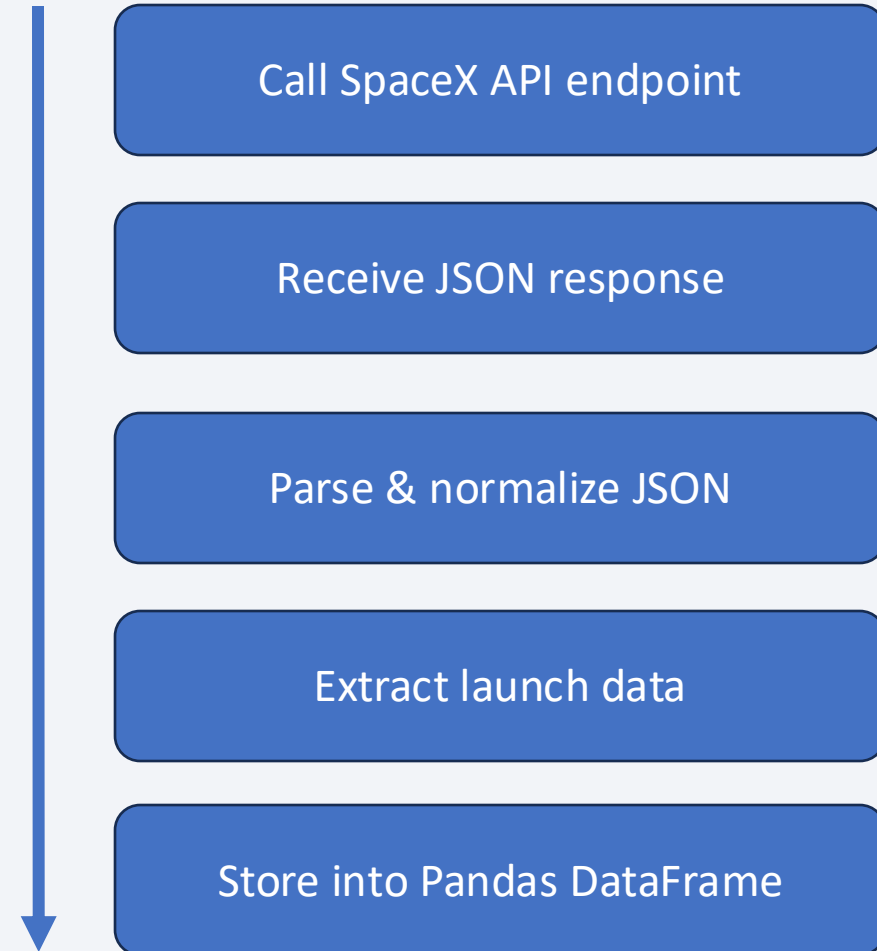
Data Collection

We gathered launch data using two techniques:

- REST API calls to structured endpoints provided by SpaceX
- Web Scraping using BeautifulSoup to extract info from Wikipedia

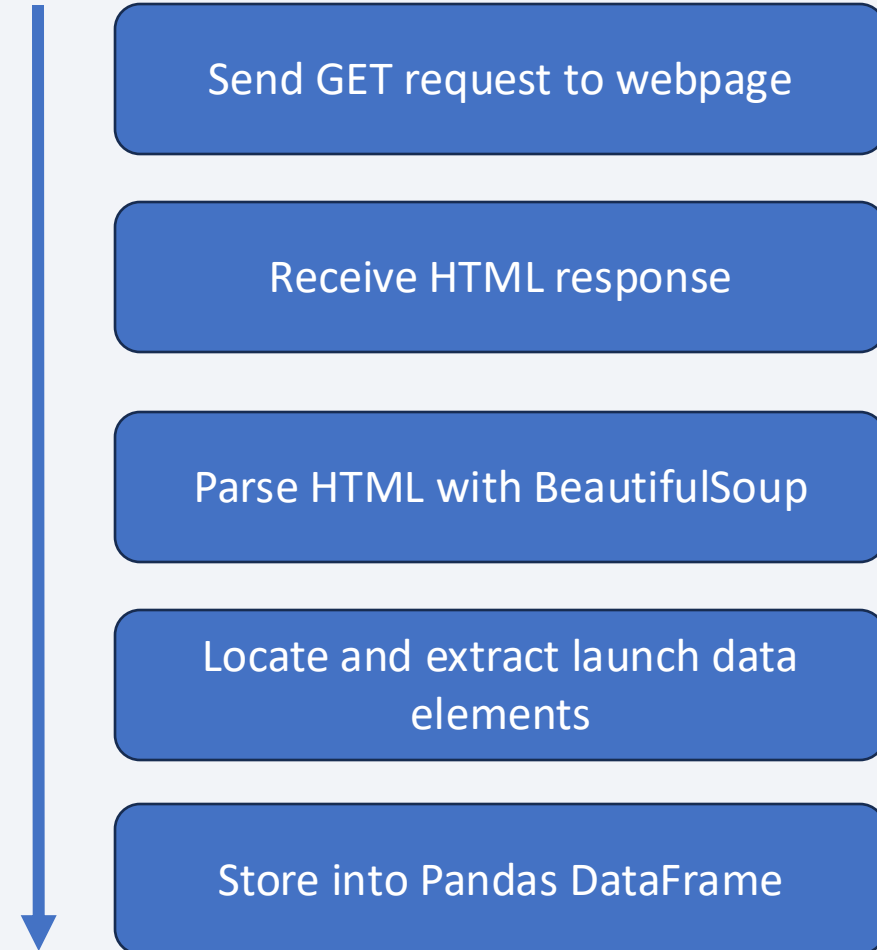
Data Collection – SpaceX API

- API calls with rich launch data
- Used GET requests with filter in /v4/launches/past
- Retrieved nested JSON: launches, rockets, payloads
- Filtered and merged data into a clean tabular format
- <https://github.com/dumburger3/ibm-final/blob/main/1-jupyter-labs-spacex-data-collection-api.ipynb>



Data Collection - Scraping

- Used BeautifulSoup with requests for scraping
- Targeted table elements containing mission data
- Cleaned raw HTML data
- Used selectors like `.find_all()` and `.text.strip()`
- Created DataFrame with the scraped data
- <https://github.com/dumburger3/ibmfinal/blob/main/2-jupyter-labs-webscraping.ipynb>



Data Wrangling

- Used Pandas and NumPy to identify weak points in the data
- Calculated the value counts of the various outcomes
- Created a column specifying success or failure of landing in binary
- <https://github.com/dumburger3/ibmfinal/blob/main/3-labs-jupyter-spacex-Data%20wrangling.ipynb>



Calculate number of launches on each site

Calculate occurrence of each orbit

Calculate occurrence of mission outcomes

Create landing outcome label from outcome column

EDA with Data Visualization

- Plotted catplot of Flight Number vs Launch Site to see the success and failures of different launch sites going from oldest to most recent flight
- Similarly created scatterplot of Payload Mass vs Launch Site for the same reasons
- Used a bar chart of success rates of each orbit as it shows the rates easily based on different categories, that being each orbit
- Created scatterplots for Flight Number vs Orbit Type and Payload Mass vs Orbit Type
- Lastly plotted a line chart of success rate by year to view the progression of success as time goes on
- <https://github.com/dumburger3/ibmfinal/blob/main/5-edadataviz.ipynb>

EDA with SQL

- Accessed the database through sqlalchemy and sqlite
- Executed SQL queries to gather information about:
 - Names of different launch sites
 - Total payload mass carried by boosters launched by NASA
 - Average payload mass carried by a specific booster version
 - Etc.
- https://github.com/dumburger3/ibmfinal/blob/main/4-jupyter-labs-eda-sql-coursera_qllite.ipynb

Build an Interactive Map with Folium


- Created a map showcasing the different launches at various launch sites using Folium
- Added markers to signify the location sites along with each specific launch. Also added circle to make it like a cluster when zoomed out to deal with clutter
- Also added lines from a launch site to other parts of the world like a city, highway, railway, etc.
- https://github.com/dumburger3/ibmfinal/blob/main/6-lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Firstly created a dropdown box with options to select info of all launch sites together or one by one
- There is a pie chart that shows the distribution of the success among all the sites or how much was success and failure for each individual one
- A range slider is made to select the amount of payload which will be set as parameters for the scatter plot
- The scatterplot will show the success and failures of the launch based on the payload range selected
- <https://github.com/dumburger3/ibmfinal/blob/main/7-spacex-dash-app.py>

