IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Vishal Purohit
18/03/2024

Outline

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

Executive Summary

Summary of methodologies

Data was collected from 2 sources: a REST api and web scraping Wikipedia SpaceX falcon 9 past launches page

The data was checked for null values and data types. Payload mass data was imputed with its mean value. Categorical variables were one-hot-encoded. The Outcome variable was converted into a binary target variable by defining good and bad outcomes. The data was standardized. We plot various variables against each other to see how they correlate. We use SQL to query the data to learn about various aspects of it. We visualize map information to see where launch sites and located and infrastructure distances. We also created an interactive dashboard that allows booster information at different launch sites to be queried. The data was split into a train and a test set. Using the train set we used 4 different algorithms (Logistic regression, SVM, KNN and Decision trees) with GridSearchCV. GridSearchCV allows us to try different hyperparameters for the algorithms that need to be set before the algorithm is run. This allows us to find the best hyperparameters to improve the accuracy of the algorithm. We set the cross-validation splits within gridsearchcv to 10 to train on 90% of the data and test on 10%. Once the best parameters are found we tested the algorithm on the test set. The results from the 4 algorithms were put into a bar chart to compare them.

Summary of all results

- We can predict whether Space X's first stage will land successfully with an accuracy of 83%. The main issue we have with our predictions is false positives. If we had additional information about cost of false positives and false negatives, we could use ROC AUC to change the probability threshold to, for example exchange false positives for false negatives.
- Space X uses 3 launch sites. Two on the east coast and one on the west coast. The site CCAFS SLC-40 is the closest of the 3 to the equator at a distance of 3,177.08 km. All 3 sites are located on the coast. CCAFS SLC-40 is 0.88 km away from the coast. Close by infrastructure are roads (0.59 km) and rail (4.28 km). The nearest city to CCAFS SLC-40 is Cape Canaveral and is 18.14 km away.
- B5 boosters have a best success rate with FT a close second. We saw from our SQL queries that B5 boosters can launch payloads of 15,600 kg. FT rockets are capable of launching all payload up to 9,600 kg.
- B4, v1.1 and v1.0 are all low success boosters. B4 boosters are newer than FT, but have lower success rates. We may see more FT launches in the future. V1.0 and v1.1 launched less the 5,000 kg payloads and are no longer used.

Introduction

- In this project, we are taking on the role of a competitor to Space X called Space Y.
- Space X's Falcon 9 rocket costs 62 Million Dollars to launch. The cost of launches from of other companies can be in the range of 165 Millions Dollars or higher. This cost saving comes from the fact that the first stage of Space X's rocket is reusable. Thus, the successful landing of the first stage of the rocket means a lot of money.
- As Space Y, we will be bidding against Space X. To maximise the success rate of our bids, we need to know:
    - if Space X's first stage will land successfully. To do this, we will use machine learning to predict landing outcomes.
    - what kind of facilities Space X has; where they are located and what infrastructure is nearby. This is to help us understand where to launch our rockets.
    - what boosters have a high landing success rates and which ones have low success rates as well as the kinds of payloads they are capable of launching. This is to save money on R&D costs.

Section 1

Methodology

Methodology

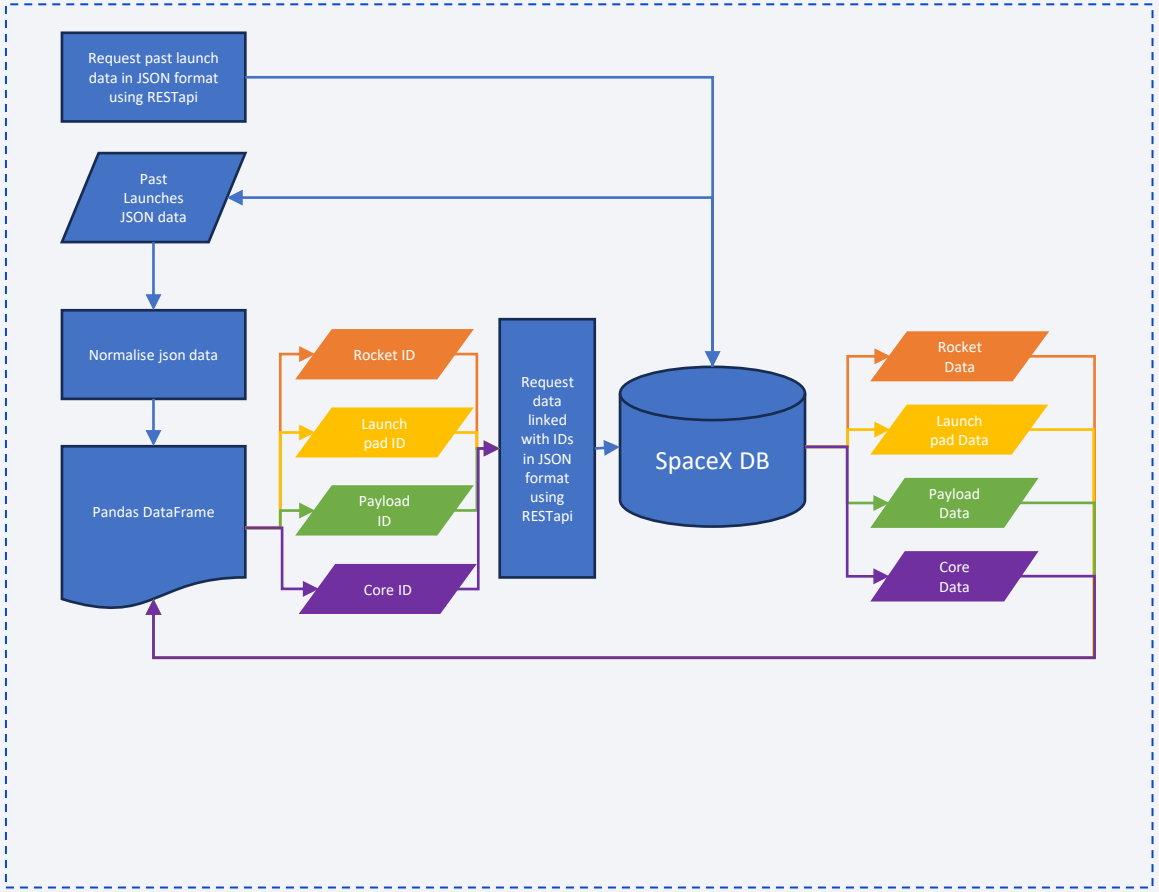
Executive Summary

- Data collection methodology:
    - Data was collected from 2 sources. The first was the SpaceX REST api. From this api we were able to get data about past launches, however we needed to query several times as many variable values were codes that needed to be looked up. The second data source was the Wikipedia SpaceX falcon 9 past launches page, from which we scraped data in tabular form.
- Perform data wrangling
    - The data was checked for null values and whether the data type made sense for each variable. Variables will null values (payload mass) were imputed with the mean value. Categorical variables were one-hot-encoded. The Outcome variable was converted into a binary target variable by defining good and bad outcomes. The data was then standardized.
- Perform exploratory data analysis (EDA) using visualization and SQL
    - We plot various variables against each other to see how they correlate. We use SQL to query the data to learn about various aspects of it, such as the maximum payload and which boosters can launch with it.
- Perform interactive visual analytics using Folium and Plotly Dash
    - We visualize map information to see where launch sites and located and infrastructure distances. We also created an interactive dashboard that allows booster information at different launch sites to be queried.
- Perform predictive analysis using classification models
    - The data was split into a train and a test set. Using the train set we used 4 different algorithms (Logistic regression, SVM, KNN and Decision trees) with GridSearchCV. GridSearchCV allows us to try different hyperparameters for the algorithms that need to be set before the algorithm is run. This allows us to find the best hyperparameters to improve the accuracy of the algorithm. We set the cross-validation splits within gridsearchcv to 10 to train on 90% of the data and test on 10%. Once the best parameters are found we tested the algorithm on the test set. The results from the 4 algorithms were put into a bar chart to compare them. All 4 algorithms yielded an accuracy of 83%.

Data Collection

Data Collection – SpaceX API

- Past launch data was obtained from SpaceX using the python command ***requests.get()*** and the url: https://api.spacexdata.com/v4/launches/past. The request was successful (status code: 200) and content of the response was normalized (***pandas.json_normalize(response.json())***) allowing the data to be viewed in a tabular format (***pandas.DataFrame()***).
- The columns: 'rocket', 'payloads', 'launchpad' and 'cores' contained IDs. SpaceX REST calls were made for each ID in each column to replace the IDs with data related to them. From the Rocket IDs we get booster type (BoosterVersion). From the Payload IDs we get the mass of the payload (Mass_kg) and the type of orbit (Orbit). From the Launpad IDs we get launch site name (LaunchSite) as well as the latitude (Latitude) and longitude (Longitude). From the Core IDs we get serial number (Serial), number of flights of the core (Flights), the landing outcomes together with their type (Outcome), if the core had gridfins (GridFins), if it was reused (Reused) and how many times (ReusedCount), if it had legs (Legs), the landing pad ID (LandingPad) and the block number/core version (Block). We restrict our data to rockets with a single Core and a single Payload.
- The SpaceX REST call used to obtain the data from the IDs were:
  - ***requests.get("https://api.spacexdata.com/v4/rockets/"+str(ROCKET_ID)).json()***
  - ***requests.get("https://api.spacexdata.com/v4/payloads/"+PAYLOAD_ID).json()***
  - ***requests.get("https://api.spacexdata.com/v4/launchpads/"+str(LAUNCHPAD_ID)).json()***
  - ***requests.get("https://api.spacexdata.com/v4/cores/"+CORE_ID).json()***



https://github.com/vishpuro/Coursera/blob/cd2e893f0e0e404f3c040b5106176654551d2d75/capstone/jupyter-labs-spacex-data-collection-api.ipynb

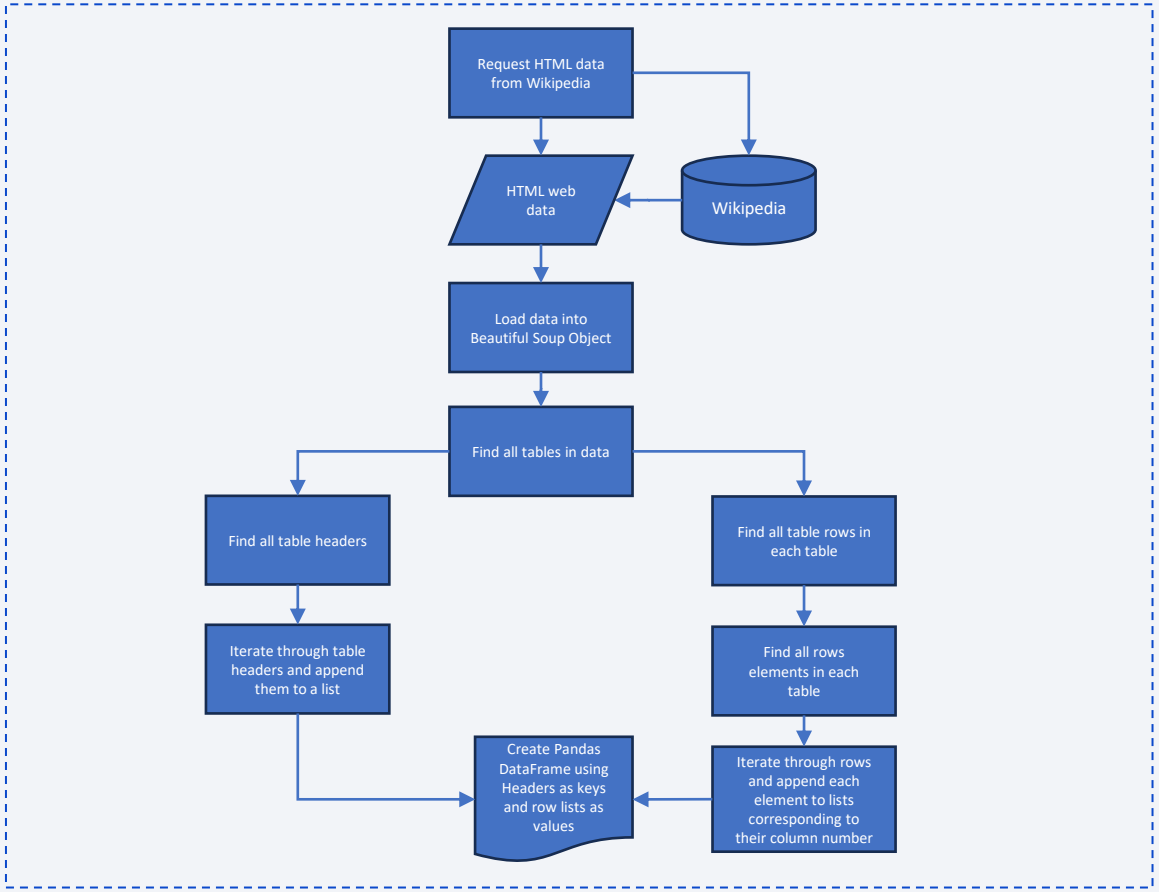
Data Collection - Scraping

- HTML data was obtained from Wikipedia using the python command ***requests.get()*** with the url: https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922 . The request was successful (status code: 200) and content of the response was loaded into a Beautiful Soup object (***BeautifulSoup(response.text)***), which is a parsed HTML data structure.

- Using the command ***.find_all('table')*** we can find all of the tables in the HTML data. Using one of the tables, we can find all of the table headers ***.find_all('th')*** and append them to a list. From that list we can create a dictionary ***dict.fromkeys(column_names)*** to store the information.

