IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Andrea D’Acquarone
15 September 2021

Table of Contents

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

Executive Summary

- Objective: Define a model in Python to predict if the first stage of the SpaceX rockets will properly land and be reusable. In this way we will be able to better predict the cost of a launch based on its specific input parameters such as payload, orbit, etc.
- Summary of methodologies: after downloading from the SpaceXdata site and from the Wikipedia web page the historical data of rocket launches since the beginning of SpaceX program, we cleaned, standardized and transformed them according to our needs. We have then analyzed the outcome of past missions and investigated data patterns and relationships between input information and outcome result. We visually explored some input factors (like payload, launch sites, destination orbit, etc.) and analyzed the impact on the outcome of the missions, in terms of success or failure of first stage landing. After the EDA we used historical data to instruct and test some Python Machine Learning prediction models, in order to find the model that statistically best fits with a sample of past missions outcomes.
- Summary of results: Some input factors have a clearer relationship with the outcome success of the mission while other determine more blurry relationship. Among the most promising factors there are the orbit radius and the payload mass. Among the tested classification models (Logistic Regression, Support Vector Machine with sigmoid kernelling, Decision Tree Classification and K Nearest Neighbor) the model that best responded was Decision tree. This model is proposed so far as the predicting model for next launches and resulted in an accuracy of feedback above 80%:
  The model may need further tuning as experience data become more and more available and SpaceX program undertakes technology upgrades.

Introduction

Project background and context

The commercial space age is here, companies are making space travel affordable for everyone. Virgin Galactic is providing suborbital spaceflights. Rocket Lab is a small satellite provider. Blue Origin manufactures sub-orbital and orbital reusable rockets. Perhaps the most successful is SpaceX.

SpaceX's accomplishments include: Sending spacecraft to the International Space Station; Starlink, a satellite internet constellation providing satellite Internet access; Sending manned missions to Space.

One reason SpaceX can do this is the rocket launches are relatively inexpensive. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards of 165 million dollars each. Much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch.

Unlike other rocket providers, SpaceX's Falcon 9 can recover the first stage. Sometimes the first stage does not properly land and crashes. Other times, Space X will sacrifice the first stage due to the mission parameters like payload, orbit, and customer.

In order to predict if the first stage of SpaceX rocket will likely successfully land in future missions, we used publicly available information to train a Machine Learning model to reproduce the expected behaviour

Problems we wish to find answers:

- Can we predict the successful landing of first stage module knowing the main input parameters of a launch?
- Can we build a model based on such parameters that statistically reproduces what happens in reality?
- Which are the most impacting input parameters on success of 1st stage landing?
- Can we determine what is the reliability of results obtained with such model?

Methodology

The project consisted of the following stages further described in detail in the next pages:

- DATA COLLECTION - Two methodologies were used:
  - Getting information from SpaceXData.com via a specific API
  - Getting information from Wikipedia through web scraping with BeautifulSoup library
- EXPLORATORY DATA ANALYSIS (EDA) - The following methodologies were applied:
  - Data Wrangling through Pandas and Numpy
  - EDA with SQL through sqlalchemy library, querying data from a DB2 database
  - Interactive visual data analysis: data have also been analyzed with interactive visual tools Folium and Plotly Dash to further identify patterns and relationships.
    - Folium: space analysis to visualize information on geo-referenced maps
    - Dash: dynamic dashboard allowing to return different graphics and charts based on input parameters
- DEFINITION OF A PREDICTIVE CLASSIFICATION MODEL

  The definition of the classification model has gone through the following steps:
  - Data cleaning, preparation and selection, already partly performed in the previous stages of the project
  - Random splitting of the available data in two sets, one dedicated to training of the ML models, the other for testing the quality of each model
  - Optimization and analytical comparison of the tested models in order to chose the most promising predictor

Data Collection – SpaceX API

- Several SpaceX data are publicly available on the web via a specific API to the page of the project. Information about available data and instructions to download may be found at https://docs.spacexdata.com/
- Separate data may be downloaded via GET requests in form of .json files.
- In order to simplify the GET requests we defined a number of functions reproducing each individual request to the API
- We extracted data of rocket launches from the spacexdata API (after the exercise we then used a static copy of the json file available on IBM cloud in order to ensure consistency of data, because information on spacexdata.com may change along time)
- We normalized and converted the data into a dataframe using Pandas command json_normalize
- We reduced the data to only those features in which we were interested and then used the predefined functions to get from the API additional details about booster_versions, payload_masses, launch_sites, etc…
- Such additional information was stored in separate lists that were then merged, together with some columns of the original dataframe, to form a dictionary with field names and list contents
- The dictionary was then converted again into a dataframe and filtered data for Falcon 9 boosters only to get the final dataset.
- We then replaced missing values on payload mass with an average value

GitHub URL to the Jupiter notebook:
https://github.com/AndreaDak/Final_Capstone_Project/blob/f90642d07efa38613ddb70b22b3f4c3d326f56d2/Data%20collection.ipynb

Data Collection - Scraping

Web Scraping of data from Wikipedia:

- For data extraction from Web pages, the BeautifulSoup library has been applied. BS is a scraping tool with embedded functions to convert html code to json and to find recurrent patterns in the code structure, so as to help retrieving data in a structured way
- Requested data from URL, using a static URL pointing at page version dated 09-june-2021 to ensure consistency
- Applied BeautifulSoup to extract all data. In the Wiki page about Falcon 9 launches, historical data start from the third table in the page and include several subsequent tables, all having a similar structure of columns with the Flight number in first column (distinguishing from other tables not related to historical data).
- Extracted column names from the header of the third table in page (index = 2) using a predefined function
- Created an empty dictionary with keys from the column names
- Looping on all tables in the Soup, checked if the first column referred the flight number (so as to select only tables with historical data) and in such case populated the previously defined dictionary with the relevant data deriving from each row of the tables
- Converted the dictionary into a dataframe using Pandas
- Exported to csv file for further usage

GitHub URL to the Jupiter notebook:
https://github.com/AndreaDak/Final_Capstone_Project/blob/18ca92a719b6eefbf312f903c9a3998bfbc0142c/Webscraping.ipynb

Data Wrangling through Pandas and Numpy

We have:

- Extracted data of Falcon 9 launches in a Dataframe
- Analyzed data as to check missing data
- Evaluated number of launches per each Launch Site and per each destination Orbit
- Created a numerical landing outcome label from Outcome column called Class which has values 0 / 1 in case of failure / success of stage1-landing respectively
- Evaluated the rate of successful Stage 1 landing as the average value of Class in dataset

GitHub URL to the Jupiter notebook:
https://github.com/AndreaDak/Final_Capstone_Project/blob/6aaa9e0c06c9c9e605ab79e4106a7a279ddca018/1.3%20Data%20wrangling.ipynb

EDA with Data Visualization

- Plotted a scatter plot of Payload vs Flight number and Success Class (as color of dot) to understand if there's a trend pattern of success vs payload along time (flight number being almost progressive along time can also be considered as a measure of elapsed time or program advancement)
- Plotted a scatter plot of Lauch site vs Flight number and Success Class (as color of dot) to understand if there is some dependency of success from launch site along time
- Plotted a scatter plot of Launch Site vs Payload and Success Class (as color of dot) to understand if there is a pattern showing relationship of these two parameters vs success
- Plotted bar chart of success rate per each individual orbit to see if there is a direct relationship between the orbit (distance) and the successful landing (orbit distance determines the launching force and acceleration onto the rocket))
- Plotted scatter plot of Orbit vs Flight number and Success Class (as color of dot) to see trend of relationship between orbit and successful landing along time
- Plotted success rate vs years to show continuous progress of program reliability in the years
- After the EDA, assuming we were able to identify important variables that would affect the success rate, we selected the features to be used in the prediction model (FlightNumber, PayloadMass, Orbit, LaunchSite, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial)
- Categorical features where then transformed in dummy numerical columns using function from OneHotEncoder and all numeric columns casted to float64 preparing input data for the prediction model

GitHub URL to the Jupiter notebook:
https://github.com/AndreaDak/Final_Capstone_Project/blob/18ca92a719b6eefbf312f903c9a3998bfbc0142c/Complete%20the%20EDA%20with%20Visualization.ipynb

EDA with SQL

The following queries have been performed:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- List the date when the first successful landing outcome in ground pad was achieved.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass. Use a subquery
- List the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015
- 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

GitHub URL to the Jupiter notebook:
https://github.com/AndreaDak/Final_Capstone_Project/blob/6aaa9e0c06c9c9e605ab79e4106a7a279ddca018/Exploratory%20Data%20Analysis%20with%20SQL.ipynb