Predictive Analysis (Classification)

- Created a column for the class (success and failure) and standardized the data. Standardized it and then split it to use the train test split data for modeling. Use grid search and confusion matrix for evaluation. Built models of logreg, SVM, decision tree, and KNN.
- https://github.com/dumburger3/ibmfinal/blob/main/8-SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb



Create Y test set with class column from data

Standardize the data with StandardScaler()

Use train_test_split()

Use GridSearchCV to find optimal parameters that fit the data

Evaluate with confusion matrix on test set

Results

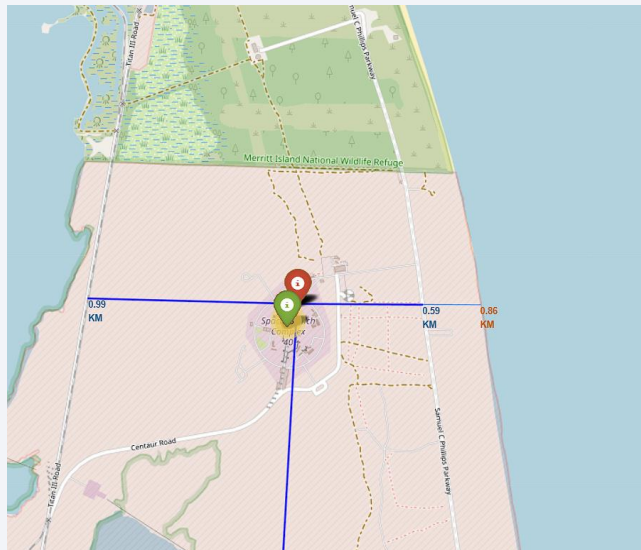
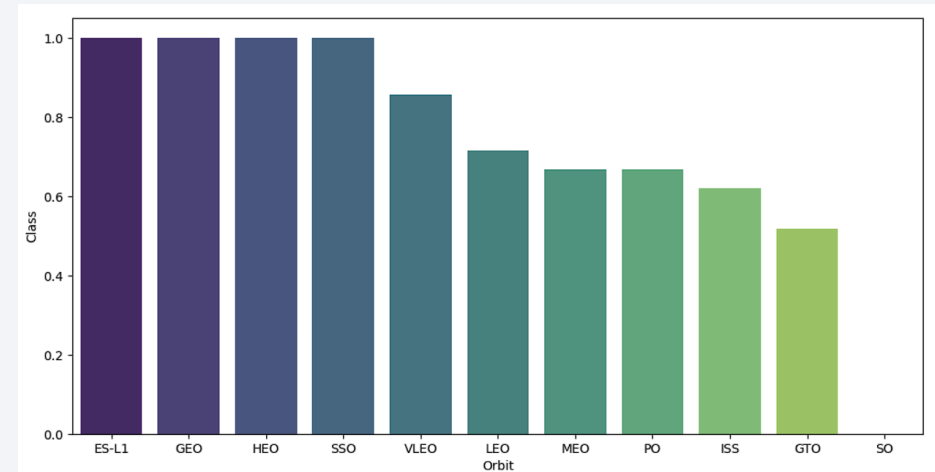
Task 3

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

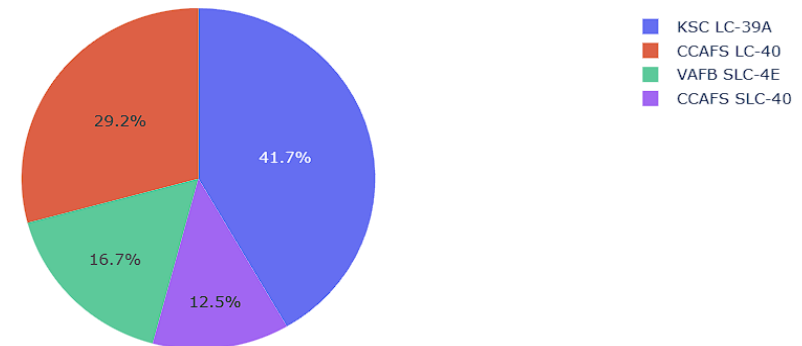
```
[13]: %sql SELECT SUM("Payload_Mass_kg_") FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)'
```

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

```
[13]: SUM("Payload_Mass_kg_")  
45596
```



Total Success Launches by Site



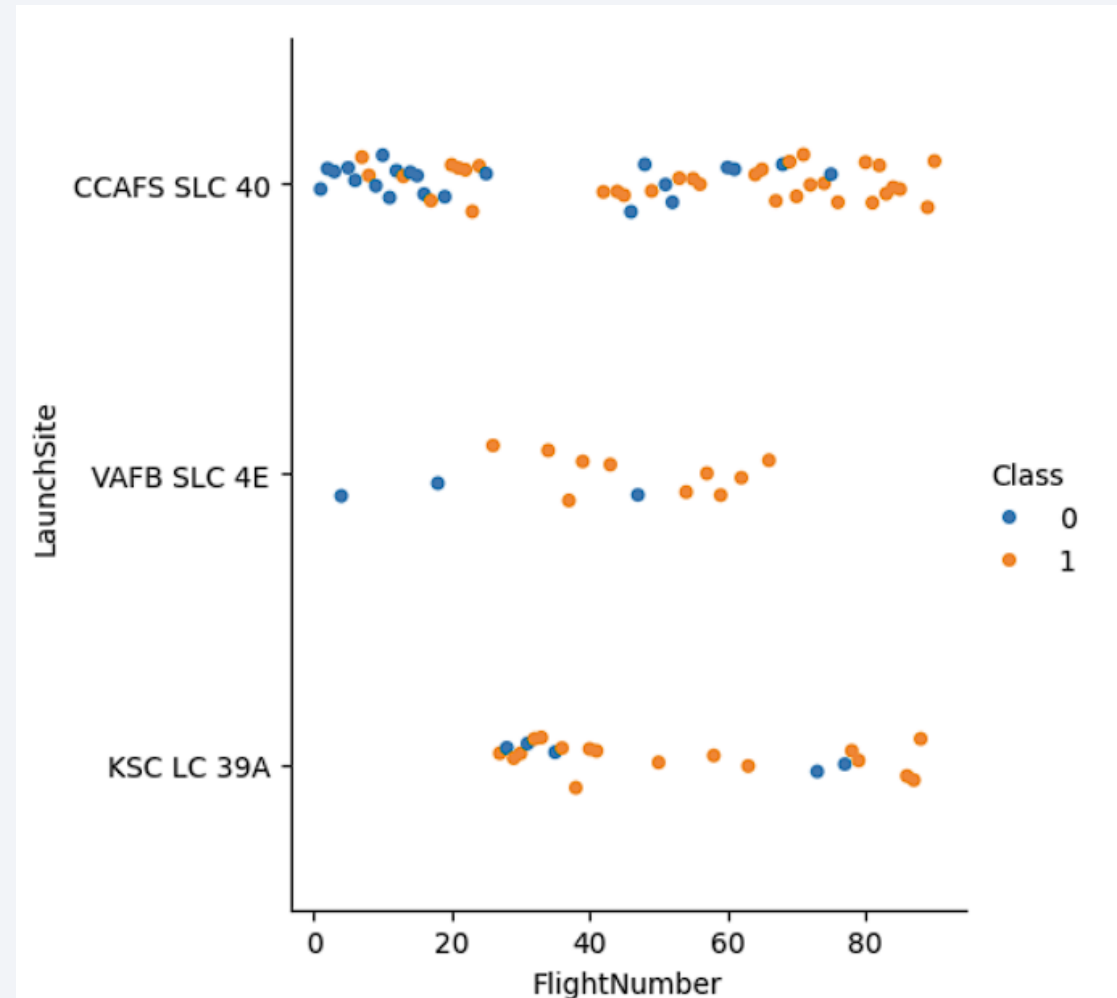
The background of the slide is an abstract composition. It features a dark blue field on the left side, which transitions into a complex pattern of diagonal streaks in shades of blue, red, and teal on the right. These streaks have a textured, almost woven appearance. Overlaid on this pattern is a faint, light blue grid that recedes into the distance, creating a sense of depth and perspective.