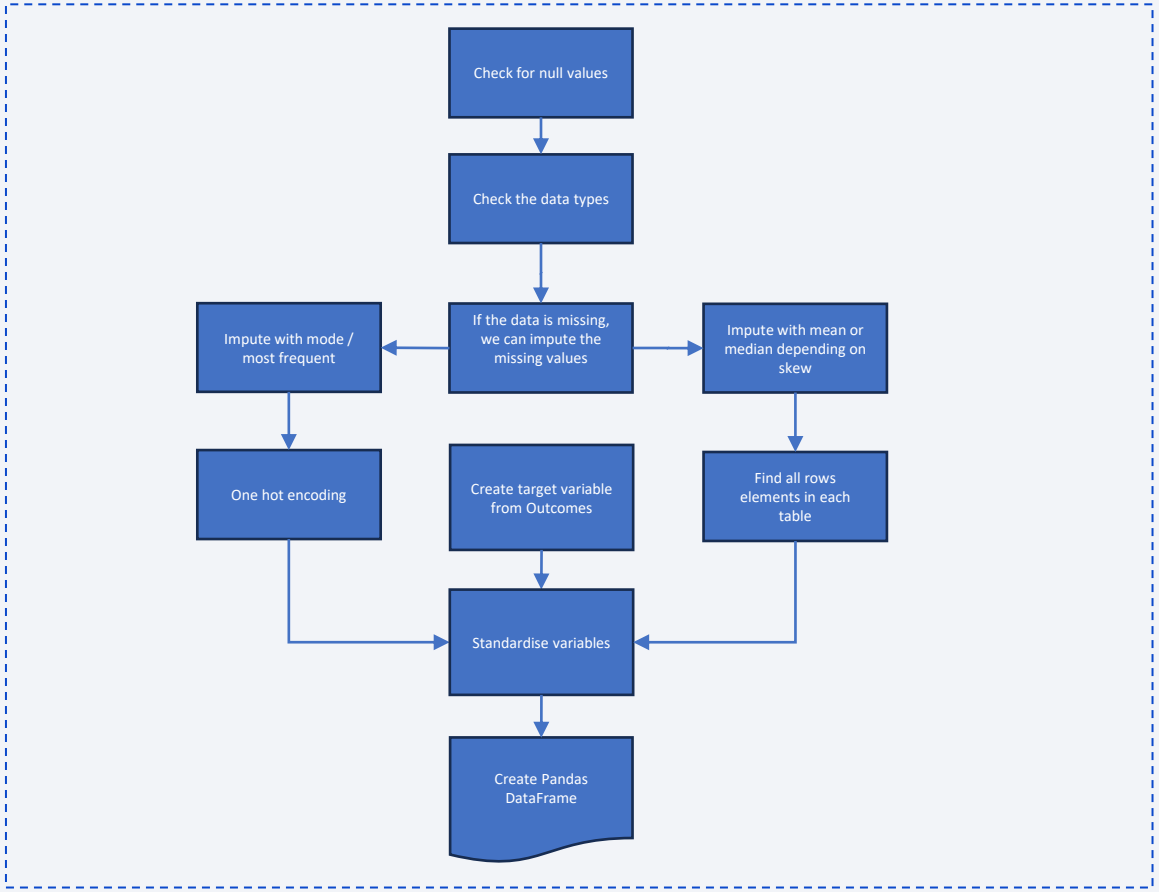
- Using the command ***.find_all('tr')*** on the tables we have already found, we can find all the table rows in those tables. On the rows, we can use ***.find_all('td')*** to find all the table data in that row. Each element in the row corresponds to a table header so they are appended to the corresponding column of the dictionary. Once the dictionary is populated, it is converted into a DataFrame (***pandas.DataFrame()***)

https://github.com/vishpuro/Coursera/blob/cd2e893f0e0e404f3c040b5106176654551d2d75/capstone/jupyter-labs-webscraping.ipynb



Data Wrangling

- Check features for null/missing values. We see that “landing pad” and “payload mass” have missing data. For “payload mass” we can impute the data, replacing the null values with the mean payload mass.
- Check the types of each of the feature columns and see if they make sense. We see that there are some categorical variables and some numeric variables. There do not appear to be any data types that don’t match the variables, except “Date” which is of the Object
- We can turn “Outcome”, which has 8 unique values into “Class” with 2 values (0:fail, 1:success). We do this by defining good and bad outcomes, then iterate through the rows and adding a 0 where Outcome is bad and a 1 where the outcome is good.
- The categorical variables like “orbit type” and “launch site” do not have an order to them, so to make sure that the algorithms don’t assume this kind of relationship, we one-hot-encode (OHE) the variables. This turns each unique value in a column into its own binary value column.
- The data in “payload mass” is in the thousands while some columns are Boolean. To ensure that all the columns are in the same range and therefore treated as equally important, we standardise the data by subtracting the mean of each column and dividing by its standard deviation.



https://github.com/vishpuro/Coursera/blob/1cfb9355102923e63c92305658ee892bb0e4a876/capstone/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

To explore the data, variables were plotted against each other to find relationship between them.

- We started by plotting the **payload mass (kg)** (dependent variable) as a function of the **flight number** (independent variable). The flight number represents flight attempts in chronological order. The data points were colour coded to represent success or failure **class** of first stage landing. The reason we plot this relationship is to see how the payload mass changes over time and also how it affects the success of the first stage landing.
- Next, we visualize the relationship between **flight number** and **launch site** with the colour of the markers indicating success or failure **class**. The size of the markers has been set to reflect the **payload mass**. We saw from the previous graph that the success rate of the flights increased over time. Now we want to see if the launch site affects the success of the first stage landing.
- We next plot the relationship between **payload mass** and **launch site.** Since we saw in the first graph that there was a relationship between payload mass and flight number, we want to see if different payloads are launched from different sites.
- We plot the average success rate (**class**) of landings from launches to different **orbits** to see if the orbit types have an affect the success of landings.
- We plot **orbit** type against **flight number,** this will show us if flights to different orbits are related over time.
- We plot the **payload mass** against **orbit** type to see if the different orbits have different payloads.
- We plot **Date (year)** against average success rate. We have been plotting variables against flight number, but the time between flight has not been considered.

https://github.com/vishpuro/Coursera/blob/cd2e893f0e0e404f3c040b5106176654551d2d75/capstone/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

To explore the SQL data, we preformed the following queries:

1. To find the list of SpaceX launch sites we queried all unique **launch sites** from the table. To do this we used the DISTINCT command. **SELECT DISTINCT** launch_site **FROM** SPACEXTABLE

2. We then wanted to see what kind of data is present in the table, so we queried 5 records with **CCAFS** launch site To do this we used the WHERE LIKE commands to filter the results and the LIMIT command to select only 5 records. **SELECT** * **FROM** SPACEXTABLE **WHERE** launch_site **LIKE** 'CCA%' **LIMIT** 5

3. We wanted to see the total **payload mass** carried for customers for example **NASA (CRS)**. To do this we used the SUM command with the WHERE LIKE commands to filter the results. **SELECT** customer, sum(PAYLOAD_MASS__KG_) **AS** Total_payload **FROM** SPACEXTABLE **WHERE** customer **LIKE** 'NASA (CRS)%'

4. We wanted to see the average **payload mass** carried by different **booster versions**, for example **F9 v1.1**. To do this we used the AVG function to get the mean with the WHERE command to filter the results. **SELECT** booster_version, AVG(PAYLOAD_MASS__KG_) **AS** Average_payload **FROM** SPACEXTABLE **WHERE** booster_version = 'F9 v1.1'

5. We wanted to see when the first successful first stage landing on a **ground pad**. To do this we used the MIN command to get the lowest value of **DATE** with the WHERE LIKE commands to filter the results. **SELECT** min(Date) **AS** First_Success **FROM** SPACEXTABLE **WHERE** landing_outcome **LIKE** 'success (ground pad)'

6. We wanted to see which **boosters** successfully landed on a **drone ship** with **payload masses** between 4000 and 6000 kg. To do this we used the WHERE LIKE commands to filter the results with an AND command to use multiple filter conditions. The second condition used the BETWEEN command which allows filtering between two values. **SELECT** booster_version **FROM** SPACEXTABLE **WHERE** landing_outcome **LIKE** 'success (drone ship)' **AND** PAYLOAD_MASS__KG_ **BETWEEN** 4000 **AND** 6000

7. We wanted to see how many **mission** successes and failures there were. To do this we used two subqueries with the COUNT command with the WHERE LIKE commands to filter the results. **SELECT**(**SELECT** count(mission_outcome) **FROM** SPACEXTABLE **WHERE** mission_outcome **LIKE** "success%") **AS** successes, (**SELECT** count(mission_outcome) **FROM** SPACEXTABLE **WHERE** mission_outcome like "failure%") **AS** failures **FROM** SPACEXTABLE **LIMIT** 1

8. We wanted to see which **booster versions** carried the maximum **payload mass**. To do this we used the DISTINCT command to get unique booster versions and the WHERE command to filter the results. The WHERE equation is found using a subquery with the MAX command to find the largest payload mass value. We then ordered the results by booster version name. **SELECT DISTINCT** booster_version, PAYLOAD_MASS__KG_ **AS** Payload_Mass **FROM** SPACEXTABLE **WHERE** PAYLOAD_MASS__KG_ = (**SELECT** max(PAYLOAD_MASS__KG_) **FROM** SPACEXTABLE) **ORDER BY** booster_version

9. We wanted to see the **dates** for **drone ship landing failures** in **2015**, which **boosters** they had, and which **launch site** they came from. To do this we used the SUBSTR command since sqllite is not able to handle month names. SUBSTR(COLUMN,6,2) gets 2 characters starting at character 6. Since the DATE is in the format YYYY-MM-DD, this gives us the month. We also use the WHERE command to filter the results with an AND command to use 2 conditions. **SELECT** substr(Date, 6,2) **AS** Month, landing_outcome, booster_version, launch_site, substr(Date,0,5) **AS** Year **FROM** SPACEXTABLE **WHERE** substr(Date,0,5)='2015' **AND** landing_outcome = 'Failure (drone ship)'

10. We wanted to see how many of each landing **outcome** there are and rank them for **dates** between 04/06/2012 and 20/03/2017 in **descending order**. To do this we used the COUNT command and the WHERE and BETWEEN commands to filter the results to dates between 04/06/2012 and 20/03/2017. We then use the ORDER command to sort the results and the DESC command to sort in descending order. **SELECT** landing_outcome, count(landing_outcome) **AS** Count **FROM** SPACEXTABLE **WHERE** Date **BETWEEN** '2010-06-04' **AND** '2017-03-20' **GROUP BY** landing_outcome **ORDER BY** count(landing_outcome) **DESC**

https://github.com/vishpuro/Coursera/blob/cd2e893f0e0e404f3c040b5106176654551d2d75/capstone/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- The first map created was to show all the launch sites. In this map circles were created to show the areas surrounding the launch site as well as a pop-up marker showing the full name of site. We also placed a line denoting the equator to answer the question of whether the sites were in proximity to the equator. The answer was no, all 4 sites were at least 3000 km away from the equator.
- The second map created showed a marker cluster, which contained a marker for each landing outcome, colored red for failure and green for success. Since we are trying to predict outcomes in this project we should show the outcomes for management to see easily.
- The third map created showed proximities of the nearest coast, road, railway and city with a line on the map to the CCAFS SLC-40 launch site. This map was made to show that the launch site was close to the coast, close to road and rail infrastructure allowing for the easy transportation of personnel and machinery as well as the site being located far from cities if something goes wrong.

https://github.com/vishpuro/Coursera/blob/cd2e893f0e0e404f3c040b5106176654551d2d75/capstone/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- The dashboard is split into 2 sections: The first is for pie charts and the second is for a scatter graph. The aim of the first section is to visualize the successes and failures of launches from different sites. The aim of the second section is to visualize the successes and failures of different booster versions.
- There is a dropdown list that allows the data to be visualised as a whole (All sites) or one can drill down into the individual sites. The success rate for sites is not a static value, in fact SpaceX have become increasingly successful over time.
- To allow this fact to be seen we added a date range slider. Now the data for the charts can be filtered to a specific date range.
- We add another slider to filter the graphs on payload mass between 0 and 10,000 kg at intervals of 1,000kg.

Section 1: Sunburst chart, Pie chart and 2 bar charts

- The pie chart for the "All sites" dropdown option is different to the individual sites. For All sites, the pie chart shows the percentages of successes attributable to each launch site. This answers the question: "Which site has the largest number of successes?" This is in contrast to the individual sites where the pie chart shows the percentage of successes and failures for a that site.
- The sunburst chart is used to visualize both the numbers of successes and failures in a single visualization. The power of the sunburst chart is that it is able to show the proportions of launches from each site and also how many of those were successes and failures.
- Bar char 1 shows the percentages of successful launches by site. While the sunburst chart is a powerful and information dense visualization, information should be accessible at a glance. This answers the question: "Which site has the launch success rate?"
- Bar chart 2 shows the percentages of successful launches by booster category. This allows us to see if some booster categories and outperforming others, we will see more on this in section 2. The point of this graph was to answer the question: "which booster version has the highest success rate?".

Section 2: Scatter graph and Histogram

- The scatter graph shows successes and failures (y-axis) as a function of payload (x-axis) broken down by booster category (colour). This graph is quite busy for all sites, which is why bar chart 2 was added.
- The histogram successes and failures as a function of payload. The point of this graph was to answer the question: "what are the most successful and unsuccessful payload ranges?"

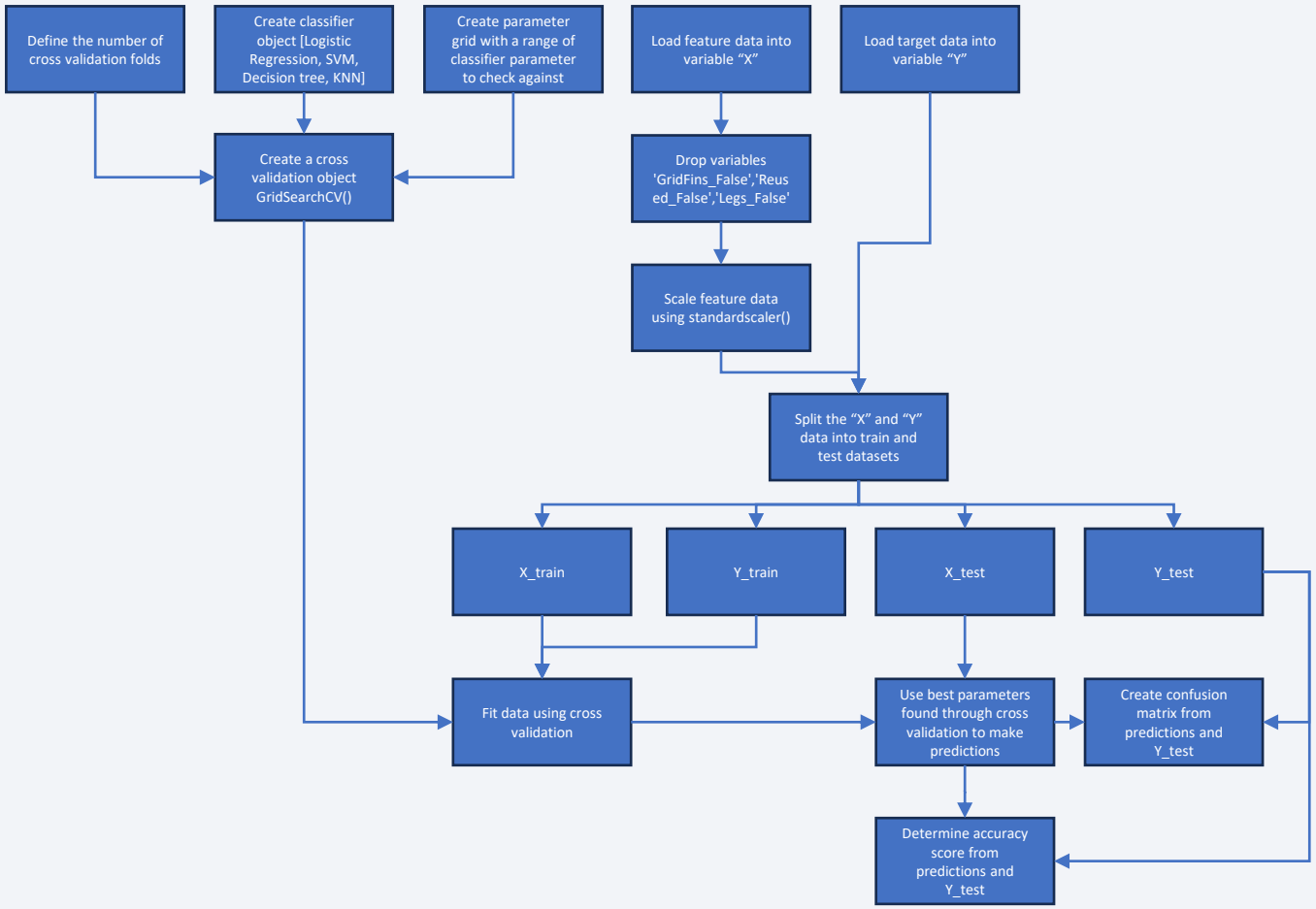
https://github.com/vishpuro/Coursera/blob/cee8b15fda133ebd410352594fdff99a87c81d6b/capstone/dashboard_orig.ipynb

https://github.com/vishpuro/Coursera/blob/cee8b15fda133ebd410352594fdff99a87c81d6b/capstone/dashboard_updated.ipynb