Build an Interactive Map with Folium

- Added each site's location on a map using site's latitude and longitude coordinates
- Marked the success/failed launches for each site on the map to display which sites have high success rates.
- Added a Mouse Position window over the map to show dynamically the coordinates (Lat, Long) of a mouse moving over the map
- Identified Points Of Interest in proximity of a launch site (e.g.closest railway line) and marked coordinates
- Calculated the distances between the launch site and the POI
- Added a marker to the identified POI with a specific icon (e.g. Railway icon), a popup text with the POI name and a tooltip showing the distance from site when hoovering over the icon
- Added a blue line between the launch site and the POI to mark the distance with a popup showing the distance in text
- Observed position of POIs vs Launch Sites to identify patterns/rules for launch site locations

GitHub URL to the Jupiter notebook:
https://github.com/AndreaDak/Final_Capstone_Project/blob/6aaa9e0c06c9c9e605ab79e4106a7a279ddca018/Interactive%20Visual%20Analytics%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

The following actions were made:

- Created a dashboard layout with the following elements:
  - A dropdown list to select Launch Site (including also ALL site option)
  - A pie chart to show the total successful launches count for the selected launch site
  - A slider to select payload range
  - A scatter chart to show the correlation between payload and launch success
- Created a callback function with site-dropdown as input and success-pie-chart as output
- Created a callback function with site-dropdown and payload-slider as inputs, success-payload-scatter-chart as output

The two callback functions have the role to properly filter the data from the data set according to the input parameters, so that the output in the charts will correspond to the input parameter. For instance, if we select launch site “A” in the dropdown list, the first callback function will return data from the dataset filtered to show only launch_site “A” and such data will be used to populate the success pie chart.

In the same way also the second callback gets input from the site dropdown list and it will modify the scatter chart accordingly

Since we have also the “ALL” option in the dropdown list, in such case the dataset shall not be filtered and data for all launch site must be taken. Therefore an if..else statement in the two callbaks allows also for such situation.

GitHub URL to the Python code:
https://github.com/AndreaDak/Final_Capstone_Project/blob/3fe1ba22e49a46787d2dd9fd2124567b4be14d82/spacex_dash_app.py

Predictive Analysis (Classification)

The process actions were:

- Input data derived from the EDA with data visualization module
- Transformed the column ‘Class’ of dataset in a Numpy array to become the target variable Y
- Standardized and transformed X Feature list with StandardScaler preprocessing function
- Splitted the data set X,Y in train and test sets with train_test_split function
- For each of the prediction models Logistic Regression, Support Vector Machine, Decision Tree Classifier and K Nearest Neighbors we did:
  - Define specific sets (ranges) of potential parameters and fit GridSearchCV object with the parameters and X, Y train set to find the best set of parameters.
  - Calculate GridSearchCV best parameters, built-in accuracy with such parameters and accuracy on test set with score method
  - Applie the model to predict results yhat with test set data and plot the relevant confusion matrix
- We have then compared accuracy findings and confusion matrix outputs to determine the most promising prediction model

GitHub URL to the Jupiter notebook:
https://github.com/AndreaDak/Final_Capstone_Project/blob/aec3967c4f956f1d8da6bd21b90055d118609246/W4.1%20ML%20Predicting%20Model.ipynb

Results

Main outcomes of the project are here summarized. Further details are available in the relevant sections

Main exploratory data analysis results:

- Rate is directly related to year of launches therefore to flight number. This somehow confirms the good progress in time for setting up a proper stage1 landing technology
- There is a probable relationship between the orbit (in particular distance from earth) and the success rate. Lower orbits seems to grant better success rate
- There is a probable relationship between the payload and the success rate.
- The apparent relationship between the launch site and the success rate may be affected and mainly determined by the unequal use of launch sites during the time. Cape Canaveral was mostly used at the beginning of the program when failure rate was higher for technological reasons.

Interactive analytics demo in screenshots feedback:

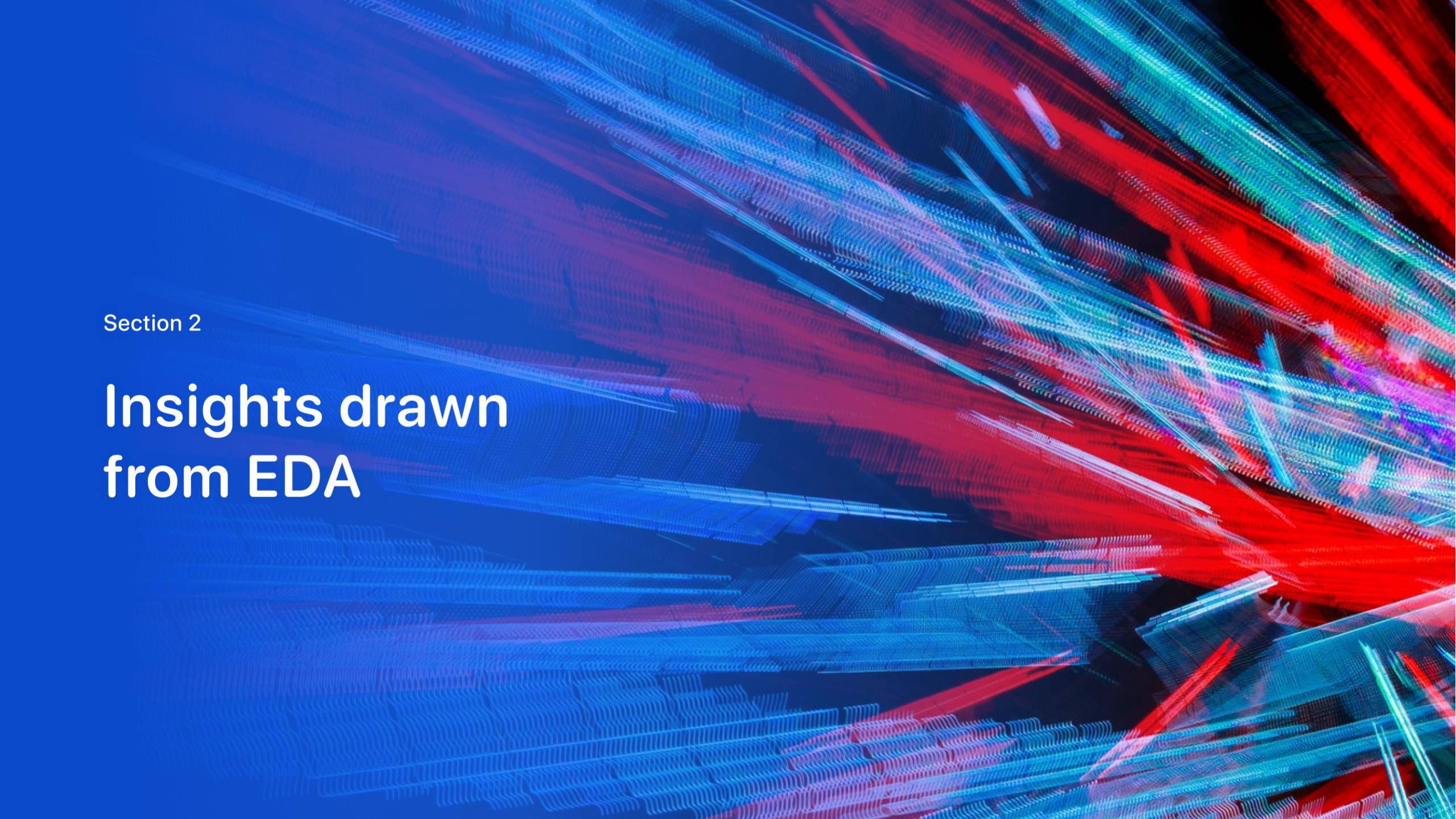
- Dash dashboard makes evident that launches with heavy payloads (>5000 kg) have much higher failure rate of stage 1 landing than light payloads, either for intentional sacrifice of the stage or for effective failure of landing
- For light payloads most of failures are related to V1 boosters while other models have much higher success rate
- Our Folium map visual analysis doesn’t provide much hints on success rate reasons (apart distribution among sites) but provides a lot of useful hints on location of launch sites and what are the triggers in the choice of such locations

Predictive analysis results:

- The most promising prediction model among those tested is the Decision Tree Classifier with parameters {'criterion': 'entropy', 'max_depth': 16, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 5, 'splitter': 'random' }
- Its accuracy on test sample is about 83%