Section 2

Insights drawn from EDA

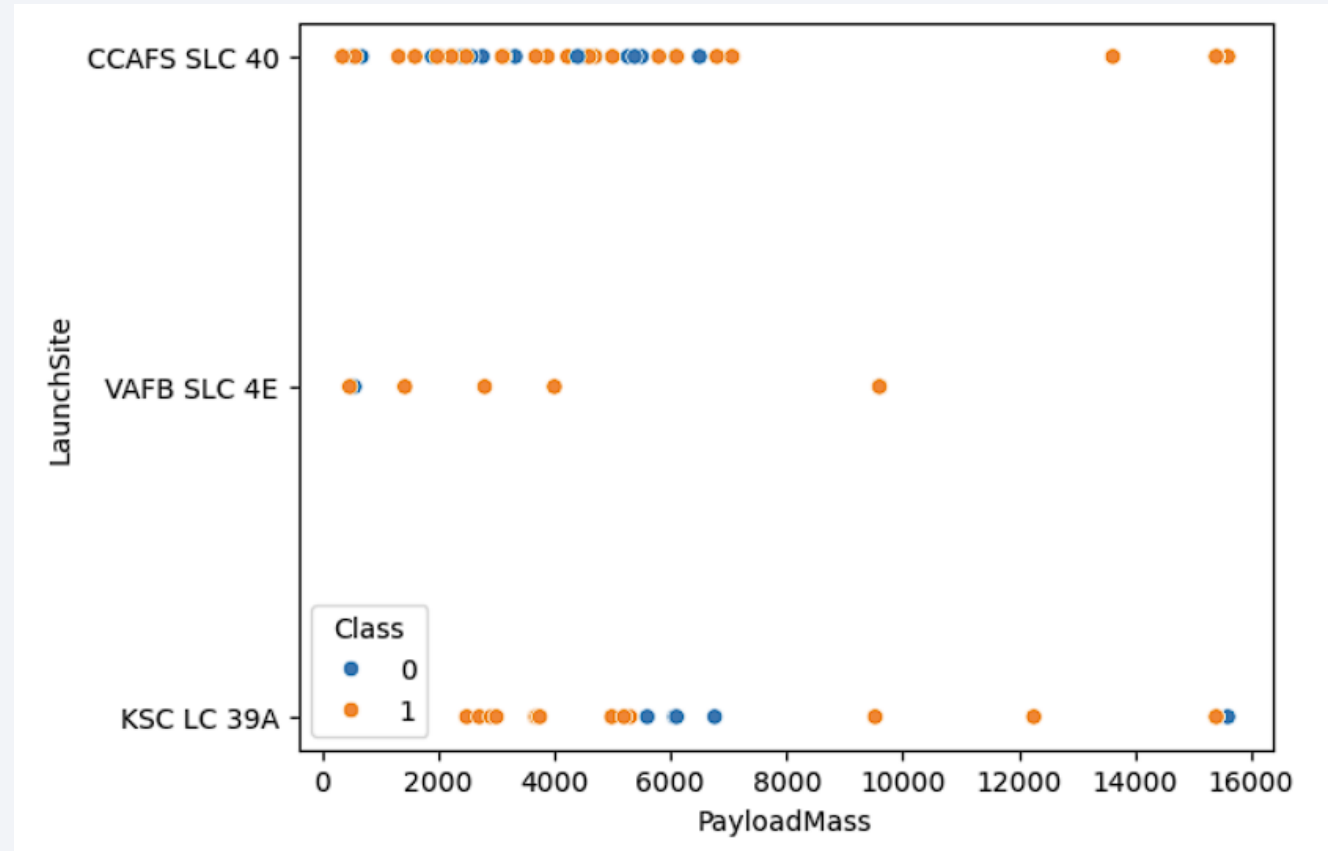
Flight Number vs. Launch Site

- For the most part, the failures mostly occur in the beginning of missions (smaller FlightNumber)
- A lot more launches in CCAFS SLC 40 than the others with VAFB SLC 4E not launching recently
- Overall can see more successes than failures



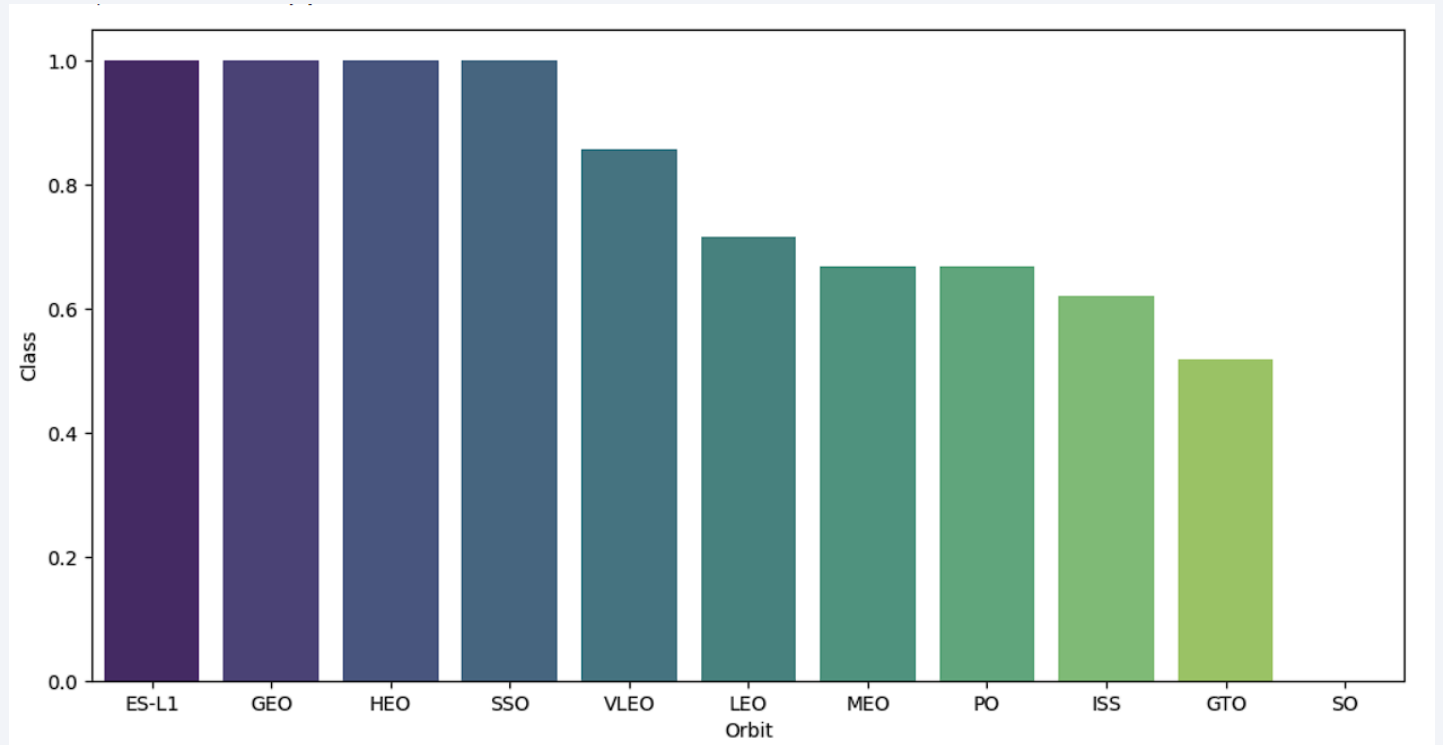
Payload vs. Launch Site

- Shows the payload with most being successes
- VAFN SLC 4E again noticeably has launched less and not at the same max payload level as the other two
- Was a failure for KSC LC 39A at their heaviest payload launch