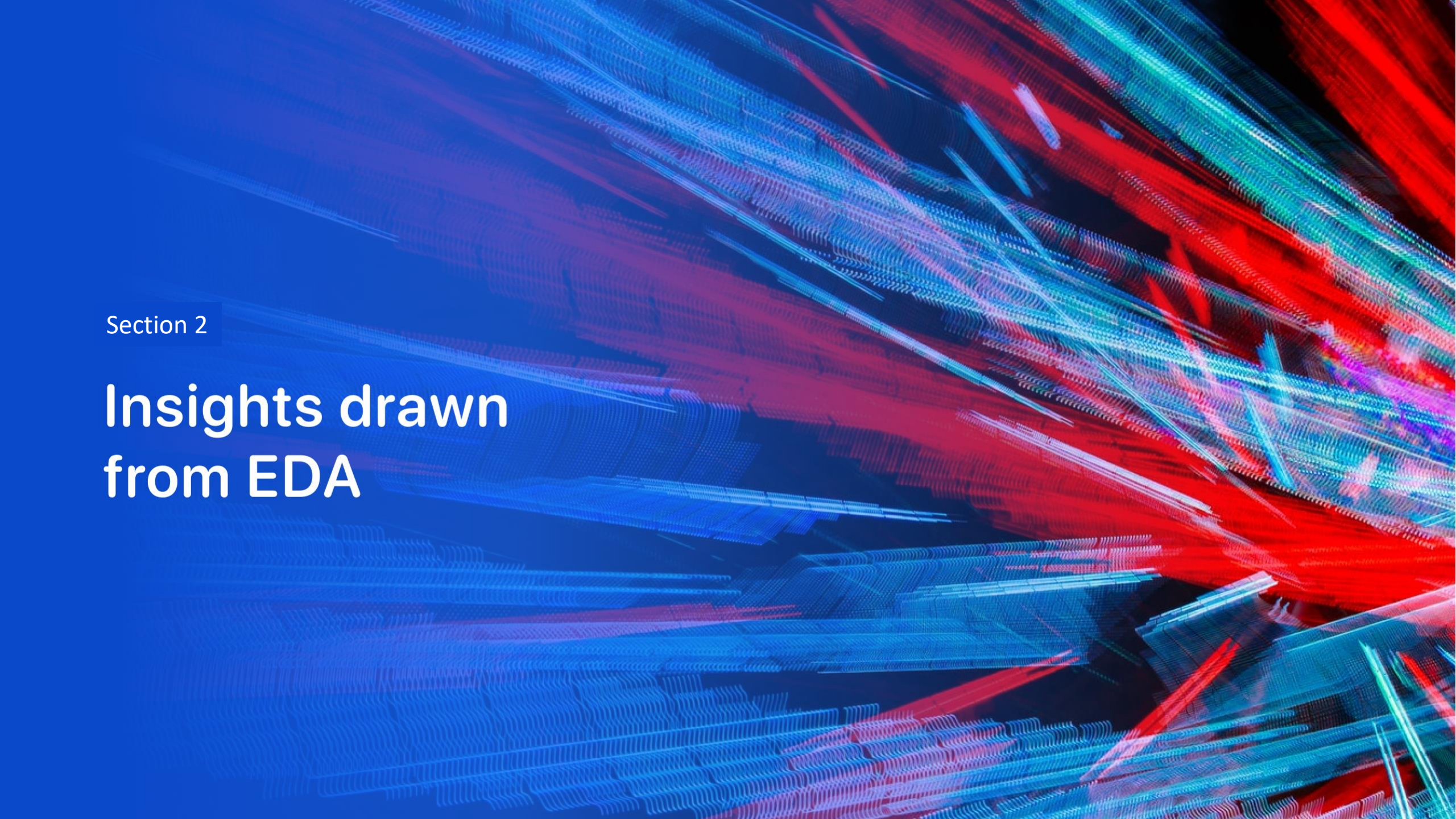
Predictive Analysis (Classification)

- We load the feature data into a variable called X, dropping **'GridFins_False', 'Reused_False' and 'Legs_False'**, since we don't need the True and False variations of the same variables. They will just show the opposite values: [0,0,1] instead of [1,1,0] for example.
- The **X** data is scaled to get the values for the variable in the same range. This is important to algorithms such as KNN that rely on distance metrics (K-**NEAREST**-Neighbours).
- We load target data into a variable called **Y**.
- We split the X and Y data into **training and testing data**. This way we test how good our algorithm building process is.
- We choose to look at 4 algorithms; **Logistic Regression, Support Vector Machines, Decision Tree and K-Nearest Neighbours** and define parameter grids for each of them. These grids contain ranges of potential values to be used by the algorithms. We will be iterating through these to find the best performing values called **Hyperparameters.**
- We fit the data using the algorithm of our choosing, in our case we do this for 4 different algorithms. **GridSearchCV** is a cross validation function from **scikit-learn** that splits the feature data into smaller sets. A loop in created where in every loop one of these sets is used as a test set and the rest are used to train the algorithm that you have selected. This allows us to estimate the accuracy of the algorithm on the Y_test set by testing the algorithm on data it hasn't seen. Each iteration in this loop is tested for each parameter combination in the parameter grid that we defined before. At the end of the loop, the results are averaged to **find the parameter combination that yielded the best overall results**.
- We then make predictions (**y_hat**) using the best parameters found through cross validation. We the made confusion matrices showing True Positives, True Negative, False Positives and False Negatives from the predicted data.

https://github.com/vishpuro/Coursera/blob/cd2e893f0e0e404f3c040b5106176654551d2d75/capstone/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite(1).ipynb



Results

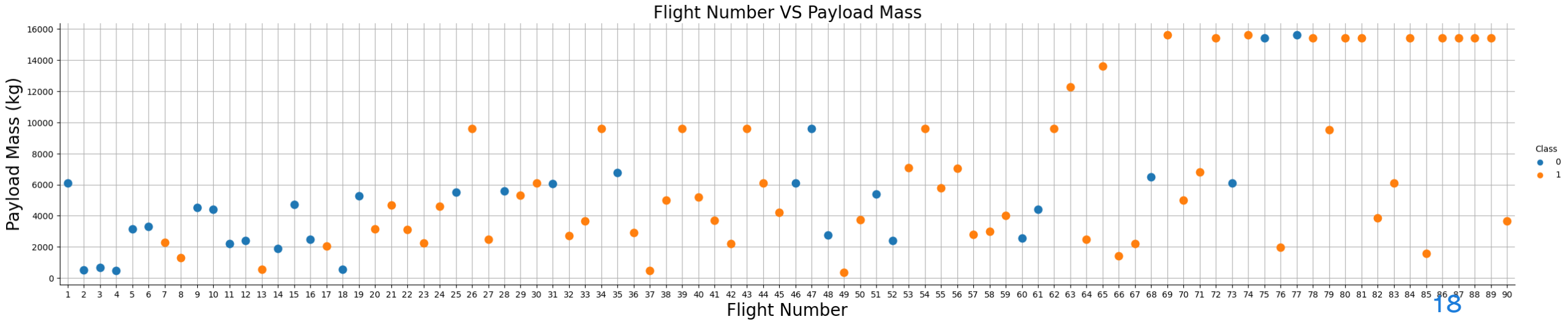

Section 2
Insights drawn from EDA

Flight Number vs. Payload Mass

From this graph, we were able to find **2 strong relationships and 1 weaker relationship:**

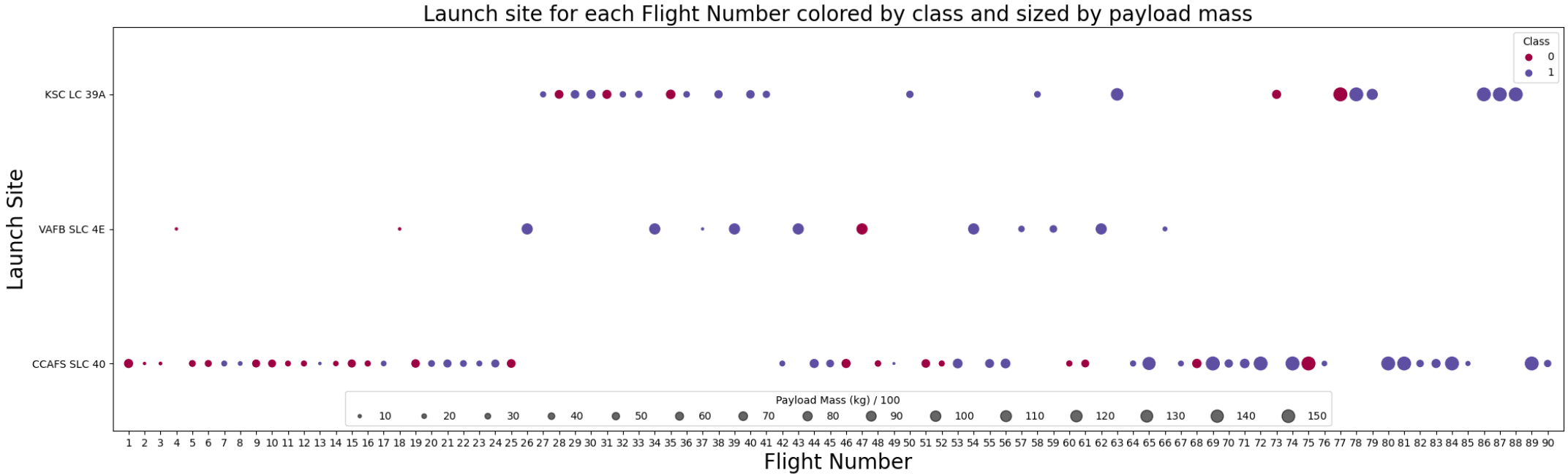
1. The payload mass increased with flight number, indicated by a positive gradient;
2. The number of failures decreased with flight number, indicated by fewer blue markers and more orange markers as the flight number increased;
3. Between flight numbers 19 and 47, the higher payloads showed more failures than lower payloads, indicated by more blue markers higher on the y-axis and more orange marker lower on the y-axis, for similar flight numbers.

https://github.com/vishpuro/Coursera/blob/cd2e893f0e0e404f3c040b5106176654551d2d75/capstone/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb



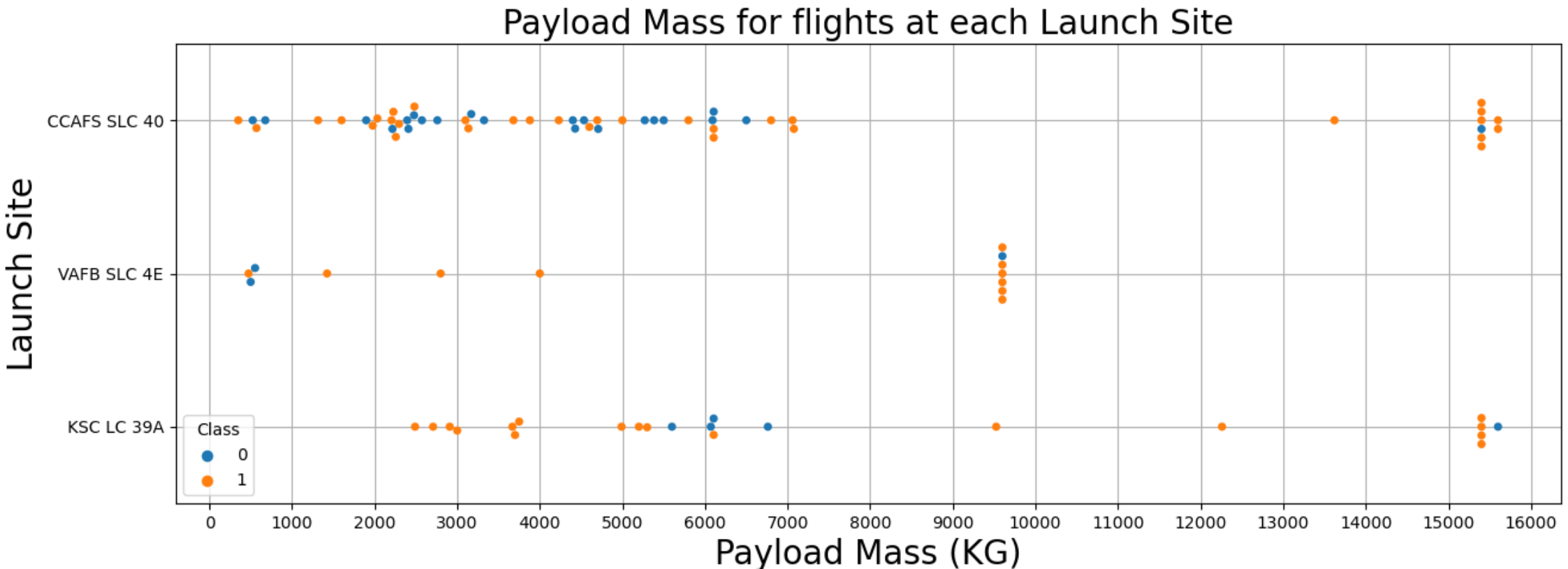
Flight Number vs. Launch Site

- From the graph we see that **CCAFS** is the main launch site, indicated by the number of markers being greater for this site and by the site being used consistently over time. KSC has no flights in the first 25, but consistent use after that time. There are fewer flights from KSC than CCAFS. VAFB has the fewest flights and there have been no flights from there at all in the last 30 launches. VAFB seems to have larger payloads before the other sites, but it is superseded by the other others. There doesn't seem to relationship between launch site and landing success that isn't explained by the success rate increasing with flight number



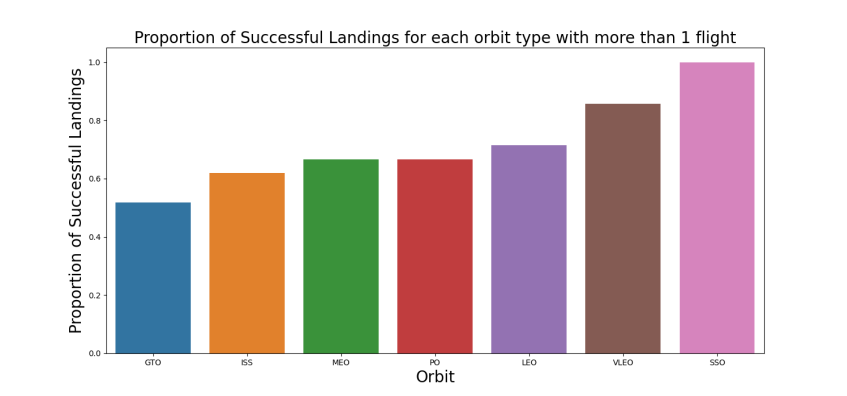
Payload vs. Launch Site

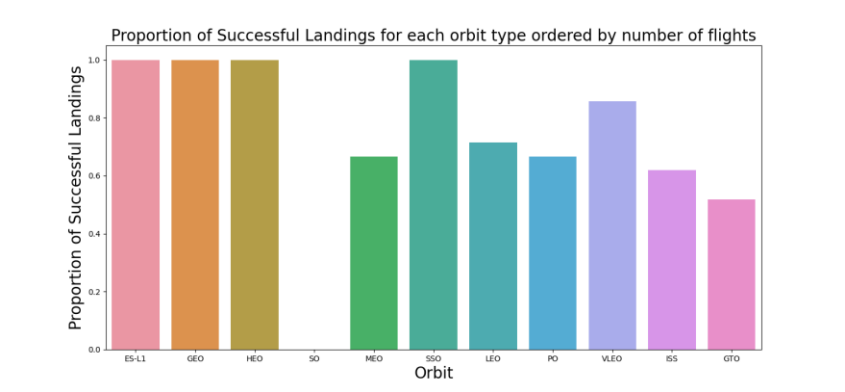
- CCAFS launches payloads of all sizes between 0 and 7000 kg. We will call this the low payload range (LPR). It also launches at a higher payload range (HPR) between 12000 kg and 16000 kg
- KSC launches fewer rockets, however its payload profile is like CCAFS. There are many flight in the LPR and HPR, but only 1 in the middle payload range (MPR) between 7000 kg and 12000 kg.
- VAFB launches the fewest rockets of any site, and its payload profile is different to the other sites. It has many flights in the MPR and a few in the LPR, but none in the HPR.