Section 2

Insights drawn from EDA

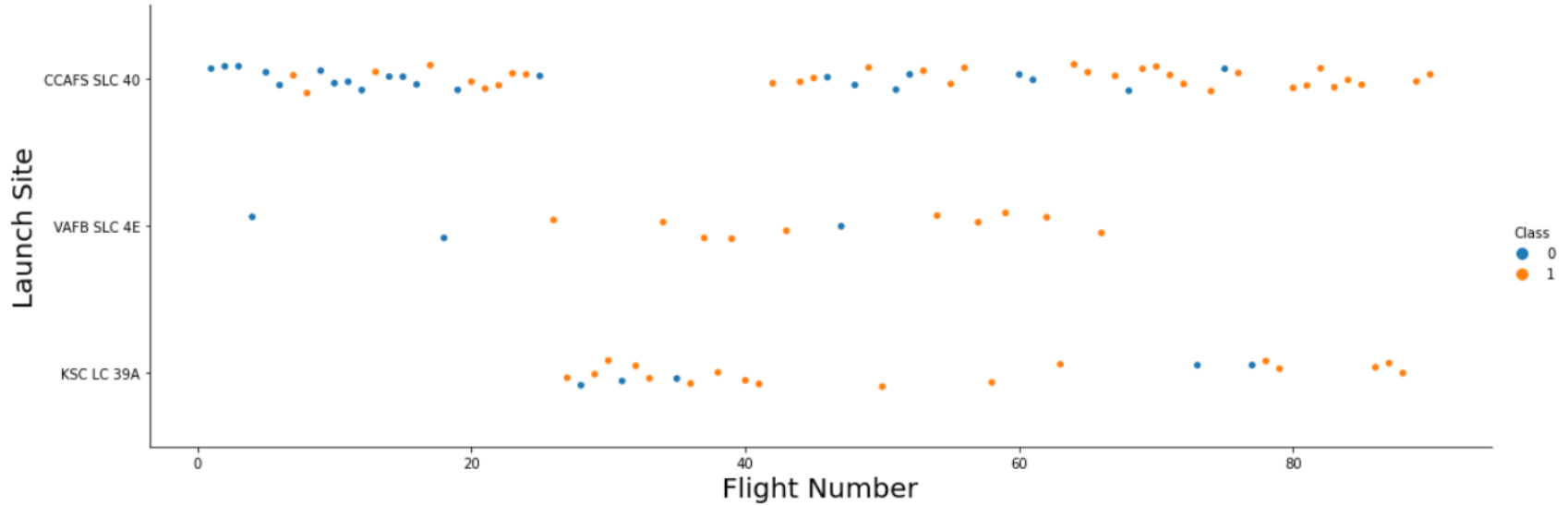


Flight Number vs. Launch Site

TASK 1: Visualize the relationship between Flight Number and Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter x parameter to `FlightNumber`,set the y to `Launch Site` and set the parameter hue to `'class'`