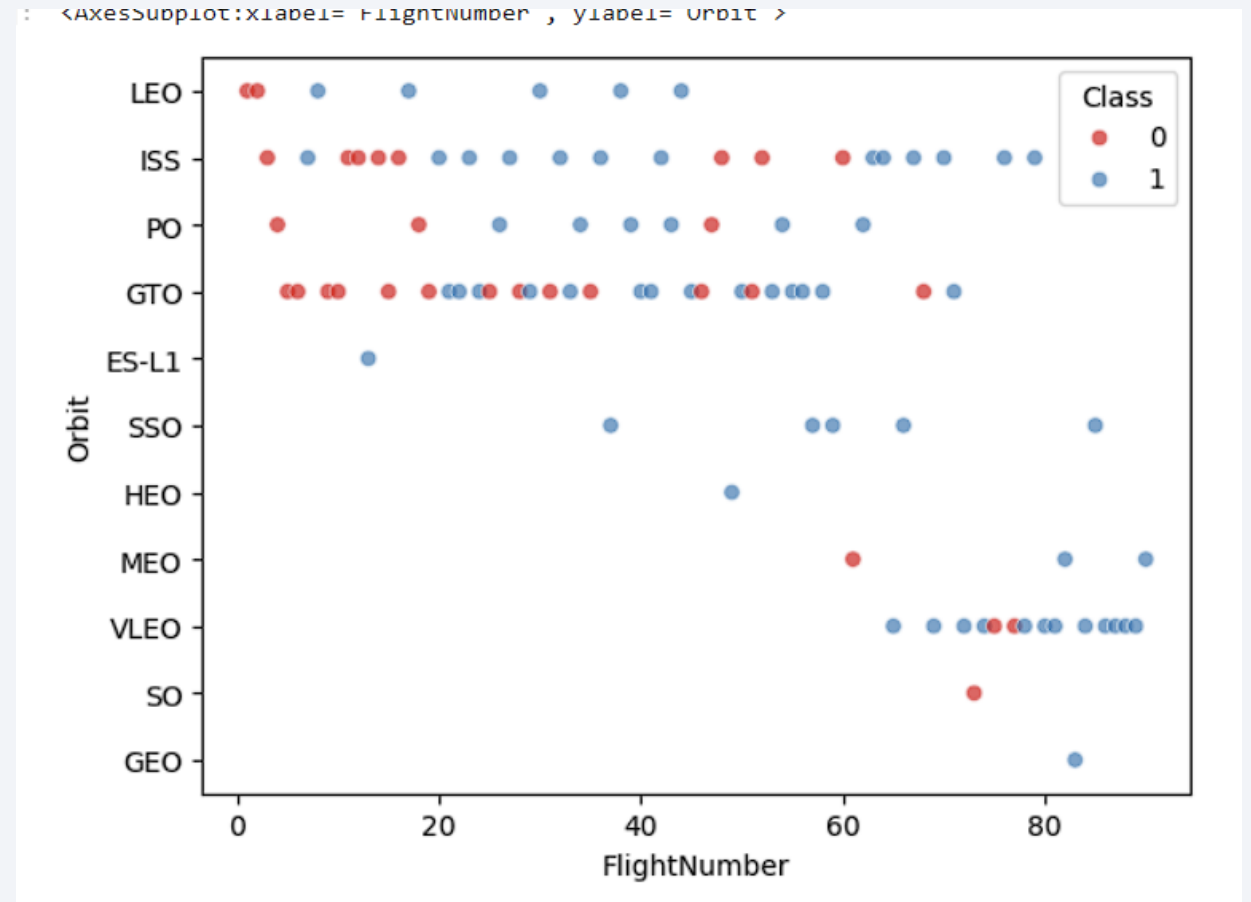
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, and SSO orbits have a 100% success rate with the other orbits having relatively around 66% on average
- The only shows success rate and though may sound great to have a high success rate, this could be due to a limited number amount of launches for that orbit



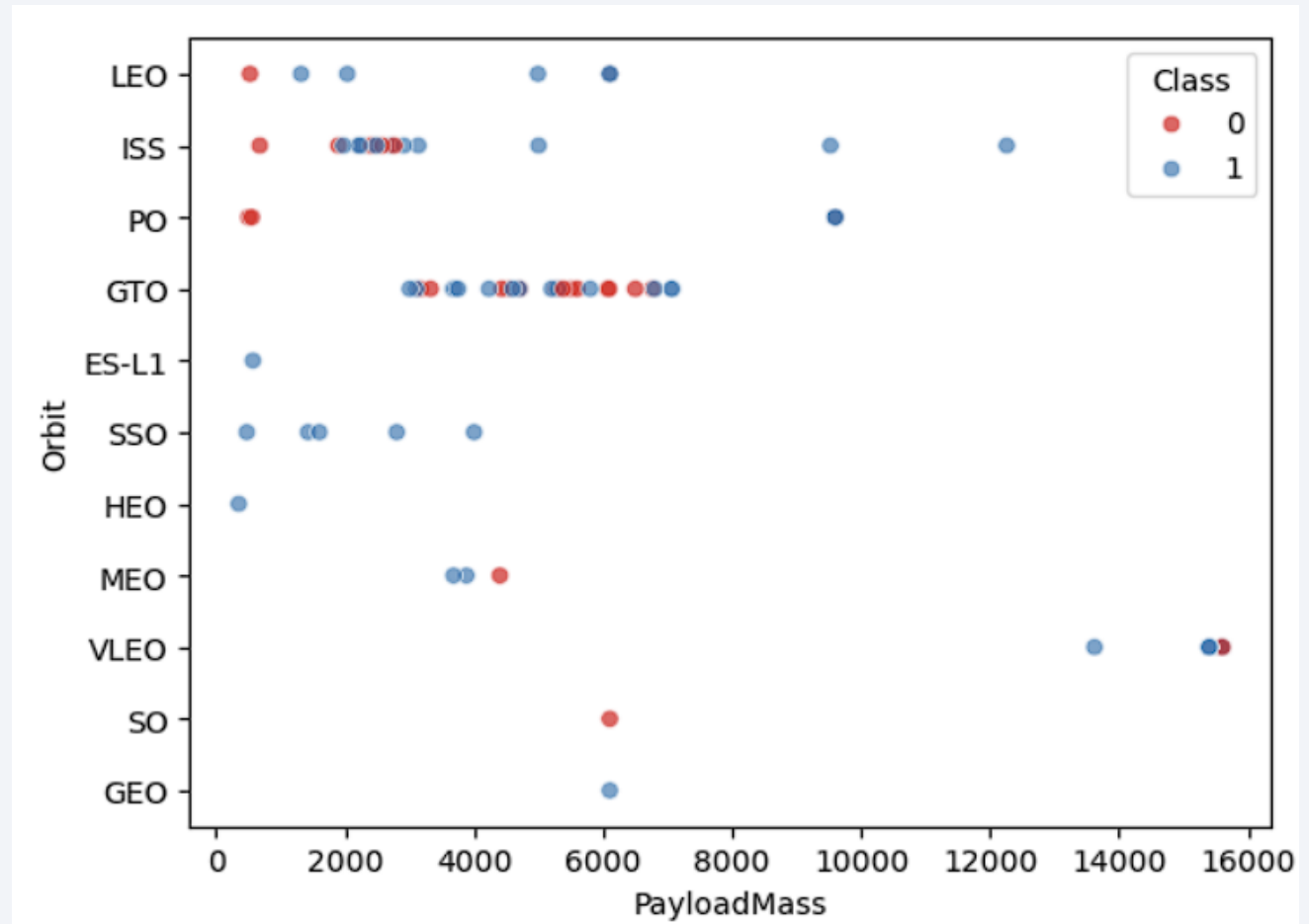
Flight Number vs. Orbit Type

- Lots of failures for the beginning launches that orbits on LEO, ISS, PO, GTO
- Later on you can see they branch out on the different orbits to be pursued with VLEO seeing a great number of success