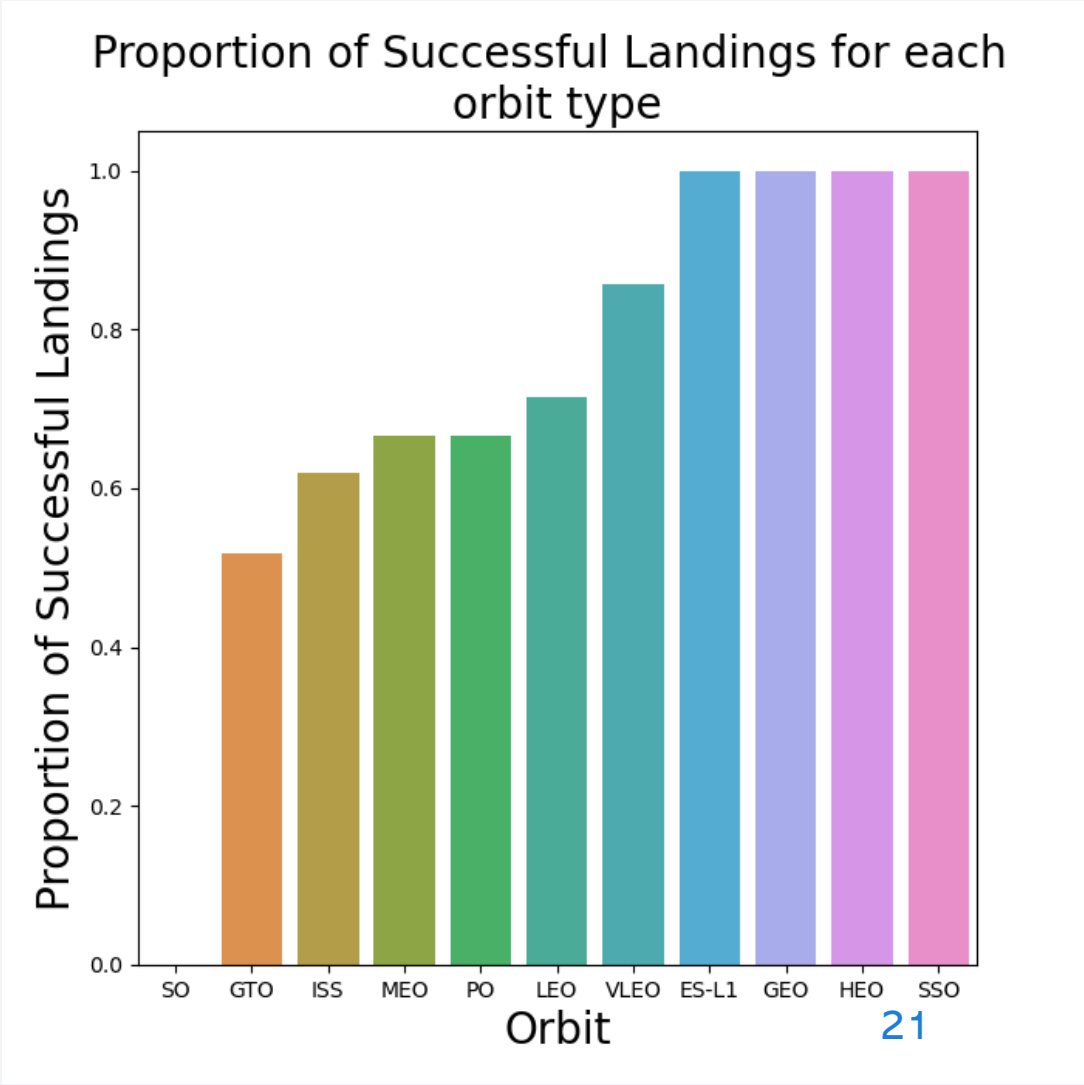


Success Rate vs. Orbit Type

- There are 4 orbit types that have 100% success rates and 1 orbit type has a 0% success rate. This requires greater investigation.
- After checking the number of flights in each orbit type (see next slide), it was found that there are 4 orbit types with only 1 flight. Claiming 100% success rates for them, while technically true, is misleading with so few data points.
- We see that with increasing numbers of flights, the proportion of successes has a negative gradient. However, this graph does not take into account improvements over time (see next slide).

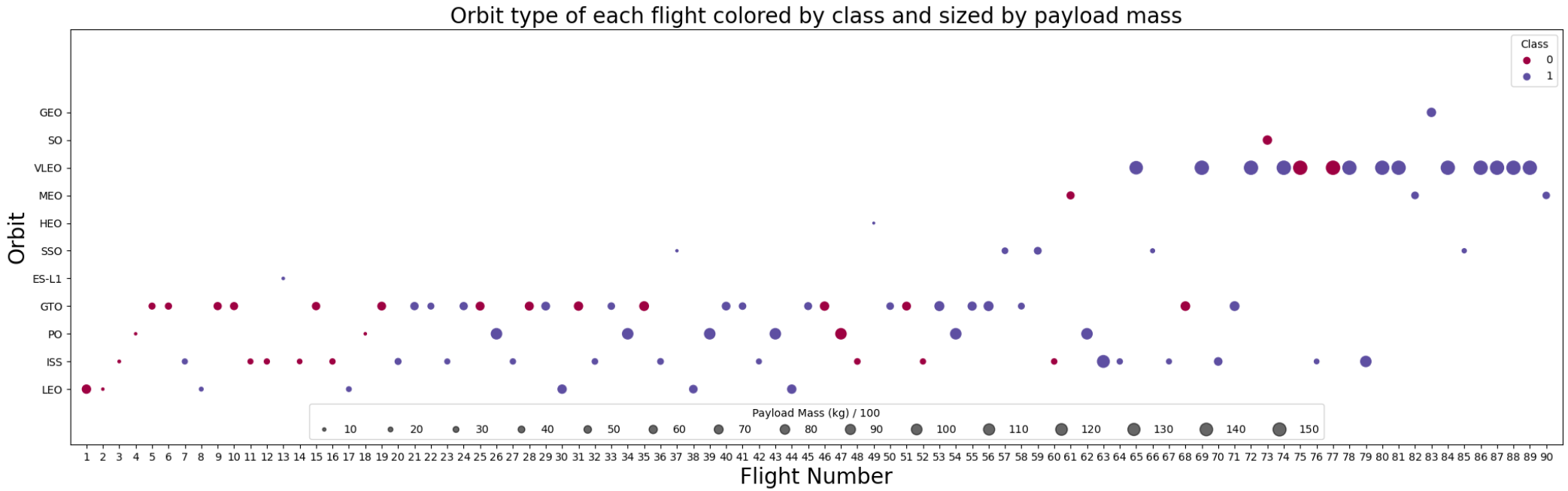






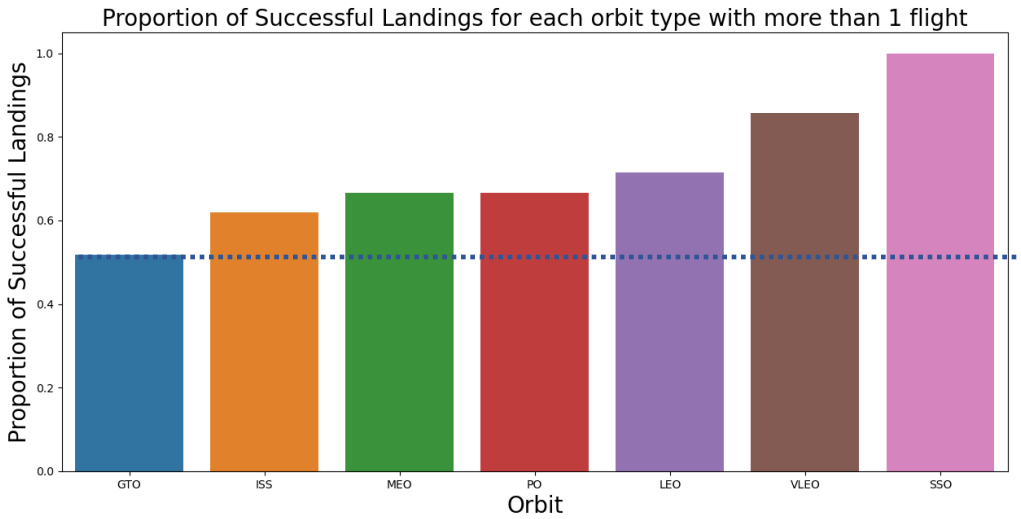
Flight Number vs. Orbit Type

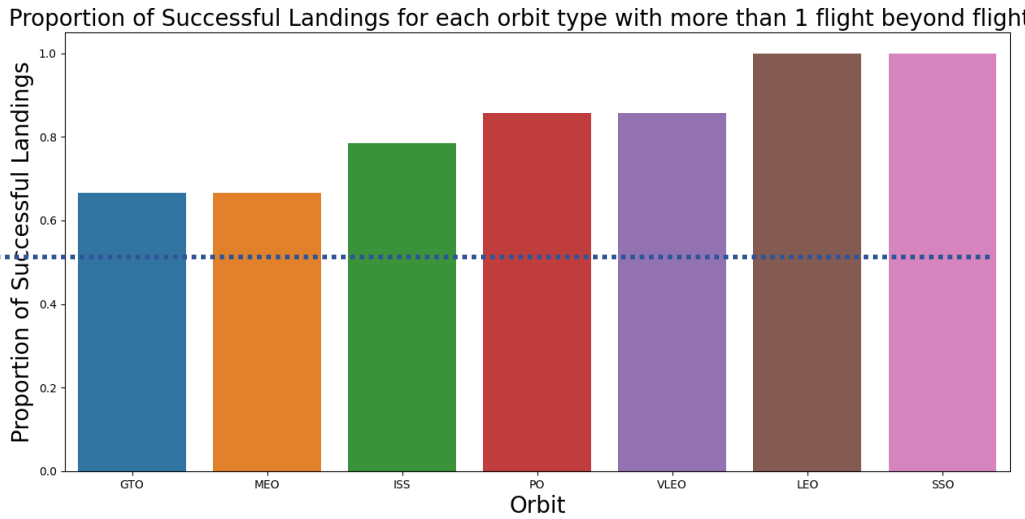
- There are 4 orbit types with only 1 flight (SO,ES-L1,GEO,HEO), of which SO failed, and the rest succeeded. This explains the 3 of the 4 100% success rates and the only 0% success rate we saw in the previous graphs.
- VLEO is the newest of the repeated flights starting at flight 65 and is the only regular flight after flight 80. It also consistently has the highest payloads.
- MEO had 3 flights, the first failed, but he subsequent 2 succeeded.
- SSO has 5 flights and every one of them succeeded.
- GTO is a regular flight starting at flight 5 and ends at flight 72. It has the highest failure rate of any orbit type that wasn't a one-off flight, but 6/13 of the failures for this orbit type occur in the first 20 flights.
- PO is a regular flight starting at flight 4 and ends at flight 62.
- ISS is another regular flight starting at flight 3. The last is at flight 79. The ISS orbit type has, like GTO has a high failure rate. Much like GTO, 5/8 failures occur early on in SpaceX's history: within the first 16 flights
- LEO was the first flight and is a regular flight until flight 45.



Success Rate vs. Orbit Type beyond flight 20

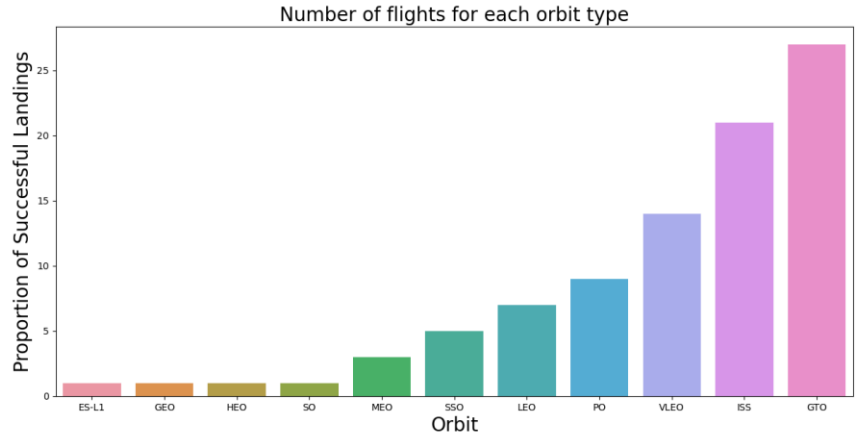
- If we look at the difference in the proportion of successful landings for each orbit type with more that 1 flight including and excluding the first 20 flights, we see that the success rate jumps up markedly for all orbit types.