In [4]:
```
# 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(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 3)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

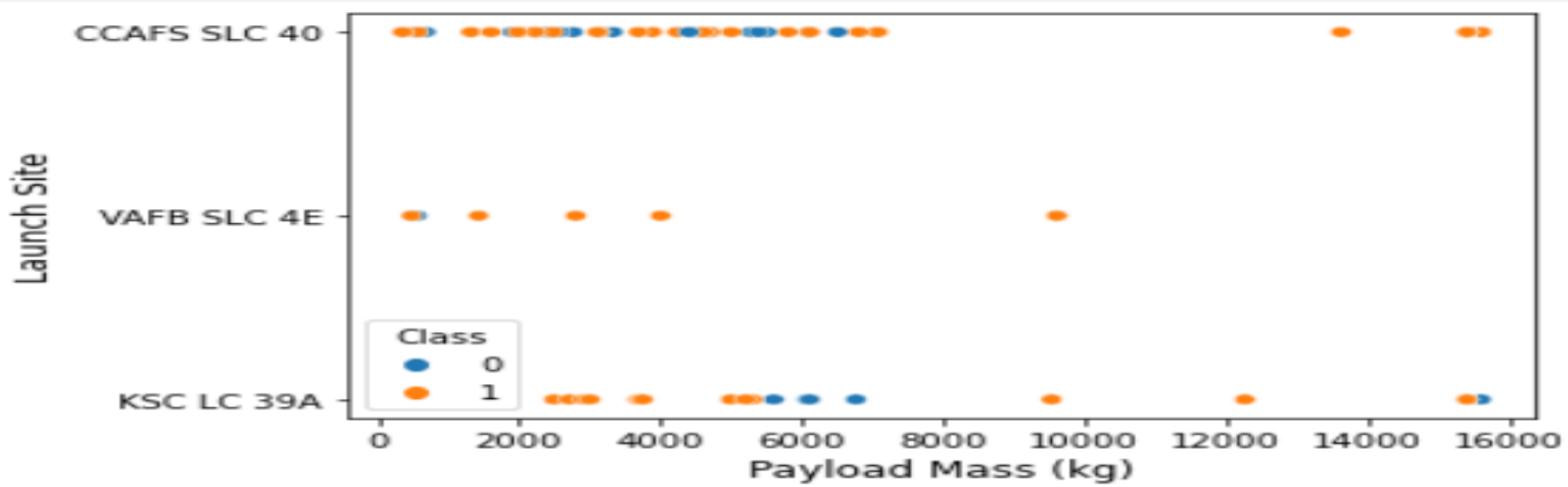


Different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E have a success rate of ~78%. Apparently Cape Canaveral has a much higher failure rate

However, this chart shows that most failures are at the beginning of the program (flight numbers 1- 20), when only Cape Canaveral was used, while failure has significantly decreased for the most recent flights also in this site.

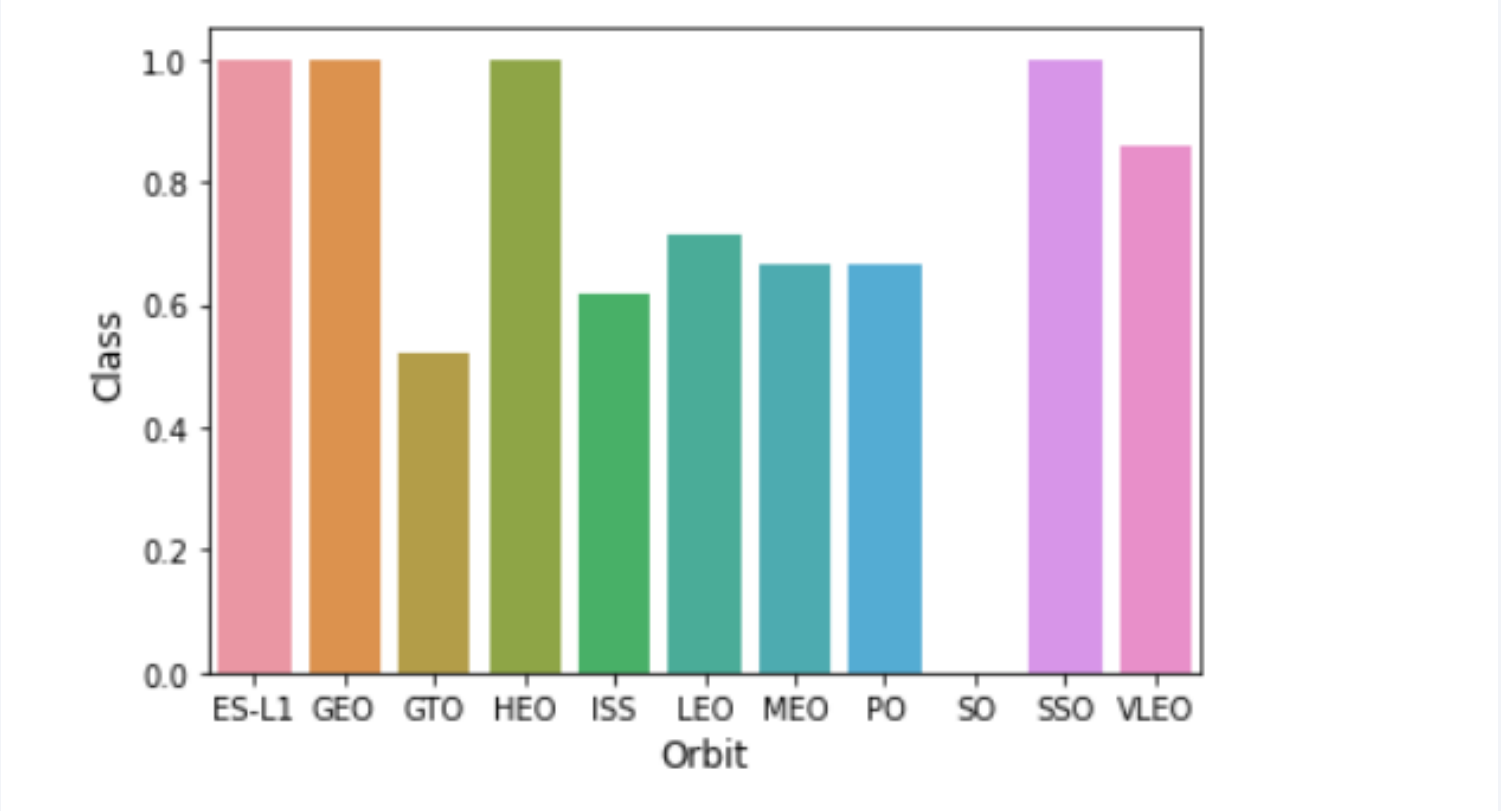
It's difficult to say whether there may be a direct relationship between the launch site and the success rate since number of launches for VAFB and KSC is limited and not equally distributed in time (that is directly related to flight number)

Payload vs. Launch Site



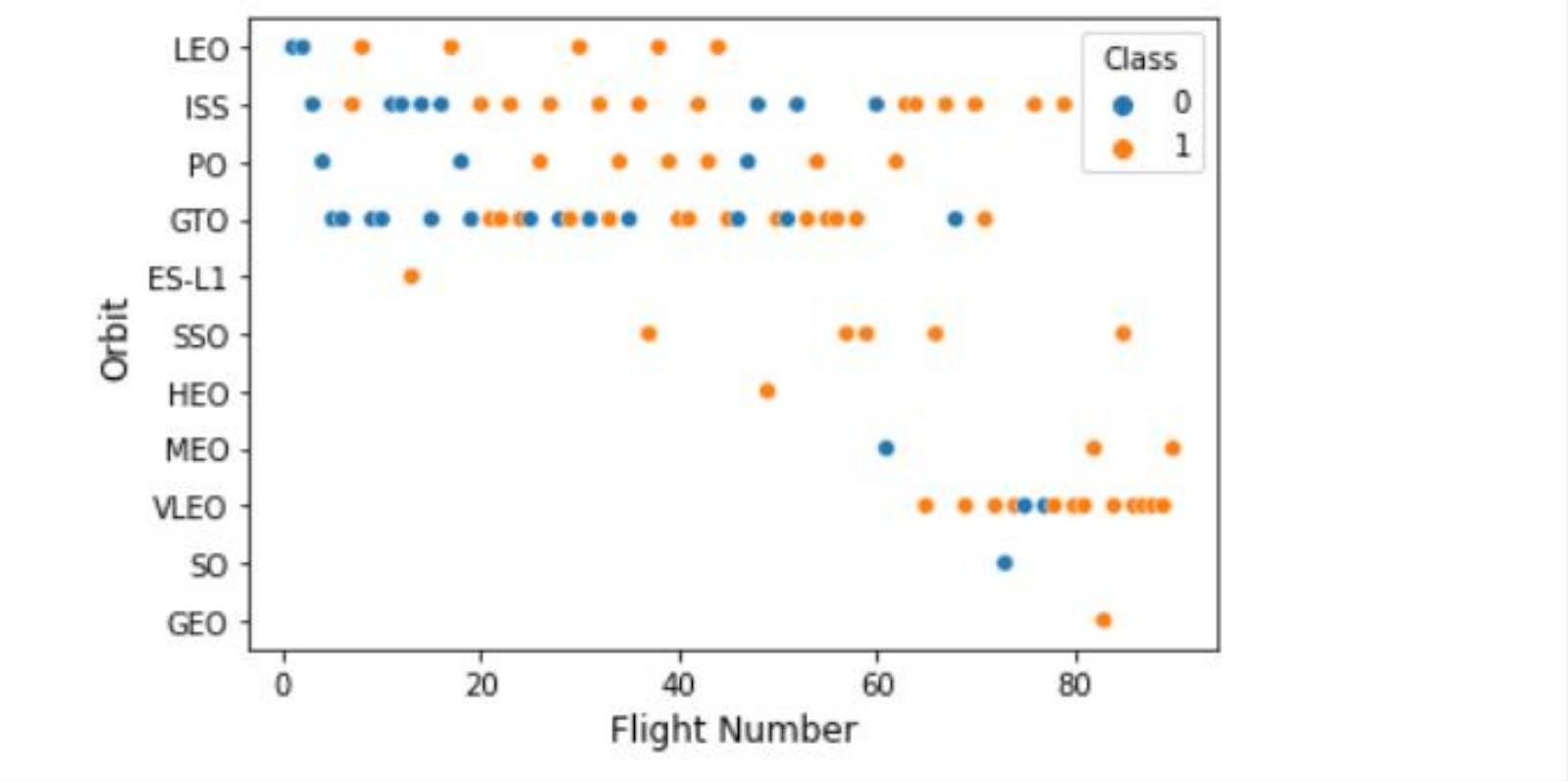
In my view this chart alone doesn't provide much reliable information, unless you are an expert of the SpaceX program and you already know other details that may justify the results. The only remarkable note may be that failures are concentrated mostly in the medium payload range, around 6000 kg. This may be due (but it's only a rekon) to the fact that failure is proportional to payload but after a certain value of payload special rockets for Heavy Payload are used, and then the failure rate decreases for high payloads

Success Rate vs. Orbit Type



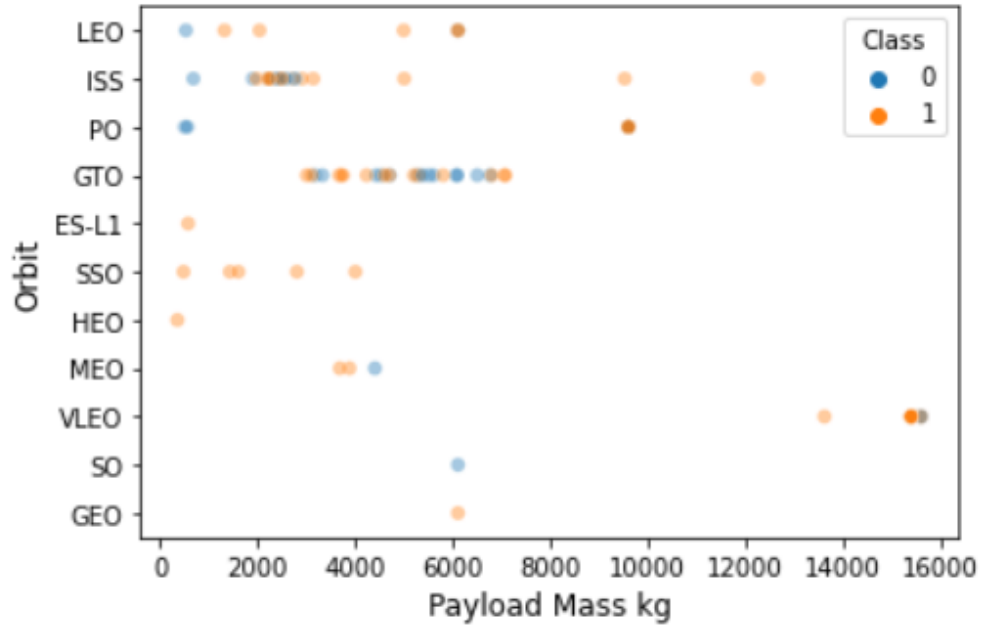
Also this chart, if taken alone, in my view doesn't provide very significant information . Apparently Orbits ES-L1, GEO, HEO and SSO have the best rate, with 100% of successes. However, if we look at the total number of flights per Orbit, we discover that ES-L1, GEO and HEO have only one launch each, therefore information is not much statistically useful. Different situation is for SSO that has a rate of successes of 100% with 5 launches; VLEO has a rate of success above 85% with 12 successes out of 14 launches. If we try to group results by distance from ground we discover that in general smaller Orbits have greater rate of success and this may be a reliable information

Flight Number vs. Orbit Type



If we compare this chart with the previous one we may understand the high success rate of VLEO, because it has been started when the SpaceX program was already quite mature, while other orbits were used since the beginning of the program and they resent of the high rate of failures in the first years

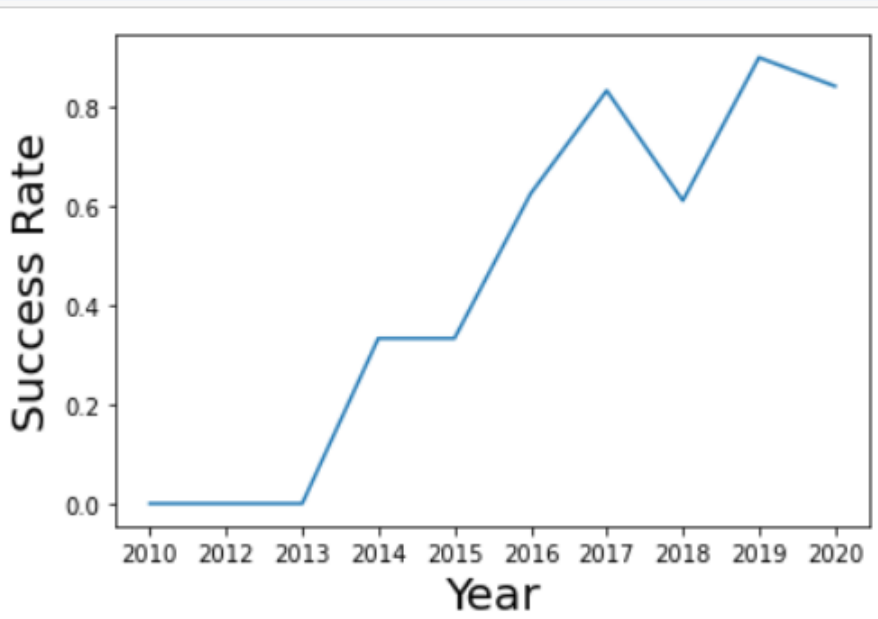
Payload vs. Orbit Type



You should observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

I honestly don't see too much from this chart but I take suggestion as a hint, since the person who has created this exercise probably knows the SpaceX program much better than me

Launch Success Yearly Trend



you can observe that the sucess rate since 2013 kept increasing till 2020

This can be easily explained by the increased maturity of the program

All Launch Site Names

Display the names of the unique launch sites in the space mission

In [4]:
```
%sql select distinct launch_site from spacextbl