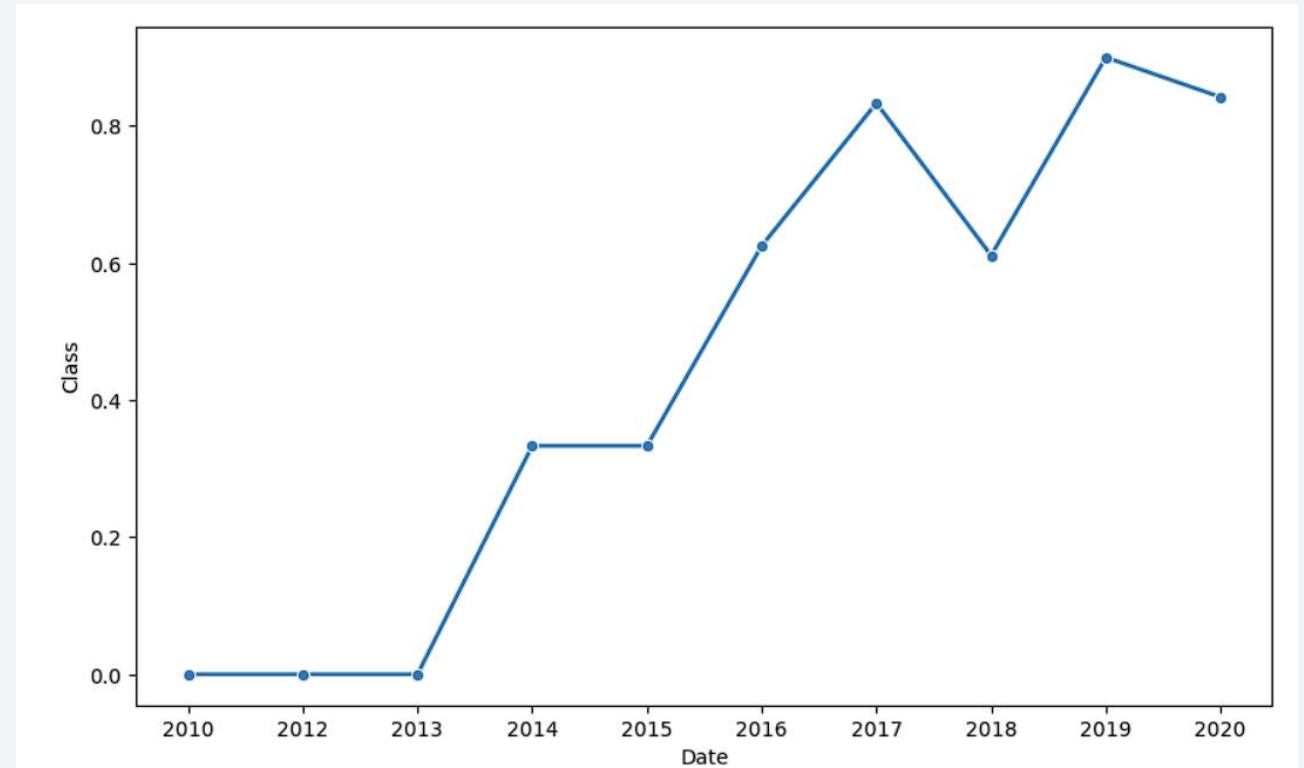
Payload vs. Orbit Type

- Notice for the most part that for certain orbits, they are only doing in a certain range of payloads (ex. SSO being in the lower payload range)
- LEO fails in very low payload mass but succeeds above their that failure



Launch Success Yearly Trend

- The success rate starts to grow at 2013 and continues to increase till 2017 when they spike down but rises again at 2019 and levels onwards



All Launch Site Names

- In ngl sql queries do not need explanation but oh well
- Distinct keyword to not get repeated ones

```
%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'

- Look for all entries that start with CCA, used like key word and limit it

Display 5 records where launch sites begin with the string 'CCA'

```
%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_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Used SUM function to add all payload mass specifically from NASA

Task 3

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

```
%sql SELECT SUM("Payload_Mass__kg_") FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)'
```

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

```
Done.
```

```
SUM("Payload_Mass__kg_")
```

```
45596
```

Average Payload Mass by F9 v1.1

- Similarly used AVG to get average specifically from F9 v1.1 booster

Task 4

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

```
%sql SELECT AVG("Payload_mass__KG_") FROM SPACEXTABLE WHERE "Booster_Version" LIKE 'F9 v1.1%'
```

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

```
Done.
```

```
AVG("Payload_mass__KG_")
```

```
2534.6666666666665
```

First Successful Ground Landing Date

- Used min function on the date to get first successful landing

Task 5

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

Hint: Use min function

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

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

Done.

first_successful_ground_pad_landing

2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Used BETWEEN keyword to get a range of number

Task 6

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

```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE "PAYLOAD_MASS_KG_" BETWEEN 4000 AND 6000 AND "Landing_Outcome" = 'Success (drone ship)'
```

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

```
Done.
```

Booster_Version

F9 FT B1022

F9 FT B1026

F9 FT B1021.2

F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- Here I just looked for all the mission outcomes and counted them but to get the exact number of success and failure you can use the case keyword and then use like keyword to get failure and success outcomes

Task 7

List the total number of successful and failure mission outcomes ¶

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") FROM SPACEXTABLE GROUP BY "Mission_Outcome"
```

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

Done.

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

Boosters Carried Maximum Payload

- Subquery where we get the booster version and payload where the payload is equal to the max in the entire table
- Outerquery just gets the name from the subquery

Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

```
%%sql
SELECT "Booster_Version"
FROM (
  SELECT "Booster_Version", "Payload_Mass_kg_"
  FROM SPACEXTABLE
  WHERE "Payload_Mass_kg_" = (
    SELECT MAX("Payload_Mass_kg_") FROM SPACEXTABLE
  )
) AS max_payload_boosters
```

```
* 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

- Used substr to get the month and year for the date which is used to acquire the 2015 launch records that are failure drone ship

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

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

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

Done.

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

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

- Instead of using substr, just use between to get the range of dates and then utilize order by to get