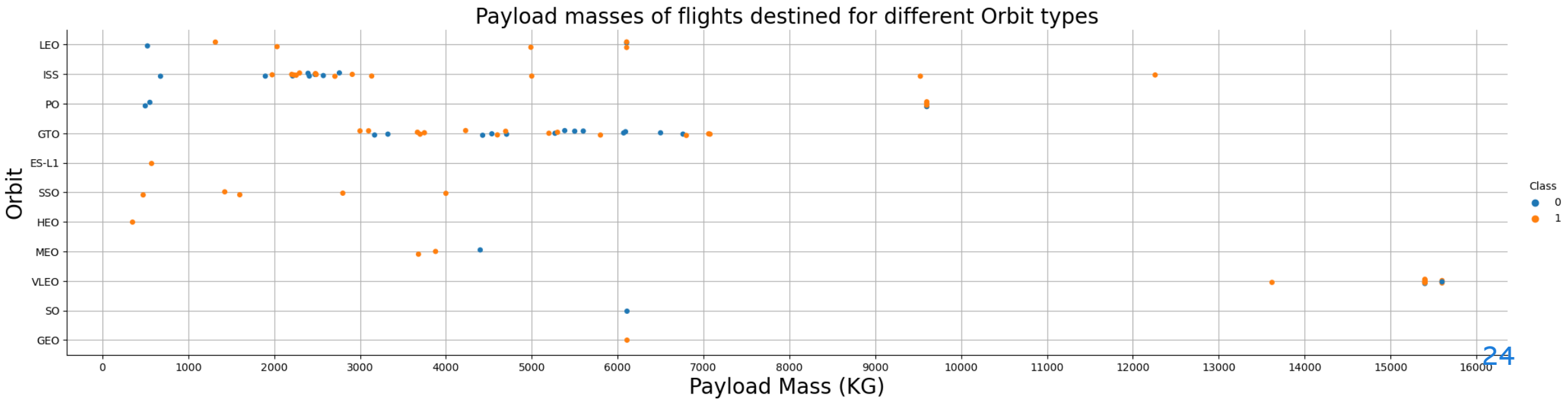




Payload vs. Orbit Type

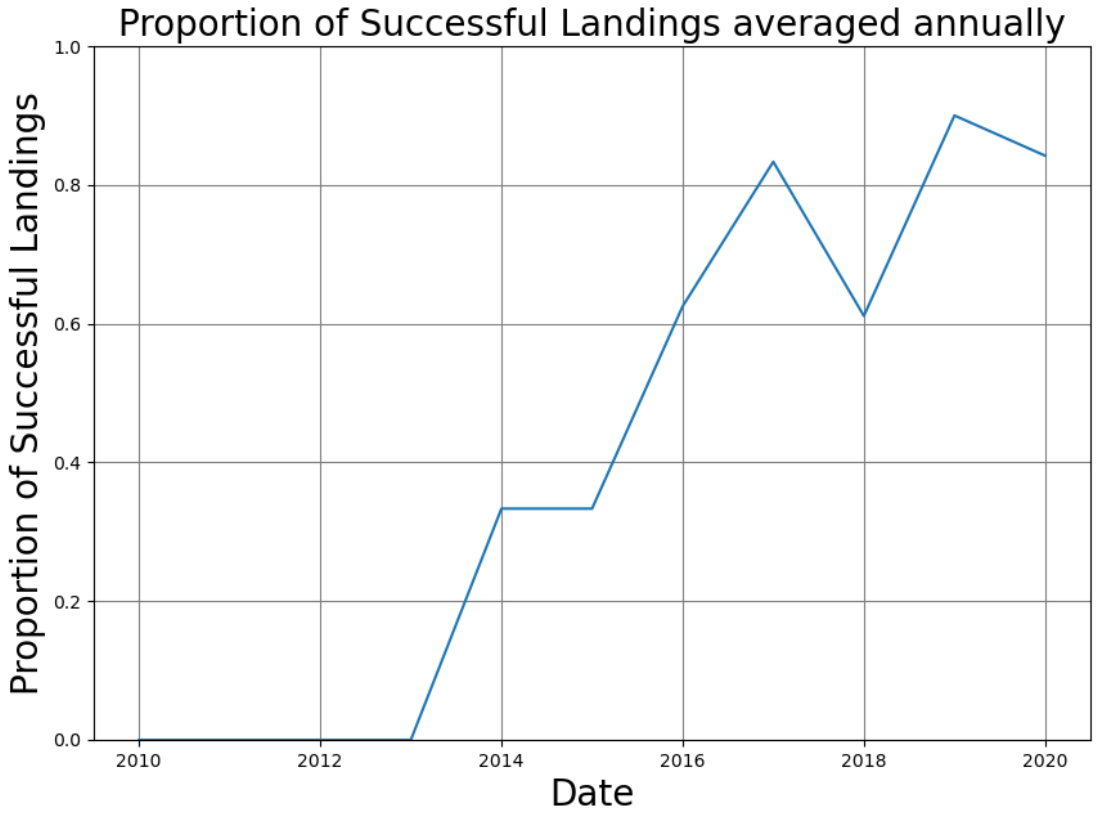
- Flights to GTO have payloads are evenly distributed between 3000 and 7000 kg.
- Flights to the ISS are mostly around 2000 kg to 3000 kg, with a few heavier loads.
- VLEO payloads are tightly clustered around the 15000 to 16000 kg range with 1 data point in the 13000 to 14000 kg range.
- PO payloads are tightly clustered with 2 payloads in the 0 to 1000 kg range and 5 in the 9000 to10000 kg range.
- MEO payloads are between 3000 and 5000 kgs.
- The other orbits have either 1 data point or large ranges. Predicting payload based on their orbits would be difficult.
- ISS, GTO and VLEO form distinctive payload bands allowing prediction of Orbit type from payload.





Launch Success Yearly Trend

- We see that there is a general trend upward trend over time. This is backed up by our graphs of success rate vs orbit type including and excluding the first 20 flights. The year looks like a very useful feature for predicting the success of a flight.



All Launch Site Names

To find the list of SpaceX launch sites we queried all unique **launch sites** from the table. To do this we used the DISTINCT command:

SELECT DISTINCT launch_site

FROM SPACEXTABLE

https://github.com/vishpuro/Coursera/blob/cd2e893f0e0e404f3c040b5106176654551d2d75/capstone/jupyter-labs-eda-sql-coursera_sqllite.ipynb

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

Launch Site Names Begin with 'CCA'

We then wanted to see what kind of data is present in the table, so we queried 5 records with **CCAFS** launch site To do this we used the WHERE LIKE commands to filter the results and the LIMIT command to select only 5 records.

```
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

Total Payload Mass

We wanted to see the total **payload mass** carried for customers, for example **NASA (CRS)**. To do this we used the SUM command with the WHERE LIKE commands to filter the results. The AS command was used to rename the results column since sum(PAYLOAD_MASS__KG_) is quite long.

```
SELECT customer, sum(PAYLOAD_MASS__KG_) AS Total_payload

FROM SPACEXTABLE

WHERE customer LIKE 'NASA (CRS)%'
```

Customer	Total_payload
NASA (CRS)	48213

Average Payload Mass by F9 v1.1

We wanted to see the average **payload mass** carried by different **booster versions**, for example **F9 v1.1**. To do this we used the AVG function to get the mean with the WHERE command to filter the results.

```
SELECT booster_version, AVG(PAYLOAD_MASS__KG_) AS Average_Payload
FROM SPACEXTABLE
WHERE booster_version = 'F9 v1.1’
```

Booster_Version	Average_Payload
F9 v1.1	2928.4

First Successful Ground Landing Date

We wanted to see when the first successful first stage landing on a **ground pad**. To do this we used the MIN command to get the lowest value of **DATE** with the WHERE LIKE commands to filter the results.

```
SELECT min(Date) AS First_Success

FROM SPACEXTABLE

WHERE landing_outcome LIKE 'success (ground pad)'
```

First_Success
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

We wanted to see which **boosters** successfully landed on a **drone ship** with **payload masses** between 4000 and 6000 kg. To do this we used the WHERE LIKE commands to filter the results with an AND command to use multiple filter conditions. The second condition used the BETWEEN command which allows filtering between two values.

```
SELECT booster_version , PAYLOAD_MASS__KG_ AS Payload_Mass

FROM SPACEXTABLE

WHERE landing_outcome LIKE 'success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000
```

Booster_Version	Payload_Mass
F9 FT B1022	4696
F9 FT B1026	4600
F9 FT B1021.2	5300
F9 FT B1031.2	5200

Total Number of Successful and Failure Mission Outcomes

We wanted to see how many **mission** successes and failures there were. This is different to the landings that we were looking at in other sections. To do this we used two subqueries with the COUNT command with the WHERE LIKE commands to filter the results.

```
SELECT
        (SELECT count(mission_outcome)
        FROM SPACEXTABLE
        WHERE mission_outcome LIKE "success%")
 AS successes,
        (SELECT count(mission_outcome)
        FROM SPACEXTABLE
        WHERE mission_outcome like "failure%")
AS failures
FROM SPACEXTABLE LIMIT 1
```

successes	failures
100	1

Boosters Carried Maximum Payload

We wanted to see which **booster versions** carried the maximum **payload mass**. To do this we used the DISTINCT command to get unique booster versions and the WHERE command to filter the results. The WHERE equation is found using a subquery with the MAX command to find the largest payload mass value. We then ordered the results by booster version name.

```
SELECT DISTINCT booster_version, PAYLOAD_MASS__KG_ AS Payload_Mass
FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (    SELECT max(PAYLOAD_MASS__KG_)
                                                FROM SPACEXTABLE
                                                )
ORDER BY booster_version
```

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

2015 Launch Records

We wanted to see the **dates** for **drone ship landing failures** in **2015**, which **boosters** they had, and which **launch site** they came from. To do this we used the SUBSTR command since SQLite is not able to handle month names. SUBSTR(COLUMN,6,2) gets 2 characters starting at character 6. Since the DATE is in the format YYYY-MM-DD, this gives us the month. We also use the WHERE command to filter the results with an AND command to use 2 conditions.

```
SELECT substr(Date, 6,2) AS Month, landing_outcome, booster_version, launch_site, substr(Date,0,5) AS Year

FROM SPACEXTABLE

WHERE substr(Date,0,5)='2015' AND landing_outcome = 'Failure (drone ship)'
```

Month	Landing_Outcome	Booster_Version	Launch_Site	Year
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015

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

We wanted to see how many of each landing **outcome** there are and rank them for **dates** between 04/06/2012 and 20/03/2017 in **descending order**. To do this we used the COUNT command and the WHERE and BETWEEN commands to filter the results to dates between 04/06/2012 and 20/03/2017. We then use the ORDER command to sort the results and the DESC command to sort in descending order.

```
SELECT landing_outcome, count(landing_outcome) AS Count

FROM SPACEXTABLE

WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'