# COMMENT: probably due to my poor English, it's not 100%  clear to me if Unique Launch sites means the distinct names of all launch site
s or the sites that have one launch only.
# I assumed the question had the first meaning.
```

```
 * ibm_db_sa://sxr34967:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/BLUDB
Done.
```

Out[4]:

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

As per the comment above, when I prepared the notebook I was not 100% sure of what was requested and assumed that was the names of all sites . Now I am more confident that it was correct.

These names have been extracted from a table stored in DB2 using the SQL magic library and SQL language.

%sql is the command to call in the Notebook the SQL interpreter,

while the rest of the line is the SQL statement to retrieve only unique values of launch_site column from the data table spacextbl

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- Present your query result with a short explanation here

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

```
In [5]: %sql select * from spacextbl where launch_site like 'CCA%' limit 5
```

```
 * ibm_db_sa://sxr34967:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/BLUDB
Done.
```

Out[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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

The SQL statement above selects all the fields in the table spacextbl for the records where the field launch_site begins with CCA, then limits the output to the first five rows only

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here

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

In [6]:
```
# there is one record whose customer name is 'NASA (CSR), Kacific 1' . It's not clear whether this record has to be accounted or not, soo I calculated both cases

a = %sql select sum(payload_mass__kg_) as total_payload_NASA_CRS_including_record_with_Kacific1 from spacextbl where customer like 'NASA (CRS)%'
b = %sql  select sum(payload_mass__kg_) as total_payload_NASA_CRS_excluding_record_with_Kacific1 from spacextbl where customer = 'NASA (CRS)'

print(a, b)
```

```
 * ibm_db_sa://sxr34967:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/BLUDB
Done.
 * ibm_db_sa://sxr34967:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/BLUDB
Done.
+-------------------------------------------------------+
| total_payload_nasa_crs_including_record_with_kacific1 |
+-------------------------------------------------------+
|                         48213                         |
+-------------------------------------------------------+ +-------------------------------------------------------+
| total_payload_nasa_crs_excluding_record_with_kacific1 |
+-------------------------------------------------------+
|                         45596                         |
+-------------------------------------------------------+
```

*As per comment above, the Database includes one record in which Nasa is mentioned as customer together with Kacific 1. at the time of the exercise I was not sure whether this record had to be included in calculation or not, so I calculated both cases with two SQL queries, stored them in variables **a** and **b**, and printed both.*

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Present your query result with a short explanation here

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

In [7]:
```
%sql select avg(payload_mass__kg_) as average_payload_F9_v1_1 from spacextbl where booster_version like 'F9 v1.1%'
```

```
 * ibm_db_sa://sxr34967:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/BLUDB
Done.
```

Out[7]:

average_payload_f9_v1_1
2534

In this query the average of the field payload mass__kg_ is calculated and stored in a column named average_payload_F9_v1_1. The average is calculated only on records starting with "F9 v1.1".

The booster version is filtered with instruction booster_version like 'F9 v1.1%' instead of booster_version = 'F9 v1.1' in order to capture also records that might have blank spaces after the end of text.

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Present your query result with a short explanation here

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

Hint:Use min function

In [8]:
```
%sql select min(date) as First_successful_landing_on_ground_pad from spacextbl where lower(landing__outcome) like '%success%' and lower(landing__outcome) like '%ground pad%'
```

```
 * ibm_db_sa://sxr34967:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/BLUDB
Done.
```

Out[8]:

first_successful_landing_on_ground_pad
2015-12-22

The first date is the date with minimum value. As far as condition, I preferred to split the long text in two parts of string , looking concurrently for lower letters of 'success' and 'ground pad'. This is to limit the risk of errors that might come from slightly different text or text written with capital letters

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
- Present your query result with a short explanation here

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

In [9]:
```
%sql select distinct booster_version from spacextbl where mission_outcome like '%Success' and landing__outcome like '%ship%' and payload_
mass__kg_ between 4000 and 6000
```

```
 * ibm_db_sa://sxr34967:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/BLUDB
Done.
```

Out[9]:

booster_version
F9 FT B1021.2
F9 FT B1031.2
F9 FT B1020
F9 FT B1022
F9 FT B1026

In this query an additional condition on payload within a range is needed

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Present your query result with a short explanation here

List the total number of successful and failure mission outcomes

In [10]:
```
%sql select distinct (select count(*) from spacextbl where lower(mission_outcome) like '%success%') as Success_cases, (select count(*) fr
om spacextbl where lower(mission_outcome) like '%failure%') as failure_cases from spacextbl
#%sql select count(*) from spacextbl where mission_outcome like '%Failure%') as failure_cases
```

```
 * ibm_db_sa://sxr34967:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/BLUDB
Done.
```

Out[10]:

success_cases	failure_cases
100	1

In this query, encapsulated select statement are provided in order to get both results in the same table. The two internal select count the number of success and failure cases respectively, while the external select puts them together in one table only. The distinct clause is to avoid to repeat the values for each record of the table

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Present your query result with a short explanation here

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
In [11]: %sql select distinct booster_version from spacextbl  where payload_mass__kg_ = (select max (payload_mass__kg_) from spacextbl)
 * ibm_db_sa://sxr34967:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/BLUDB
Done.
```

Out[11]:

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

In this query we need first to calculate the maximum payload and then to filter the records that have payload equal to maximum payload. That's why we have an encapsulated select query in the condition clause

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Present your query result with a short explanation here

Task 9

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

In [12]:
```
%sql select landing__outcome, booster_version, launch_site from spacextbl where lower(landing__outcome) like '%failure%' and lower(landin
g__outcome) like '%drone%' and year(date) = 2015
```

```
 * ibm_db_sa://sxr34967:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/BLUDB
Done.
```

Out[12]:

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

This query needs multiple conditions to be concurrently satisfied in the where clause with an “and” statement

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
- Present your query result with a short explanation here

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

In [13]:
```
%sql select landing__outcome, count(*) as count_ from spacextbl where date between to_date('2010-06-04', 'yyyy-mm-dd') and  to_date('2017-03-02', 'yyyy-mm-dd') group by landing__outcome order by count_ desc
```

```
 * ibm_db_sa://sxr34967:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/BLUDB
Done.
```

Out[13]:

landing__outcome	count_
No attempt	9
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

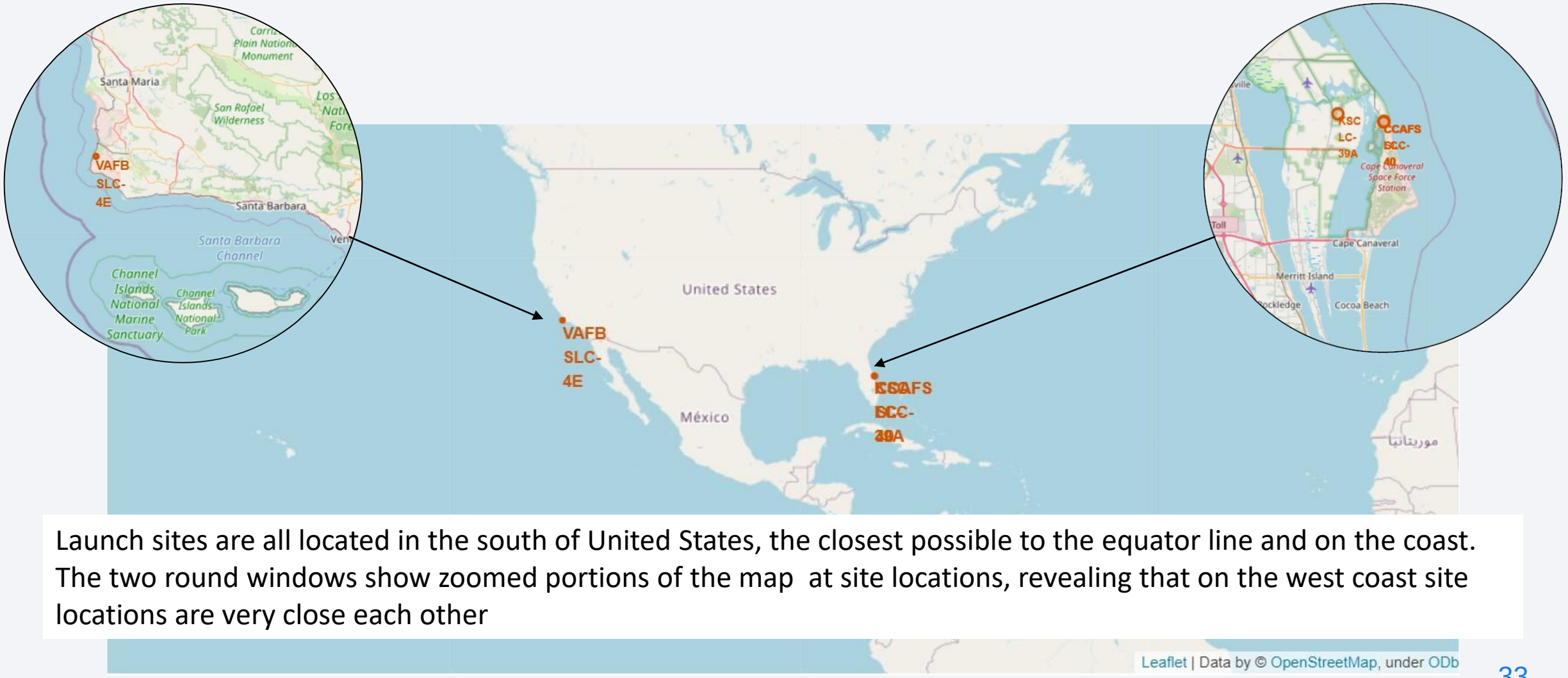
In this query several instructions were needed:

- *Properly read formatted input text dates and convert them into machine date format with the function "to_date(…)" in order to make them comparable with the dates in the database*
- *Select from spacextbl the records with dates within the requested range with the instruction "where date between …. and …."*
- *Group such records according to the content of field landing_outcome with the instruction "group by …"*
- *For each group, select the group identifier with "select landing_outcome" , count the number of records present in the group and store it in a field named 'count_' with the instruction "count(*) as count_"*
- *Order the output table by descent order of counted number of records*

Section 4

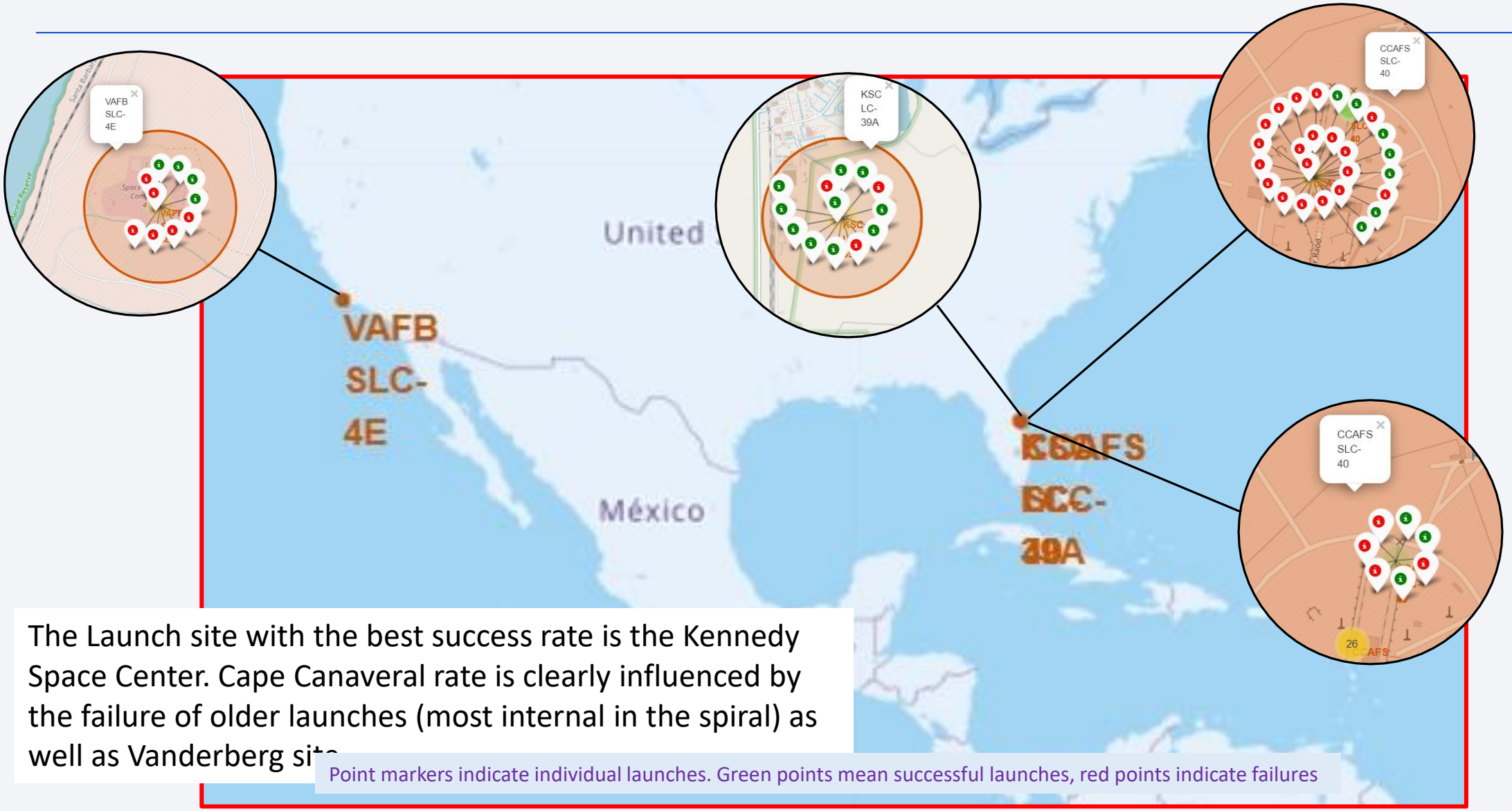
Launch Sites Proximities Analysis

Map of Space X launch site locations



Launch sites are all located in the south of United States, the closest possible to the equator line and on the coast. The two round windows show zoomed portions of the map at site locations, revealing that on the west coast site locations are very close each other

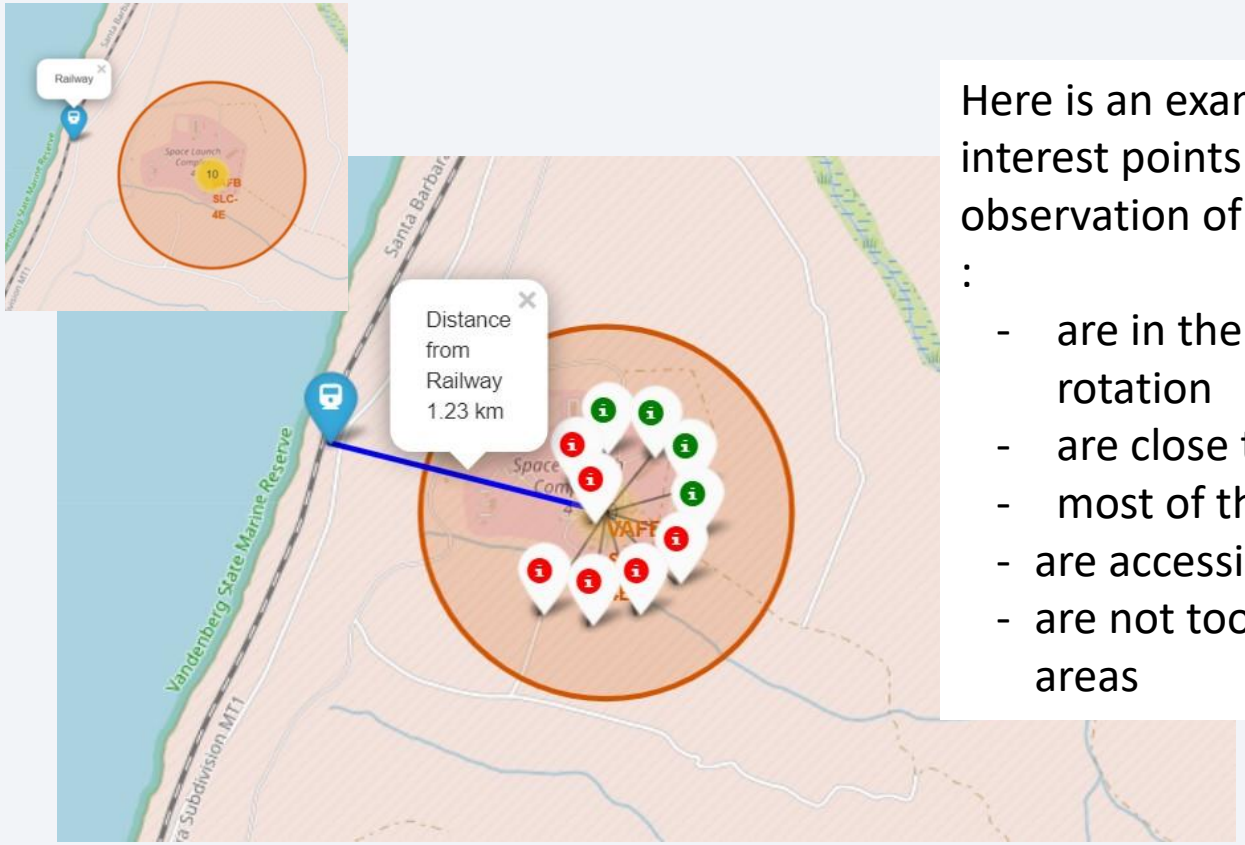
Success/failed launches for each site



The Launch site with the best success rate is the Kennedy Space Center. Cape Canaveral rate is clearly influenced by the failure of older launches (most internal in the spiral) as well as Vanderberg site