Task 10

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" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY outcome_count DESC
```

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

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
Failure (parachute)	2
Precluded (drone ship)	1

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The image is a composite of a solid blue background on the left and a satellite photograph of Earth on the right. The Earth's surface is dark blue, with numerous bright yellow and orange lights representing cities and urban areas. The horizon line of the Earth is visible, separating the dark surface from the blackness of space.

Section 3

Launch Sites Proximities Analysis

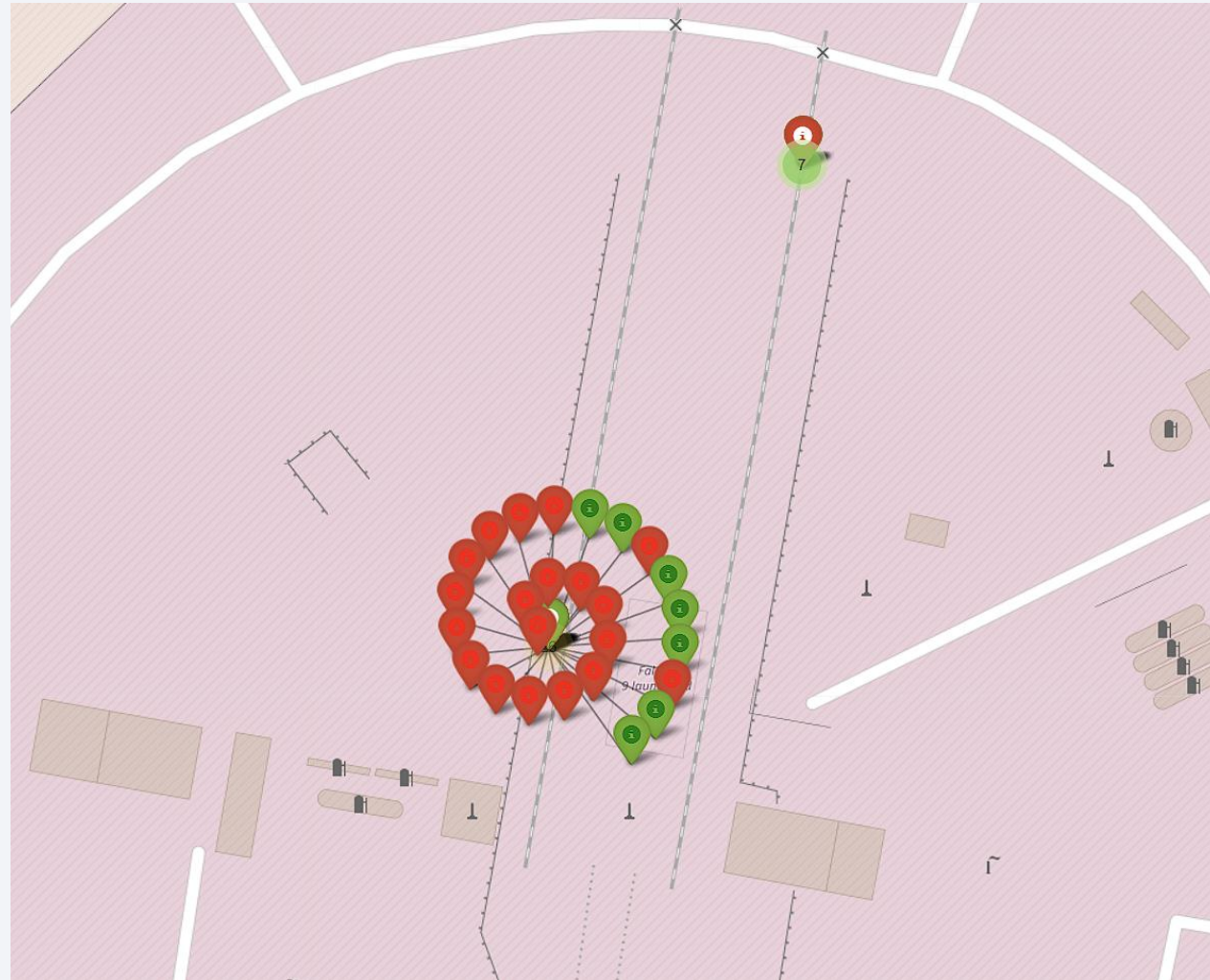
Launch Site Map

- You can see around two dots on the right and one on the left which represents the launch sites
- Notice how they are both by the water



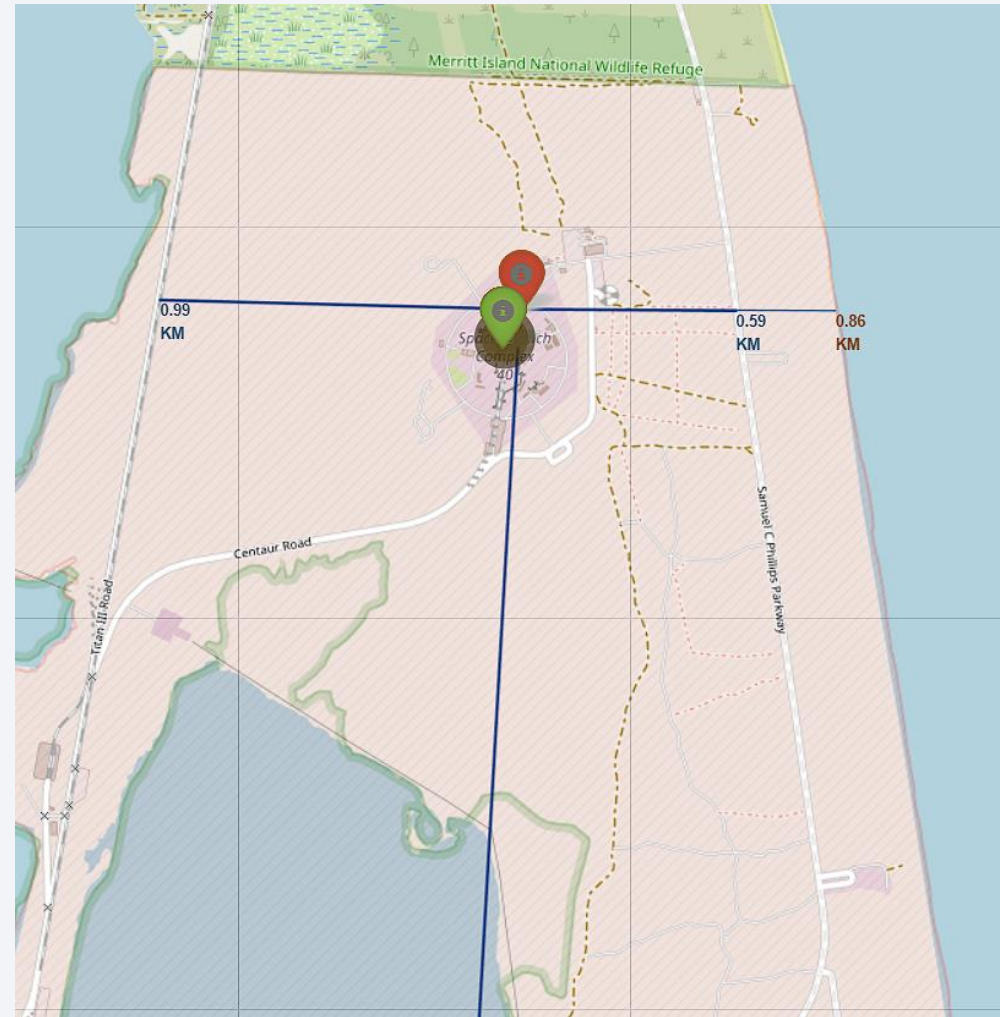
Launch Outcome Map

- The map now showcases the number of launches associated with each launch site
- Once clicked on the location it'll show whether it was a success or failure as well