GROUP BY landing_outcome

ORDER BY count(landing_outcome) DESC
```

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

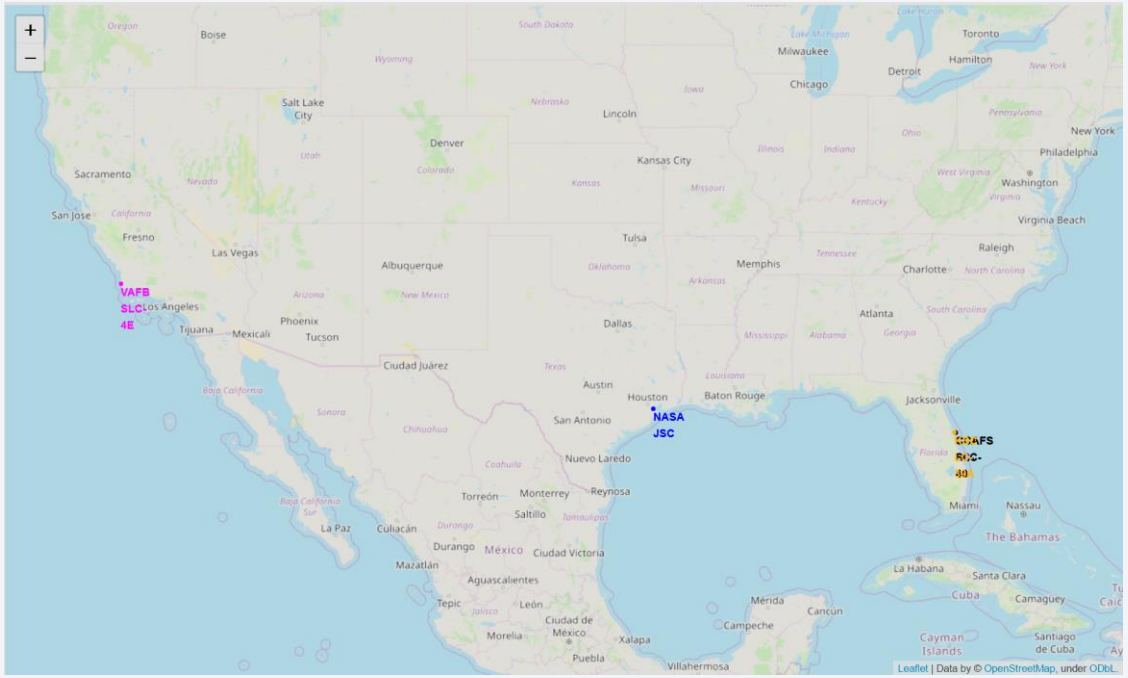
Launch Site Locations

The map shows the locations of 3 launch sites. CCAFS SLC-40 was previously known as CCAFS LC-40 so those two (purple and black) are in fact the same site. The map also shows the location of NASA JSC. We can answer 2 questions from this:

1. Are all launch sites in very close proximity to the coast? Yes, there are 2 sites on the east coast and 1 on the west coast. Nasa JSC is also on the coast.

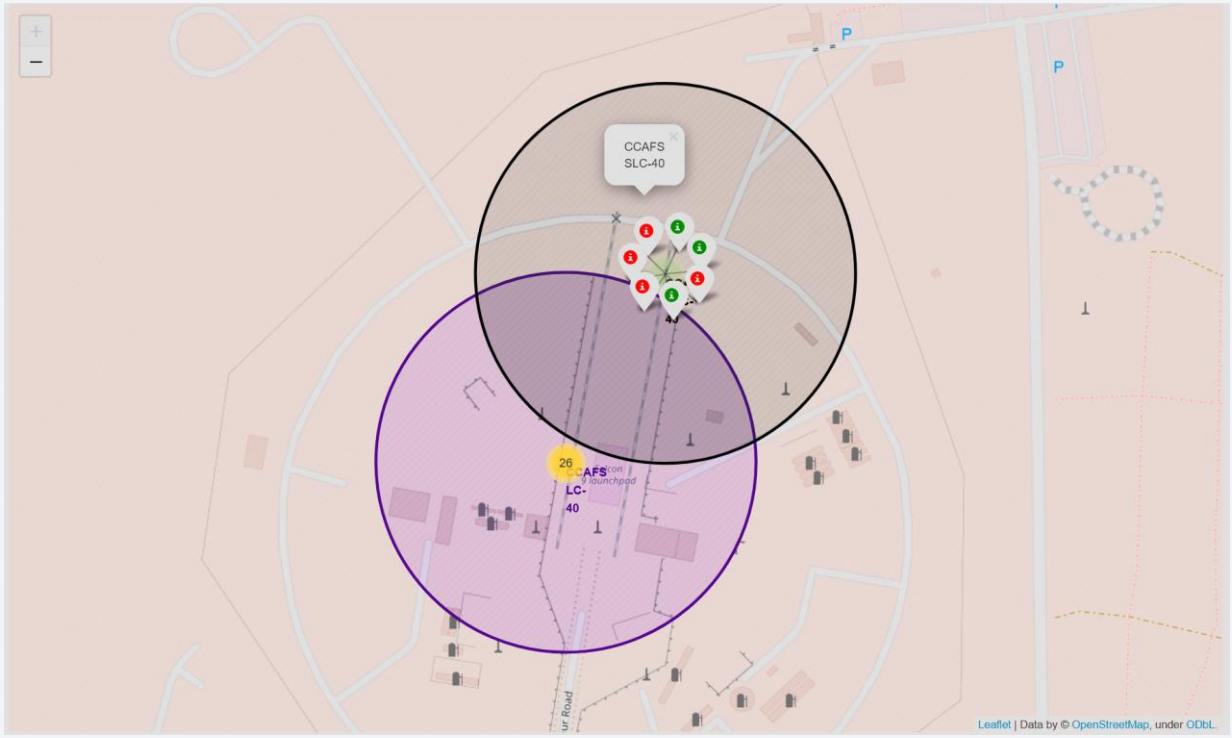
2. Are all launch sites in proximity to the Equator line? No, they are quite far away:

The distance from CCAFS LC-40 to the equator is: 3176.98 km
The distance from CCAFS SLC-40 to the equator is: 3177.08 km
The distance from KSC LC-39A to the equator is: 3178.20 km
The distance from VAFB SLC-4E to the equator is: 3852.20 km



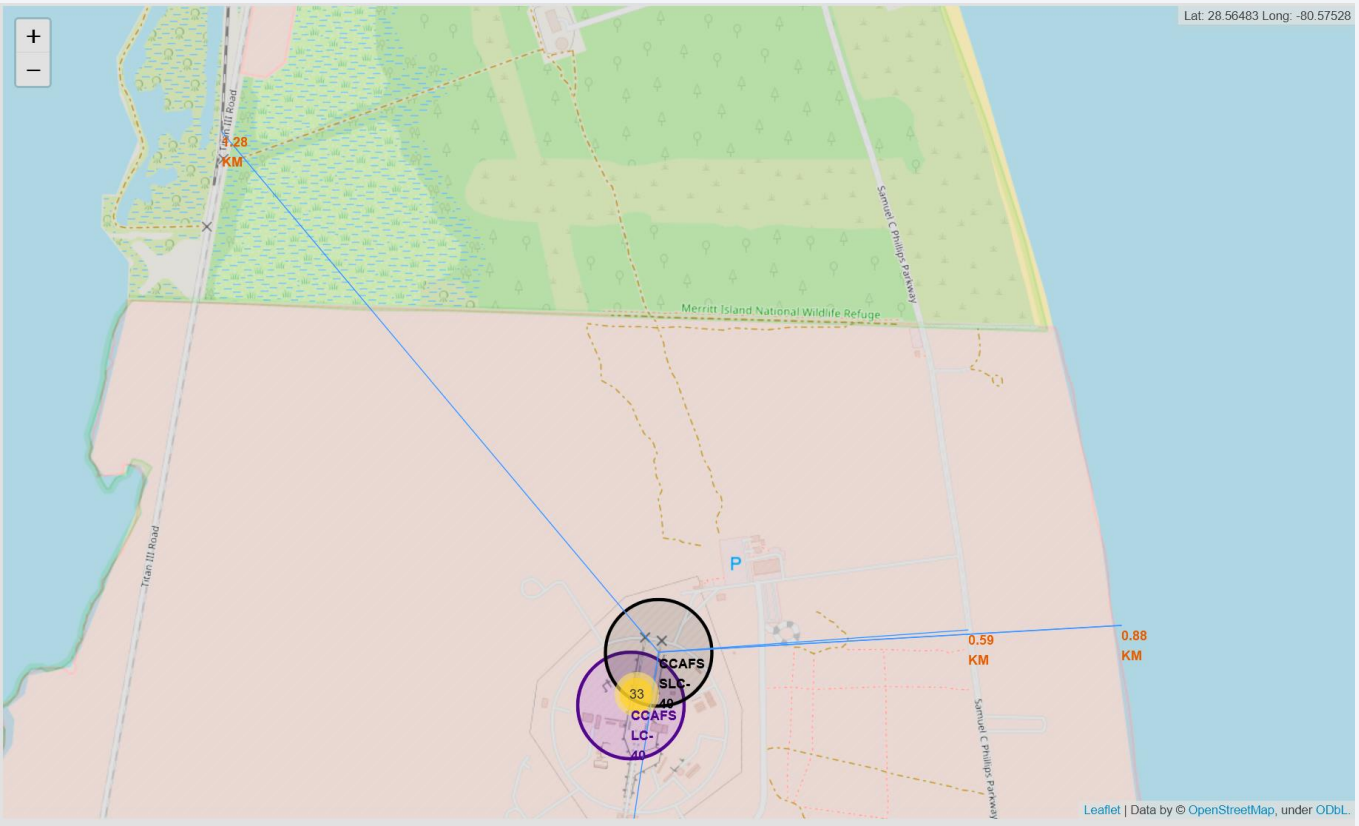
Launch outcomes at sites

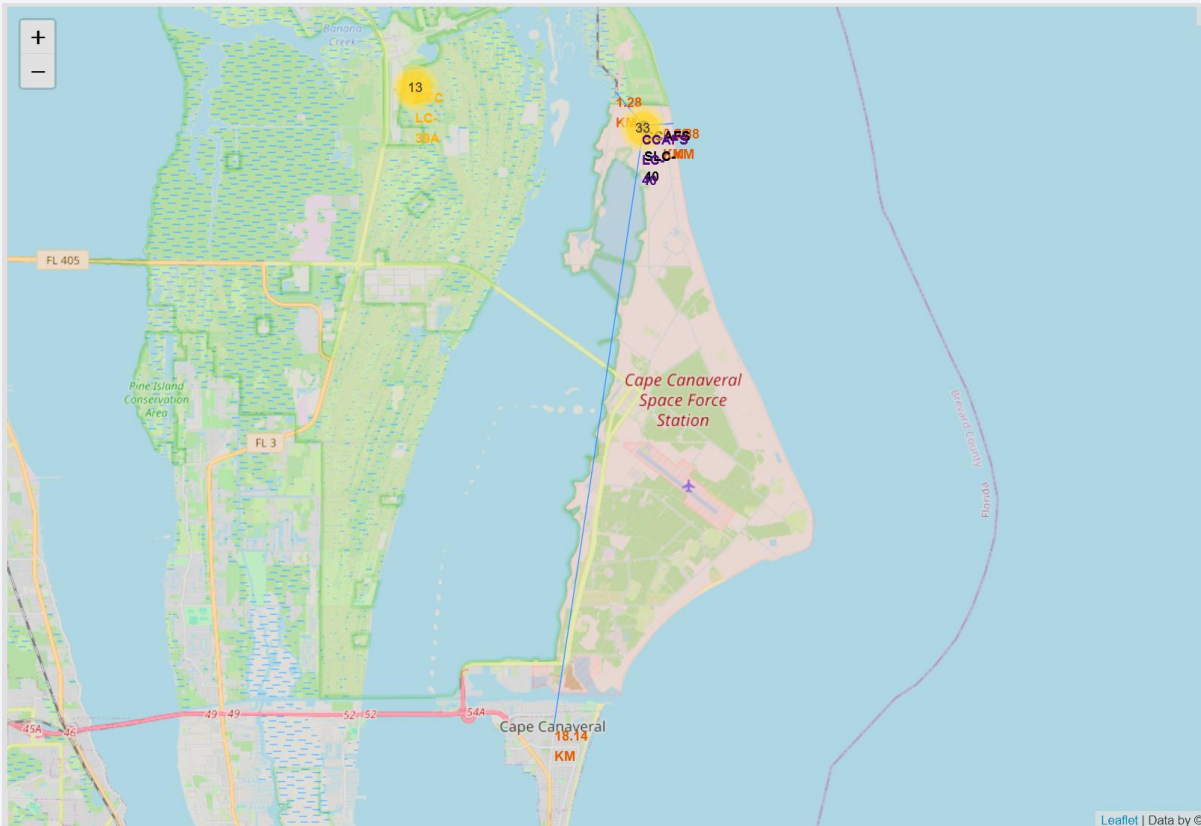
- Here we see CCAFS SLC-40 with coloured markers (green for success and red for failure) allowing us to see at a glance the outcomes of launches.



Infrastructure proximity

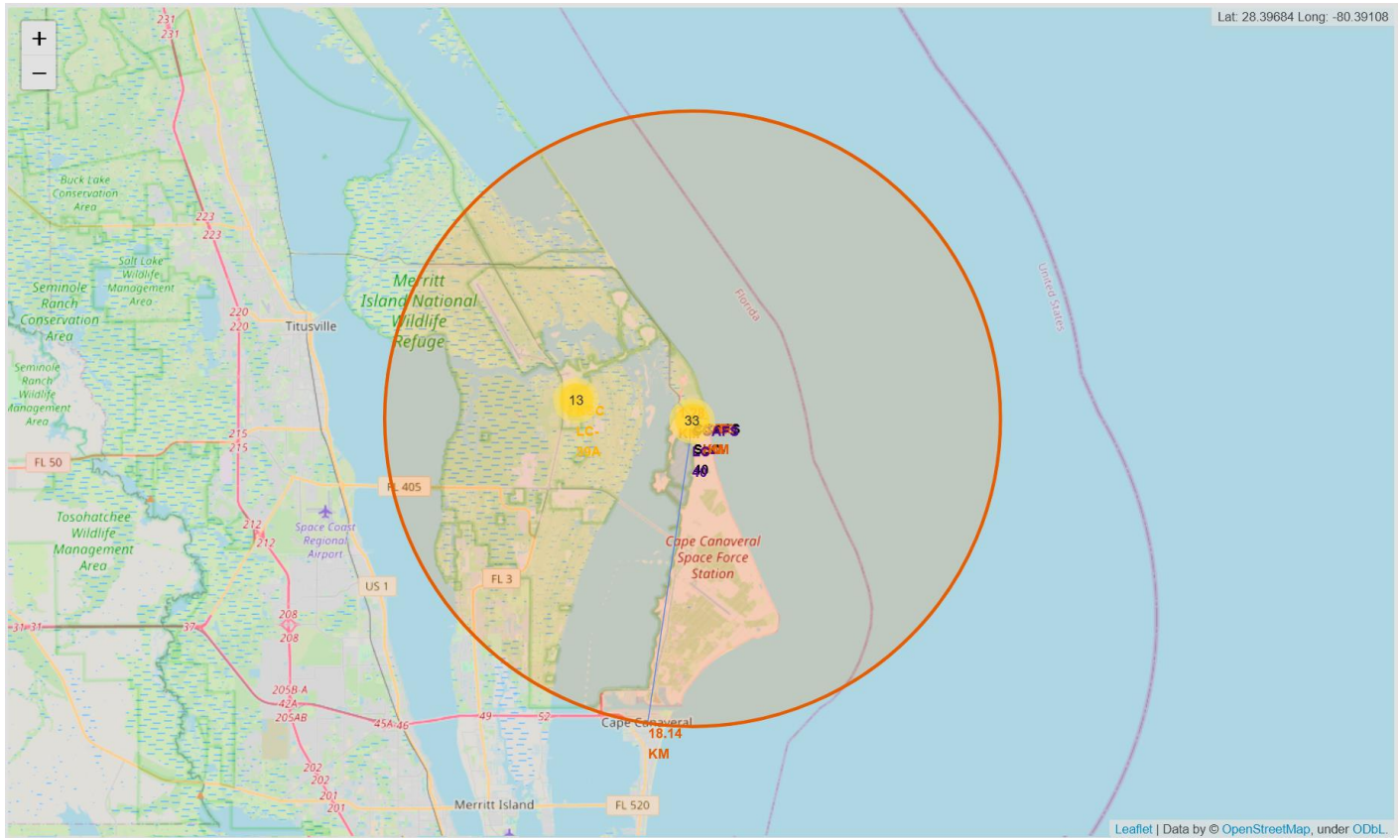
- This map was made to show that the launch site was close to the coast (0.88 km), close to road and rail infrastructure (0.59 km and 4.28km) allowing for the easy transportation of personnel and machinery as well as the site being located far from cities (18.14 km) if something goes wrong.

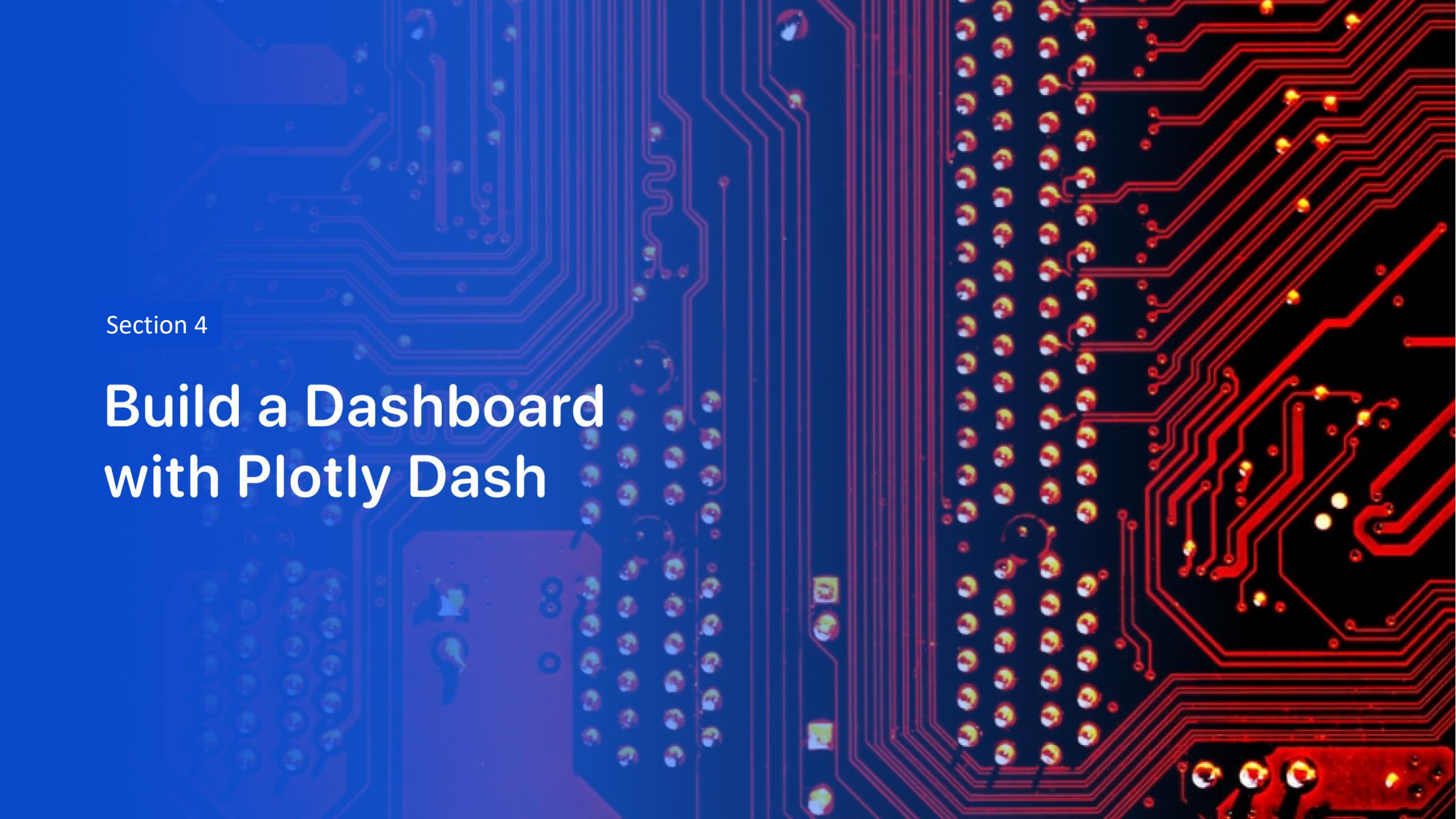




Is Cape Canaveral the closest city to CCAFS?

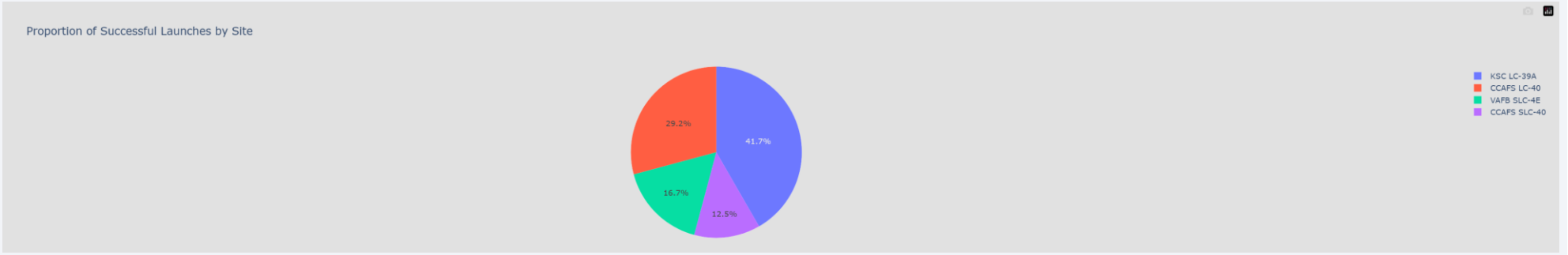
- Yes. A circle to show that Titusville is farther away than Cape Canaveral.

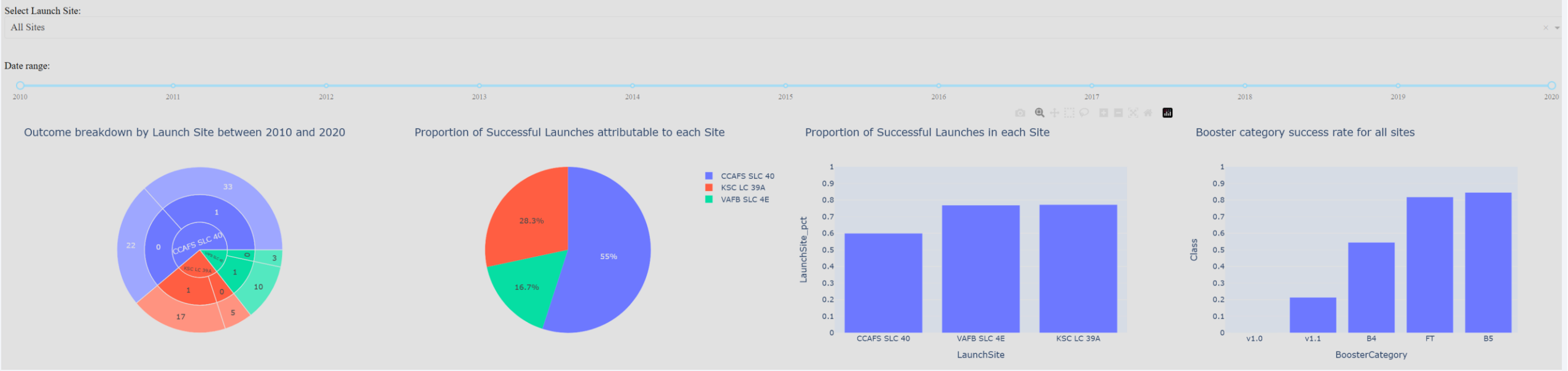



Section 4
Build a Dashboard with Plotly Dash

Launch Success (All Sites)

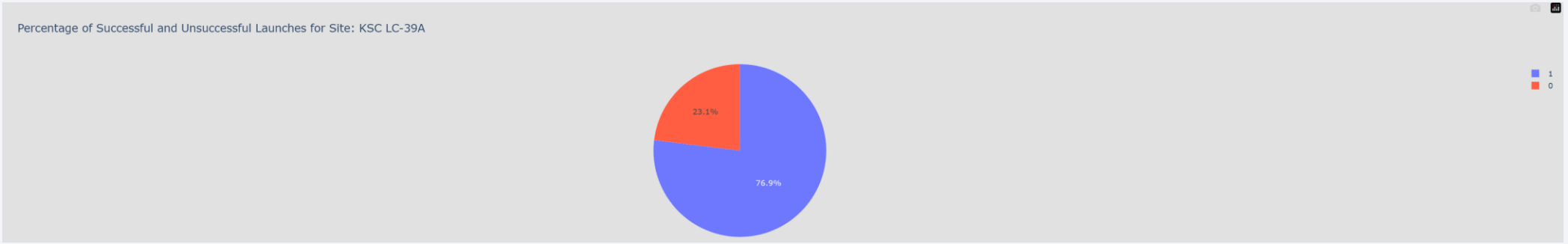
Shown below are the original pie chart as well as an updated dashboard. The original dashboard had data up to June 2018; this has been extended to 2020 in the update, which also added a slider to change the time frame of interest. The updated dashboard also corrects the CCAFS name change so that all CCAFS data is attributed to CCAFS SLC-40. We can see that a sunburst chart gives information about the successes (1) and failures (0) at each site, while the pie chart tells us what proportion of the total number of successes are attributable to each site. A bar chart shows the successes rate at each site as another bar chart shows the success rates for different booster types. These charts give us addition information such as: despite VAFB having the smaller proportion of successes between 2010 and 2020 (pie chart) than CCAFS; VAFB has a higher success rate than CCAFS (bar chart 1)

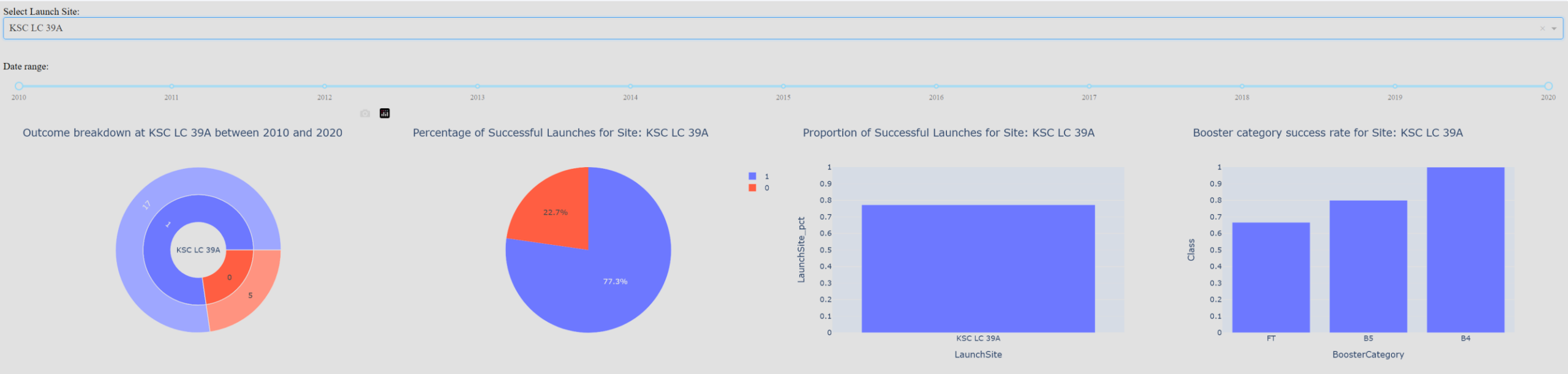