Point markers indicate individual launches. Green points mean successful launches, red points indicate failures

Details of additional points close to launch sites

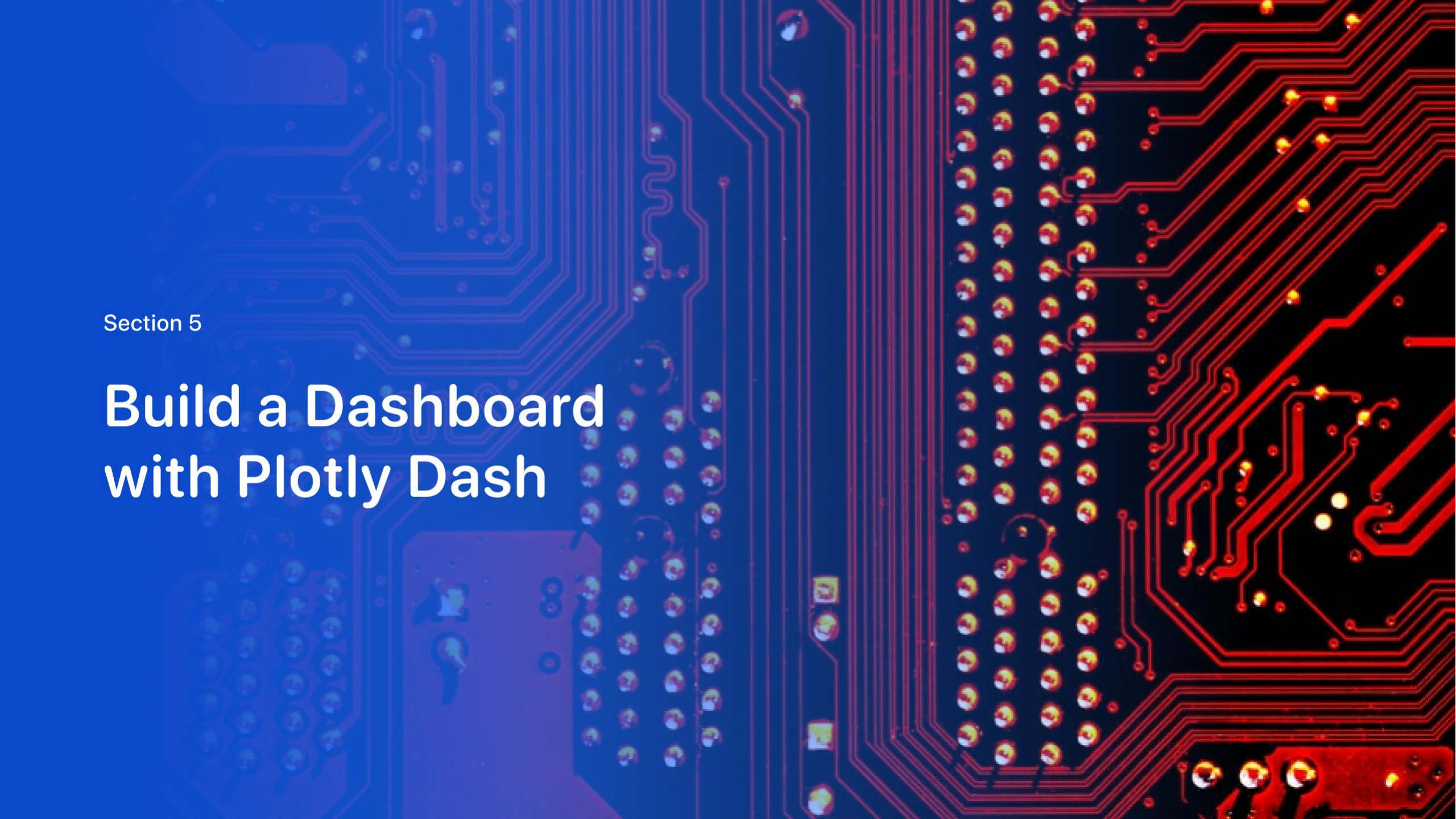


Here is an example of elements that can be added in the map such as interest points and distance lines with details. From a further observation of the launch locations we may say that launch locations :

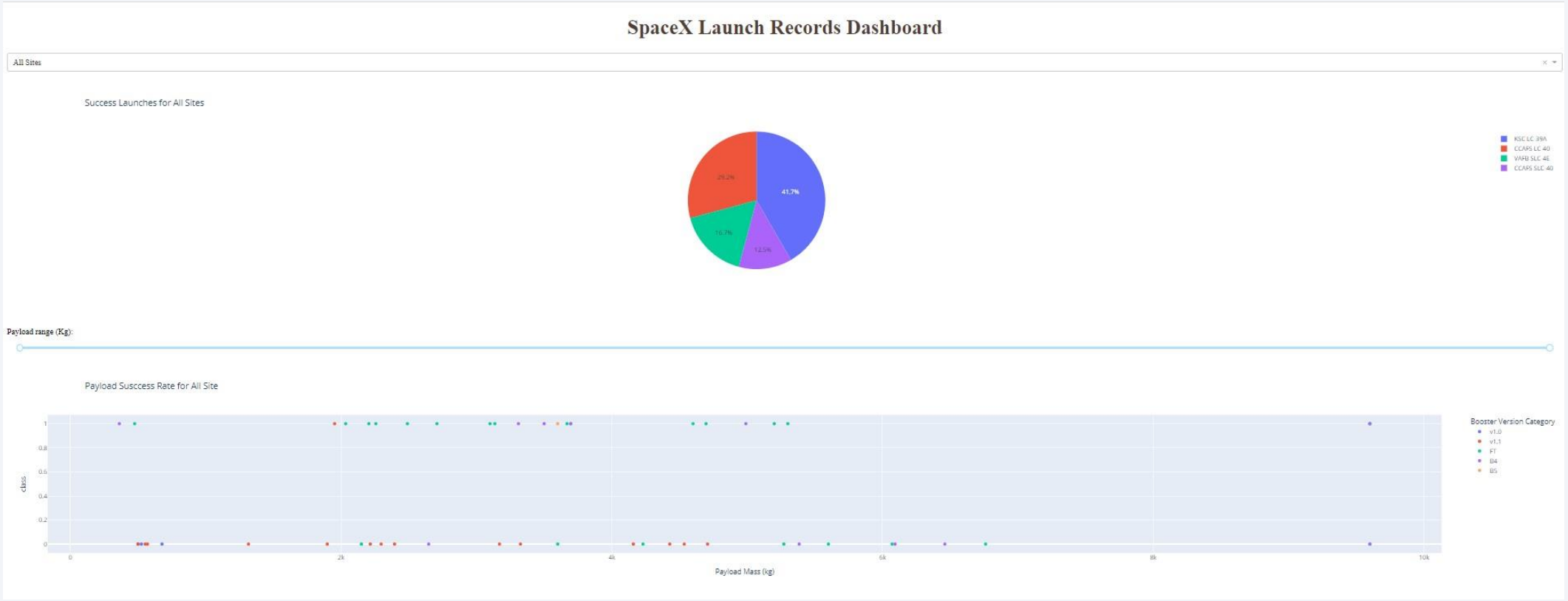
- are in the south of US, close to equator to benefit of earth rotation
- are close to the coast likely for safety reasons
- most of them are on the east coast to benefit of earth rotation
- are accessible with railway
- are not too closed to urban areas or high density population areas