Launch Site Distance Map

- Here the map also shows the distance of the launch sites to different parts of the world around them
- The coastline, highway, and railway are close to the launch site being around 0.6 to 1 KM away, whereas the closest city, Melbourne, is 54 KM away south





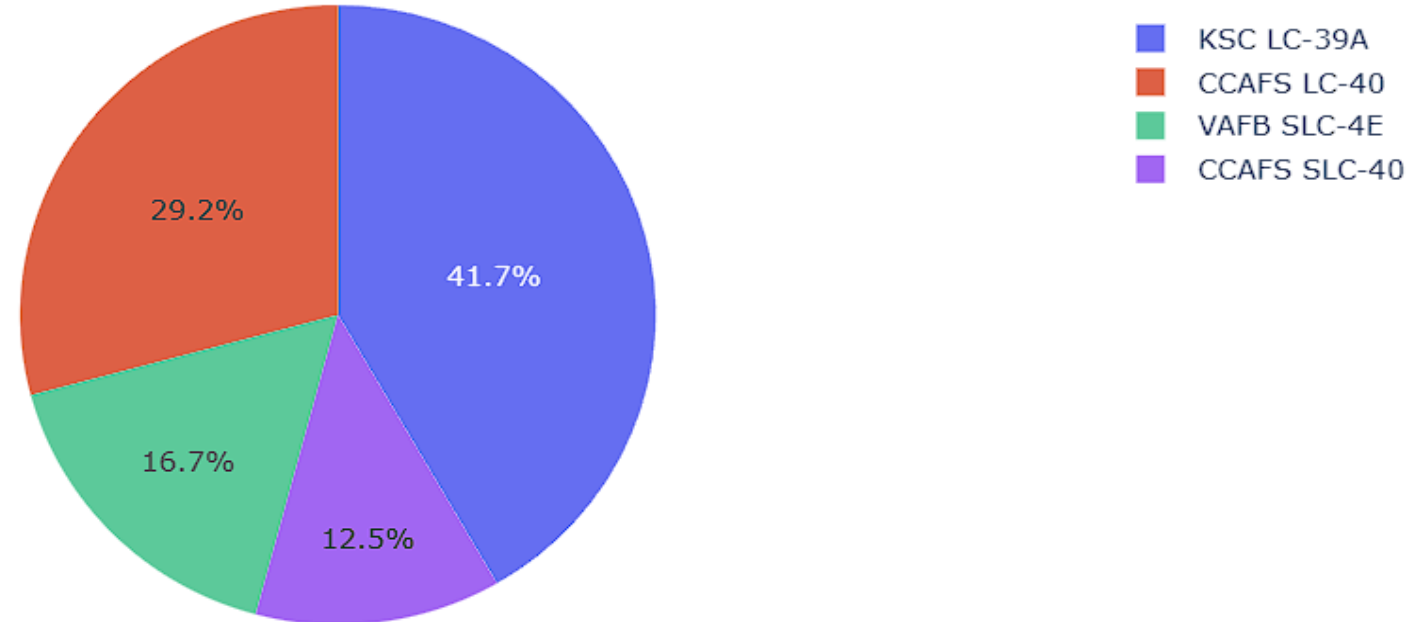
Section 4

Build a Dashboard with Plotly Dash

Launch Success of All Sites Pie Chart

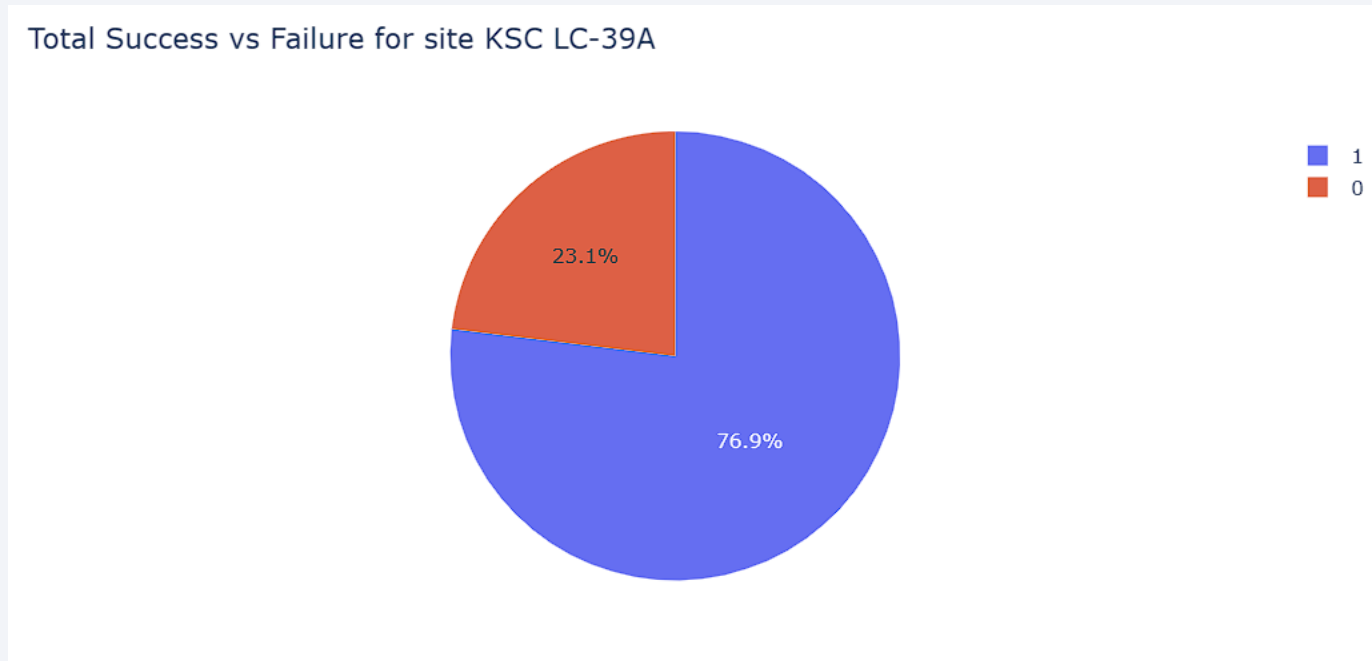
- We can see the distribution of success across all the launch sites
- KSC LC-39A has the greatest portion of success while CCAFS SLC-40 finds the worst

Total Success Launches by Site



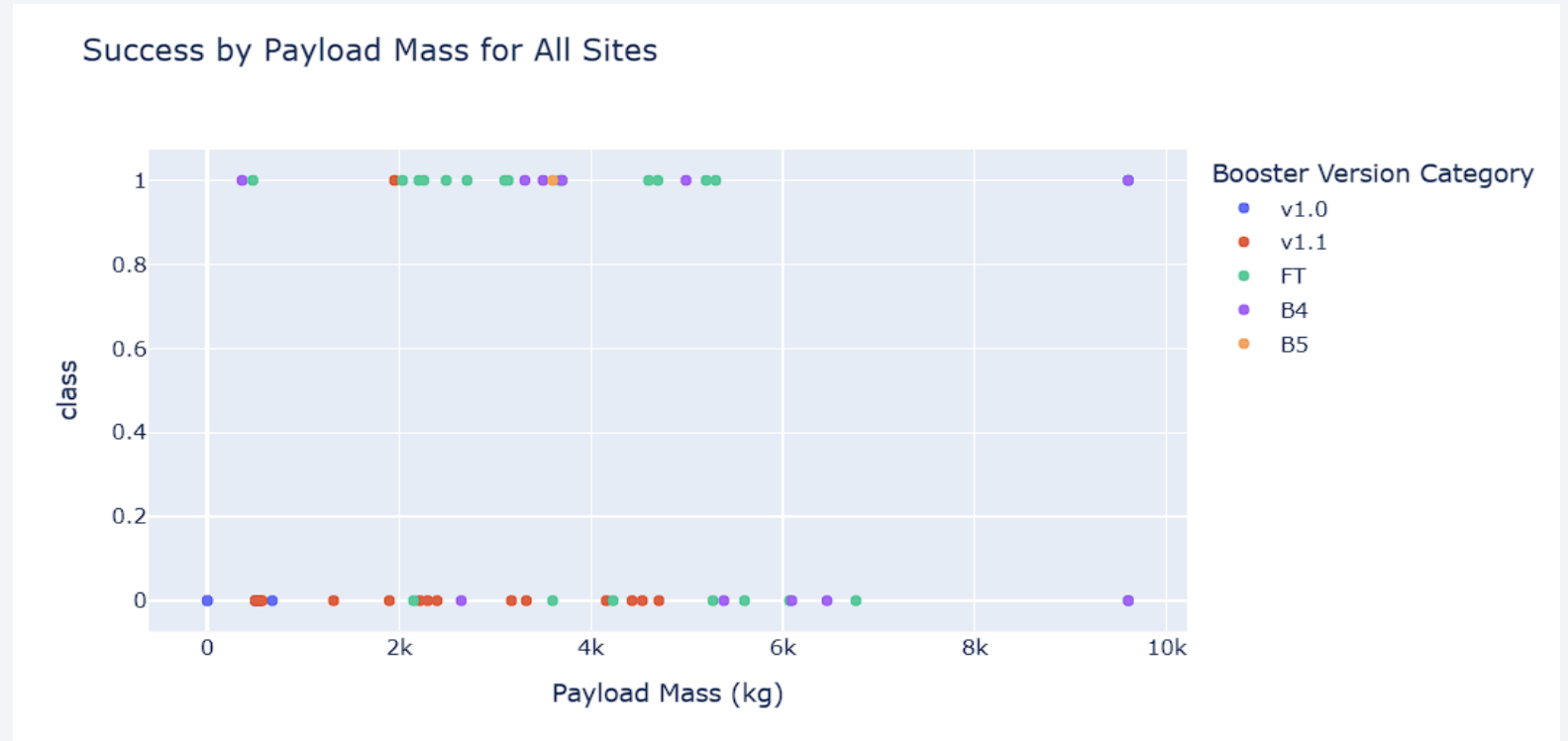
Highest Success Rate Launch Site Pie Chart

- Here we see the launch site with the highest success rate, KSC LC-39A
- There is more than 3/4ths success in all their launches which is significantly good



<Dashboard Screenshot 3>

- We can get a good amount of info from this chart which showcases the successes and failures of launches at different payloads and varying booster version
- Ex. Only B4 Boosters have been used for high mass launches and experienced both a success and a failure