Best success rate: Launch Site KSC

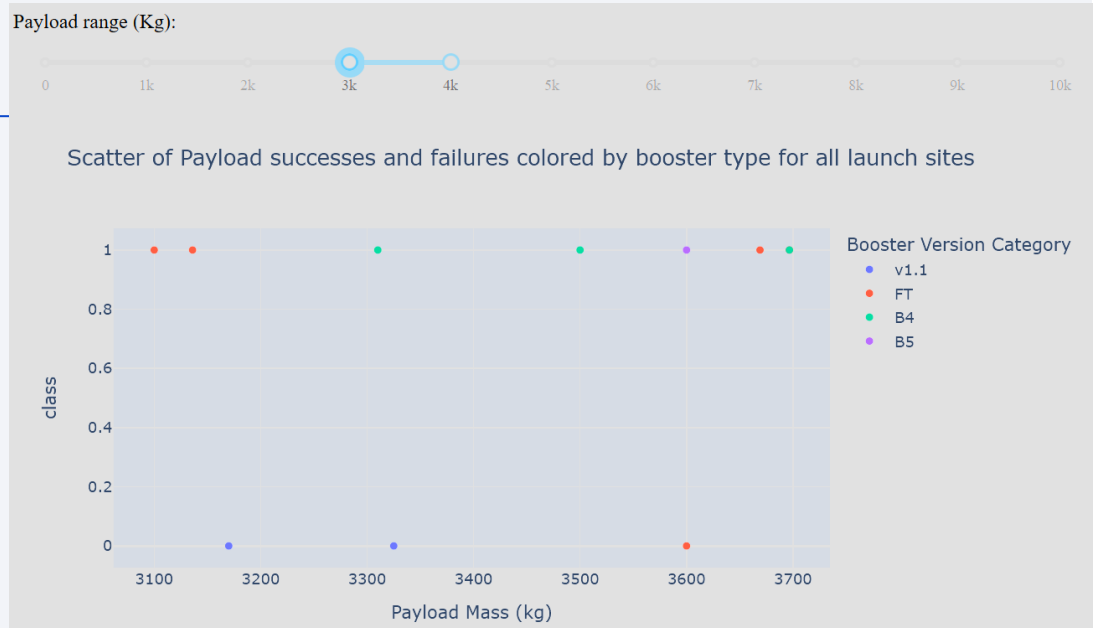
- In the original dashboard we had to go through the sites individually to find the launch site with the higher launch success ratio, but we could see this information on the ‘All Sites’ page on the updated version. If we drill down into the KSC page we see that between 2010 and 2020 there were 17 successes and 5 failures, giving a success rate of 77.3% For the data in the original which stopped in June 2018, site KSC had 1 fewer success, CCAFS has 3 fewer successes and VAFB had 3 fewer successes making KSC once again the site with the best success rate.

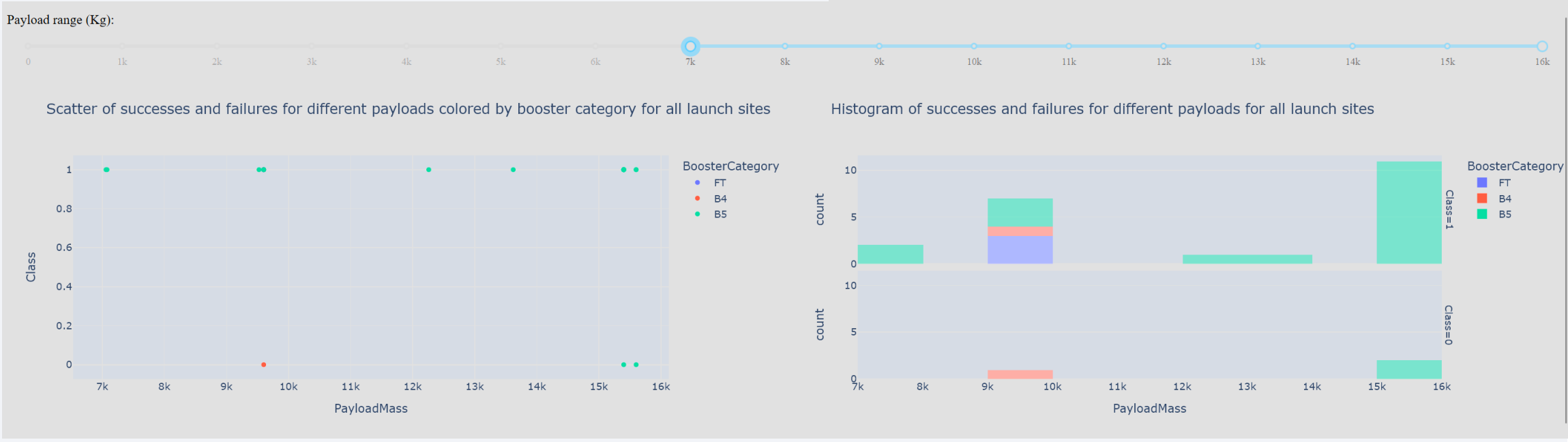




Payload Vs Outcome

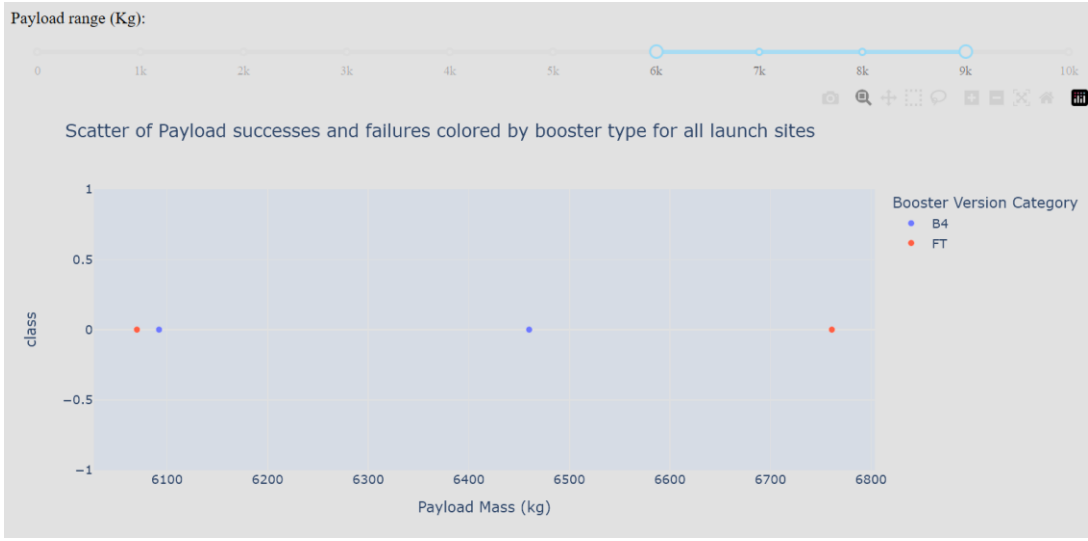
- The original dashboard had just a scatter graph. The updated dashboard has a histogram for the different outcomes. We can see that in the update that the histogram tells us that there are many points with the same payload mass. We cannot see this information in the original dashboard.
- The extended timeframe for the updated dashboard also changes the answers to some questions, we see that the highest payload success rate occurs in the 3000 to 4000 kg range, whereas in the update the 7000 to 16000 kg range is better. This range did not exist in the old dashboard, which highlights SpaceX's improvement over time.

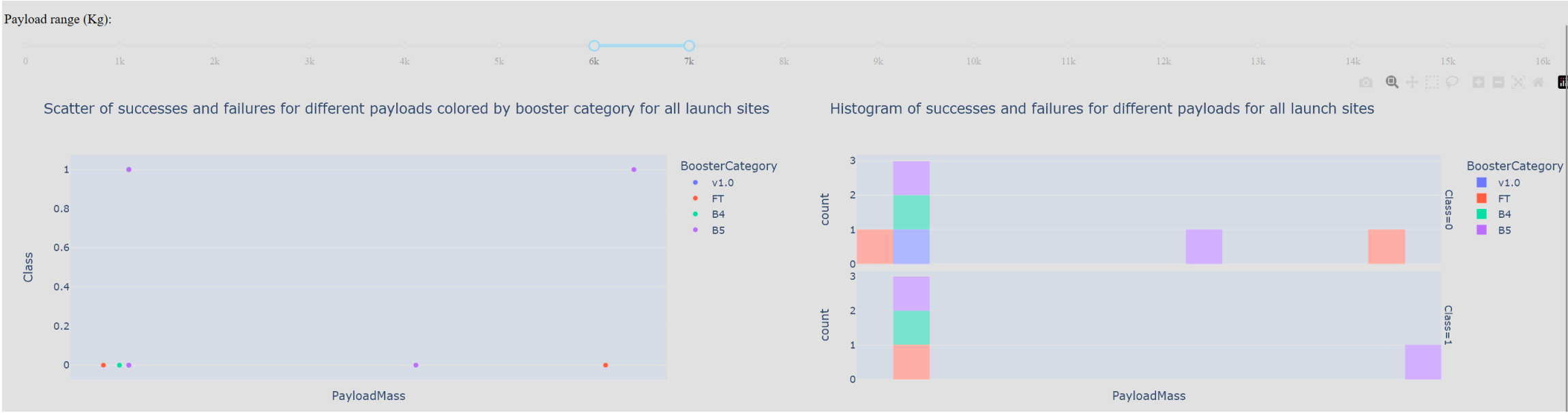




Payload Vs Outcome 2

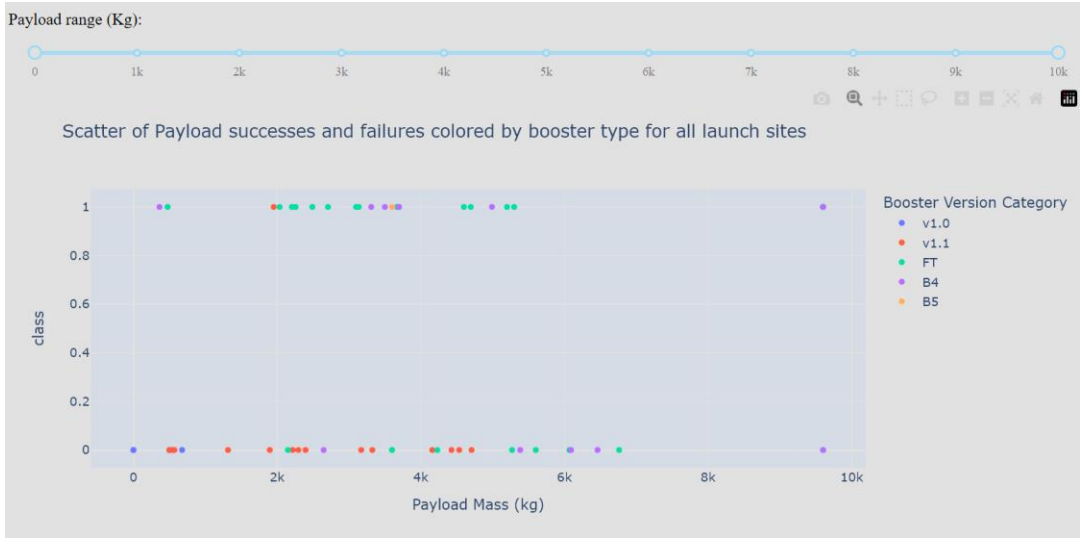
- The old dashboard shows us that the lowest payload success rate occurs in the 6000 to 7000 kg range, where there are zero successes. In the update the 6000 to 7000 kg range has the lowest success rate. We also see that the scatter graph did not show us 2 successes that were hidden behind another scatter point.

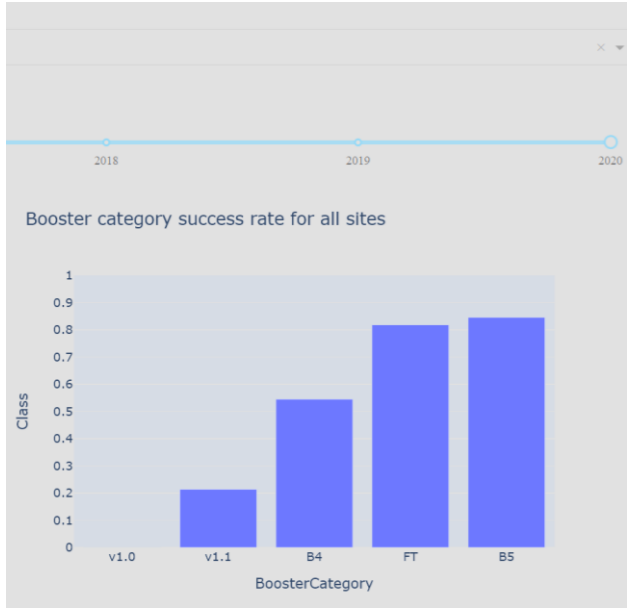




Payload Vs Outcome 3

- Another question that we could look at is: Which booster has the best success rate?
- We see that in the old dashboard, FT is the best booster. We have to count the markers to determine this. Technically B5 is the best, but there is only one data point.
- In the updated dashboard, we see that B5 becomes the best booster, with FT close behind. We don't need to count anything, because we have a bar chart to tell us at a glance. If we wanted to count, we would be better off counting on the histograms, because they allow us to see data points hidden behind others on the scatter graph.



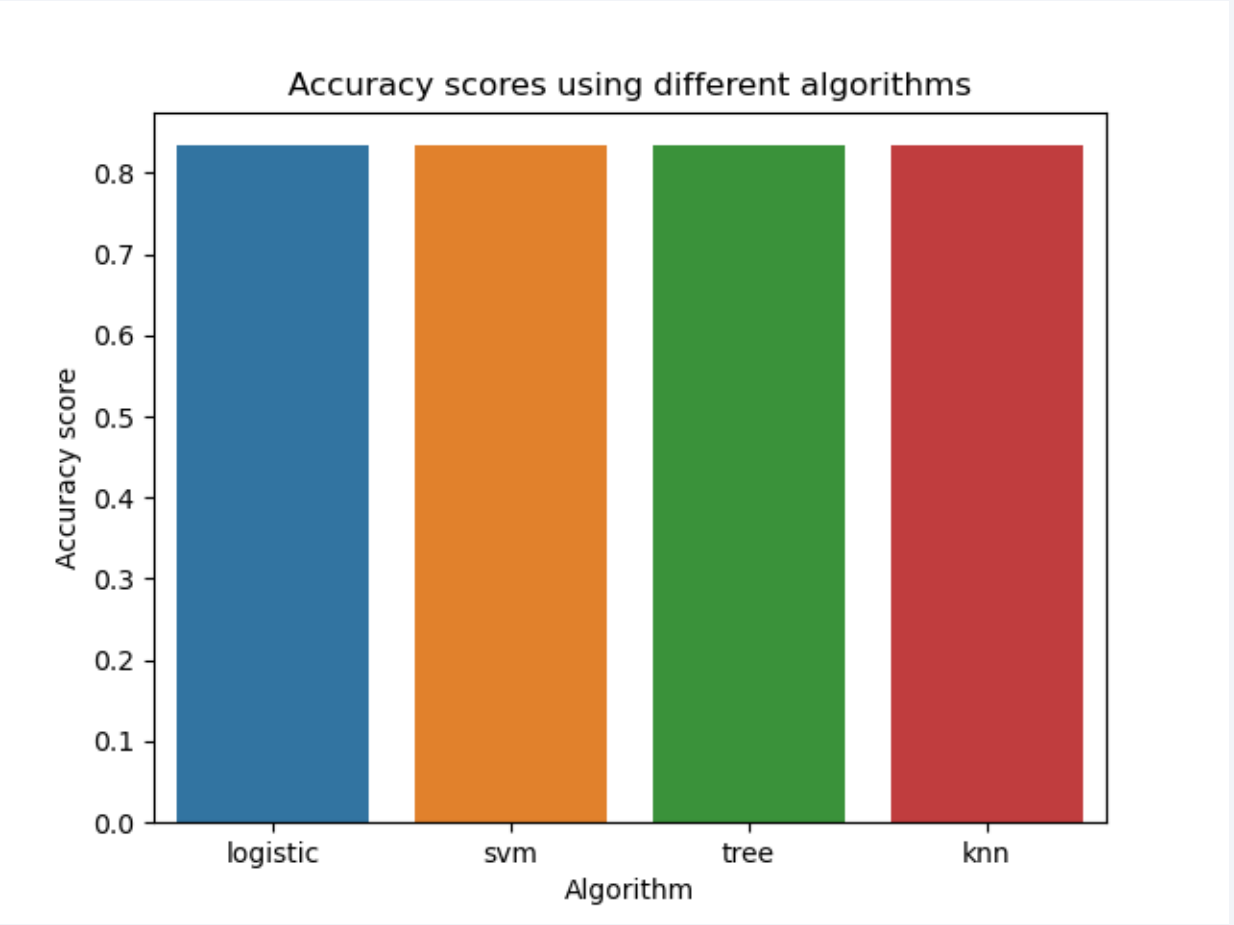


Section 5

Predictive Analysis (Classification)

Classification Accuracy

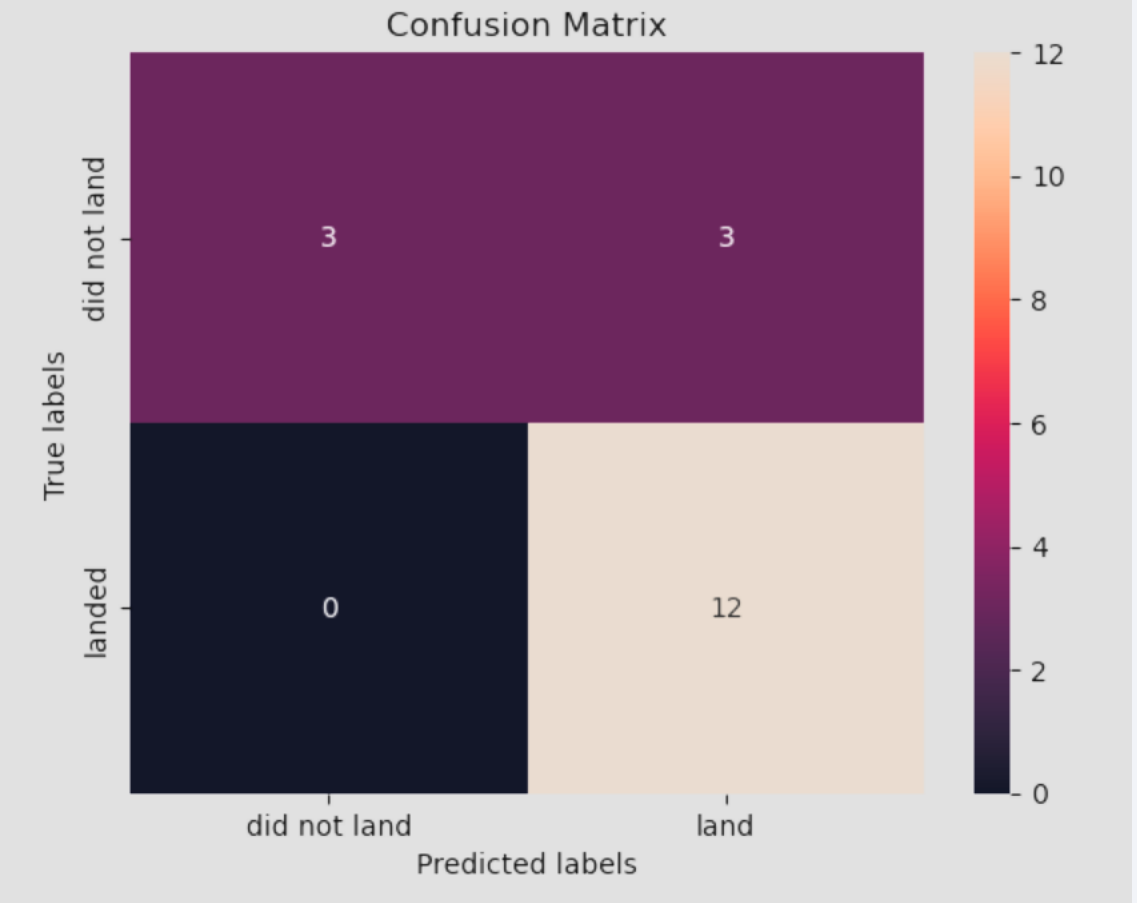
- We can see from the bar chart, Logistic regression, Support Vector Machines, Decision tree classifier and K Nearest Neighbours all yield 83% accuracy.



https://github.com/vishpuro/Coursera/blob/cd2e893f0e0e404f3c040b5106176654551d2d75/capstone/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite(1).ipynb