- V1.1 booster has almost a 100% failure rate as there seems to be only one success

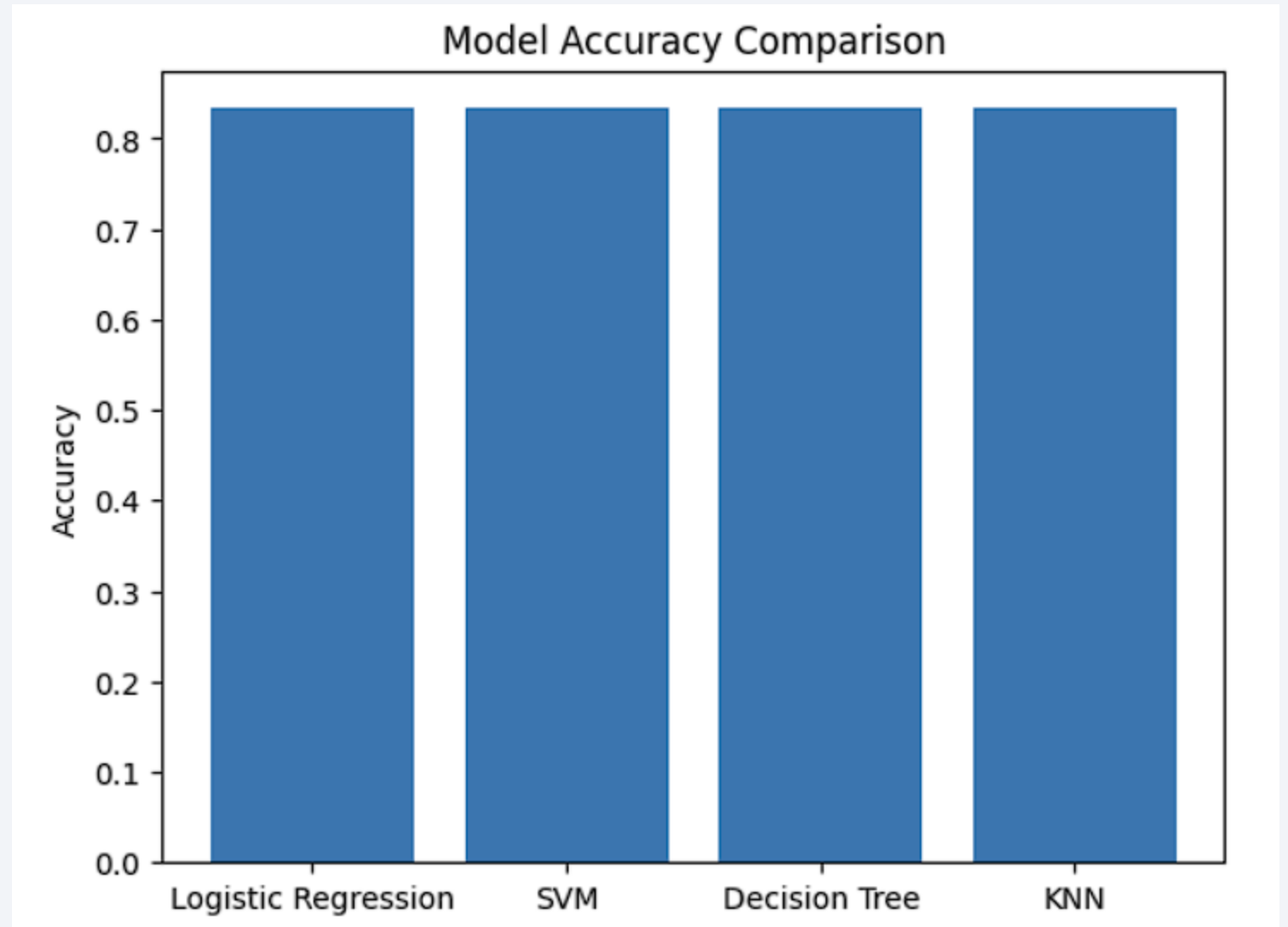


Section 5

Predictive Analysis (Classification)

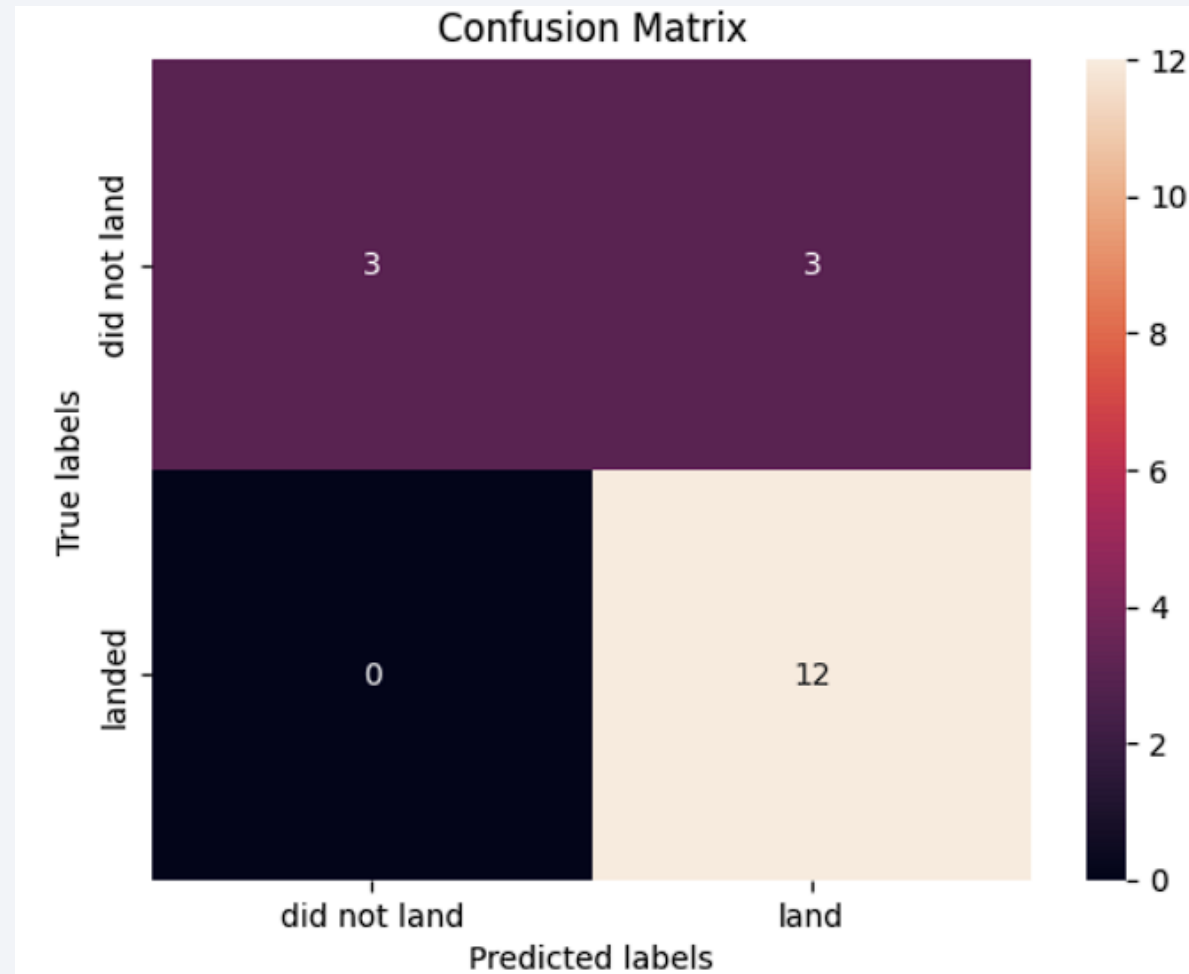
Classification Accuracy

- All the machine learning algorithms, that being logistic regression, svm, decision tree, and knn, achieved the same accuracy when it comes to the test set
- I chose decision tree as the best classifier tho as it had the highest accuracy on training data at 88.75%



Confusion Matrix

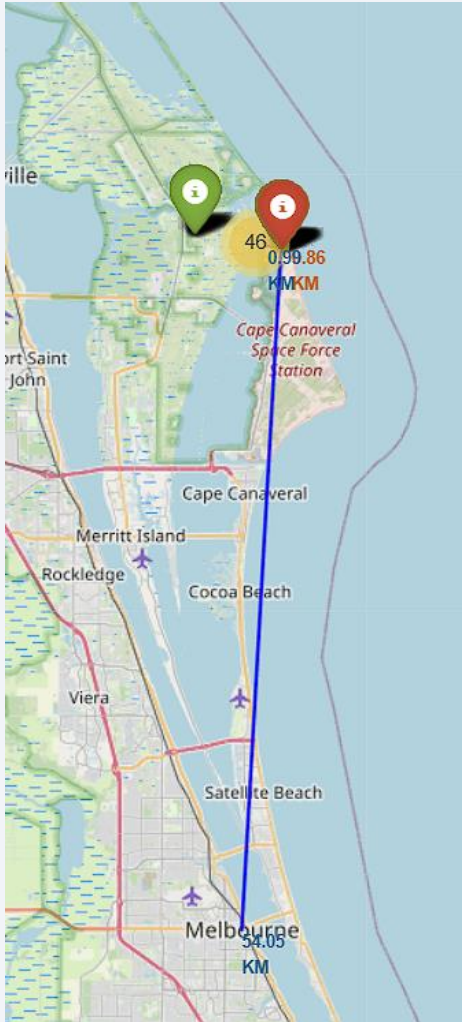
- The model was able to perfectly predict all the launches that actually landed (12)
- However it struggled with false positives as it marked 3 of the launches that did not land as land, which if it was utilized in the real world can result in major resource lost



Conclusions

- Payload mass and booster type significantly impact launch success
- SQL & Dash reveal trends, while ML predicts outcome
- Launch sites should remain near coastal lines and far from cities
- Decision Tree = kind of best performing model

Appendix



Thank you!