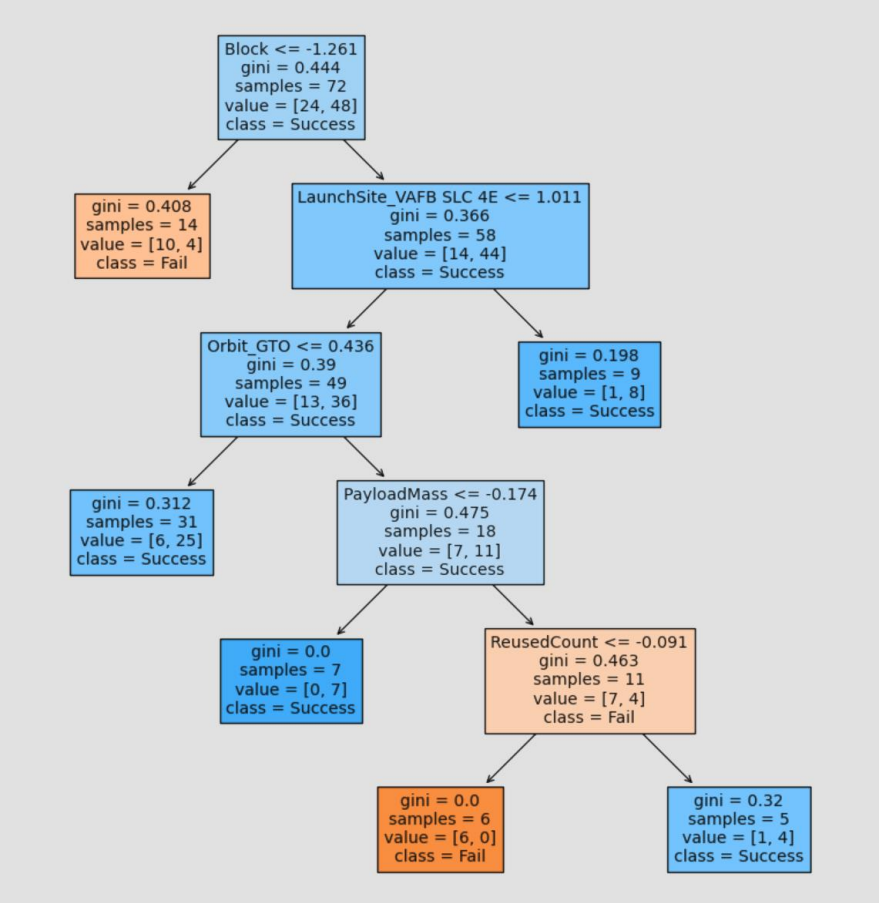
Confusion Matrix

- All 4 models produced the same confusion matrix. We see that:
- True Positives are 12, False positives are 3, True Negatives are 3 and False negatives are 0
- This means that we are “overestimating” the number of successful landings. As Space Y, if we had some idea of how the false positives and false negatives affect us, we could alter the probability thresholds using algorithm.predict_proba() instead of algorithm.predict() control the number of false positives and false negatives. As it stands overestimating the number of Space X success may be useful to our company as we would strive to outdo Space X. It also means that we could focus bidding for contracts where we are sure that Space X will fail.



Decision Tree

- We see that the important decisions are the block type, launch site, orbit type, payload mass and reused count. This makes sense as the block type tells us about the booster version. We saw from our plotly dashboard that the launch site VAFB had a high success rate, but was only used for mid range payloads. We saw in our EDA section that different Orbit types had different payload ranges, so it makes sense that the next two decisions are about the GTO orbit type and payload. Finally, we see that reused count is a good decision. This shows that first stage rockets that survive are more likely to survive again.



Conclusions

- We can predict whether Space X's first stage will land successfully with an accuracy of 83%. The main issue we have with our predictions is false positives. If we had additional information about cost of false positives and false negatives, we could use ROC AUC to change the probability threshold to, for example exchange false positives for false negatives.
- Space X uses 3 launch sites. Two on the east coast and one on the west coast. The site CCAFS SLC-40 is the closest of the 3 to the equator at a distance of 3,177.08 km. All 3 sites are located on the coast. CCAFS SLC-40 is 0.88 km away from the coast. Close by infrastructure are roads (0.59 km) and rail (4.28 km). The nearest city to CCAFS SLC-40 is Cape Canaveral and is 18.14 km away.
- B5 boosters have a best success rate with FT a close second. We saw from our SQL queries that B5 boosters can launch payloads of 15,600 kg. FT rockets are capable of launching all payload up to 9,600 kg.
- B4, v1.1 and v1.0 are all low success boosters. B4 boosters are newer than FT, but have lower success rates. We may see more FT launches in the future. V1.0 and v1.1 launched less the 5,000 kg payloads and are no longer used.

Appendix

- https://github.com/vishpuro/Coursera/tree/1cfb93551029 23e63c92305658ee892bb0e4a876/capstone

All files used can be found through the hyperlink provided

Thank you!