Section 5

Build a Dashboard with Plotly Dash

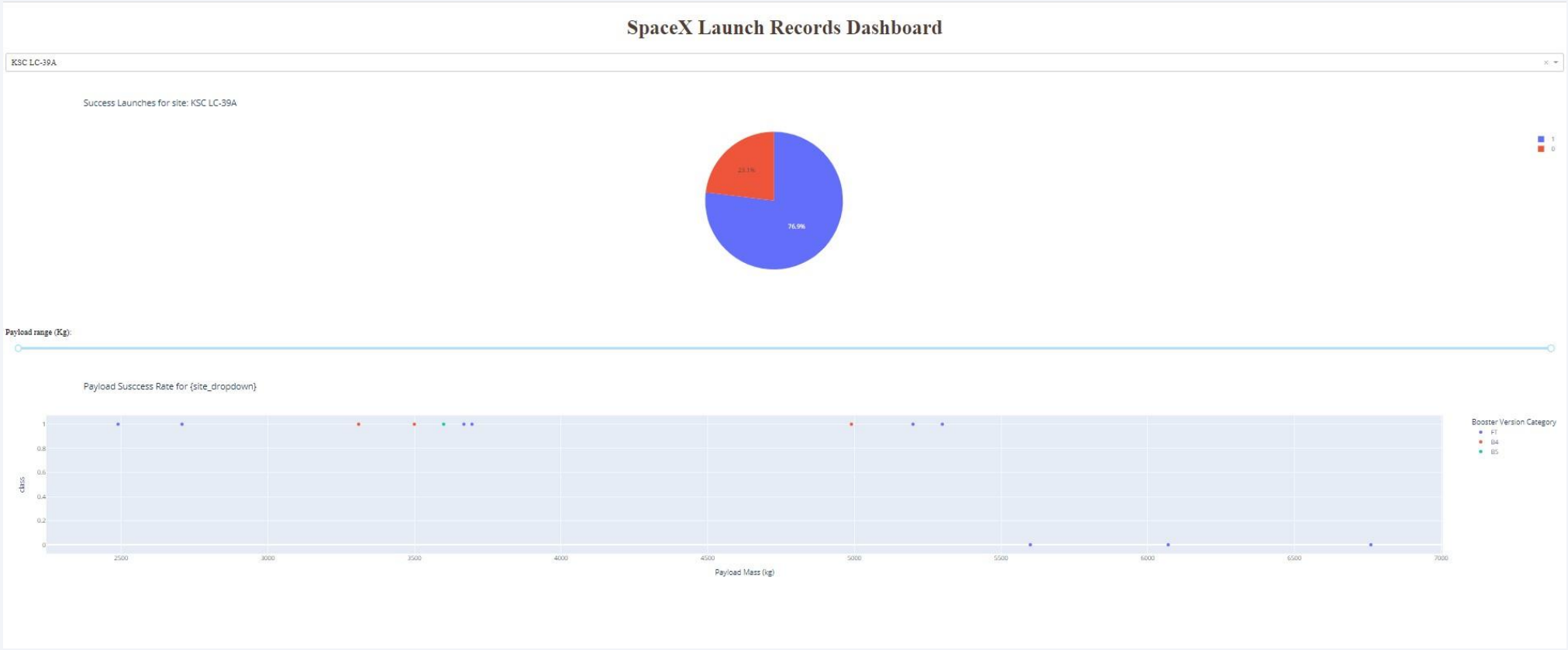


Successful Stage 1 landing for all launch sites



Considerations:
The pie chart shows the overall contribution of each launch site to success missions. However it doesn't show the land-failing launches that re visible in the scatter chart and are quite numerous
From the scatter chart we see that most of the landing with booster V1.x have failed while recoveries with booster FT and B4 were quite successful for payloads until 5000kg and failing above this threshold

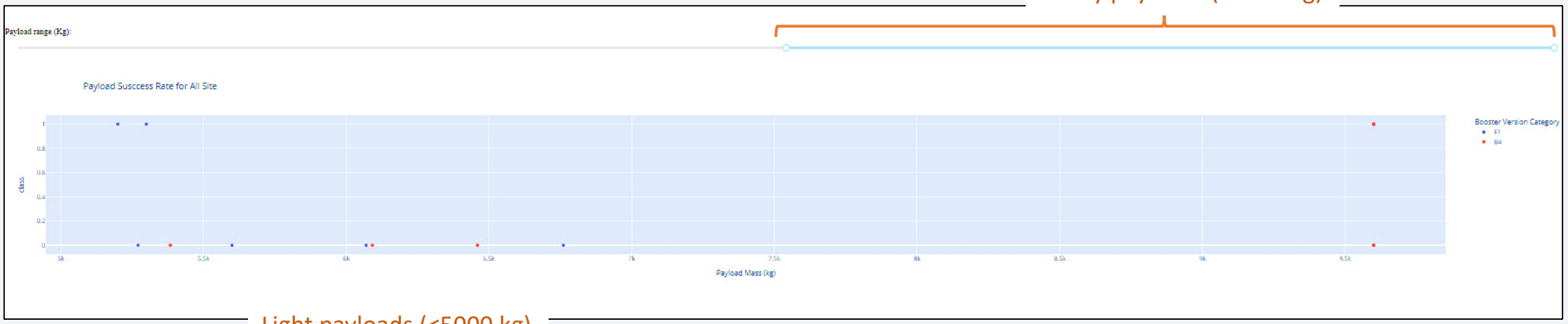
Successful Stage 1 landing for the site with the highest success ratio



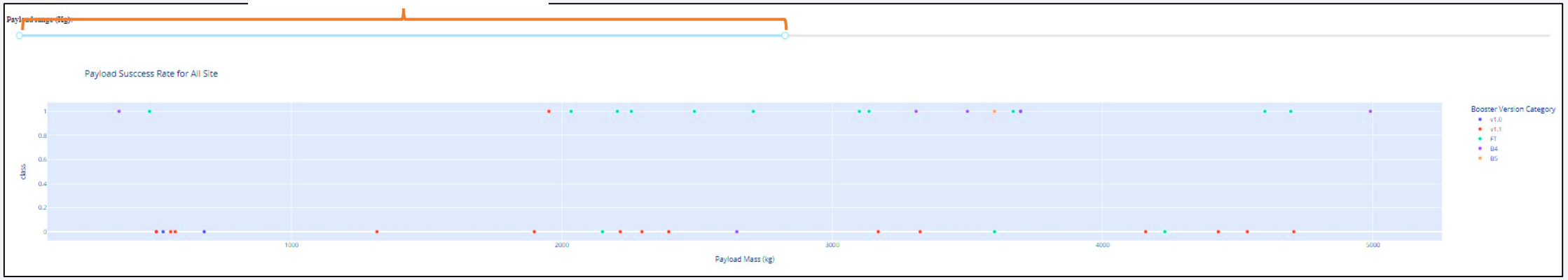
Considerations:
Kennedy Space launch site has an overall high success rate; all launches with payload below 5500kg were successful, while all launches with payload above 5500kg failed

Success/Failure for all sites, light and heavy payloads

Heavy payloads (>5000 kg)



Light payloads (<5000 kg)

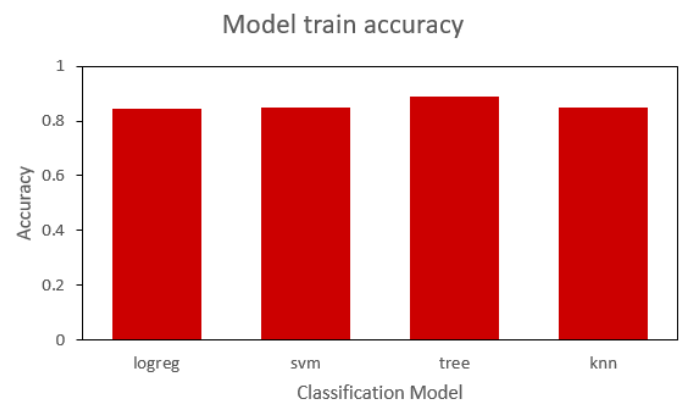


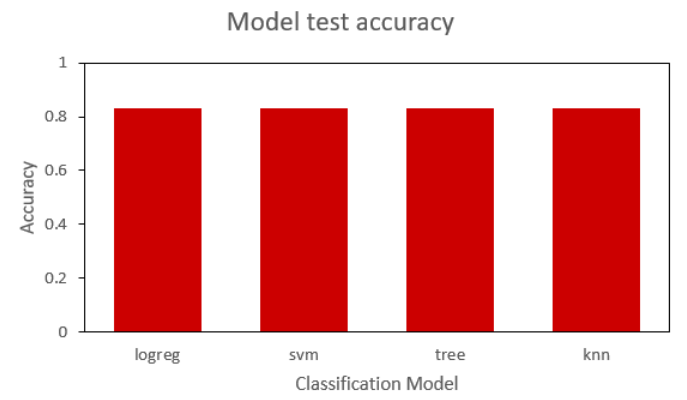
Considerations:
Heavy payloads have a much higher failure rate of stage 1 landing (class = 0), either for intentional sacrifice of the stage or for effective failure of recovering
For light payloads most of the failures are caused by V1 boosters while new generations have a much higher success rate

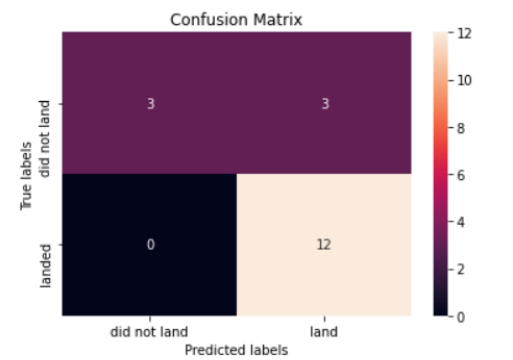
Section 6

Predictive Analysis (Classification)

Classification Accuracy

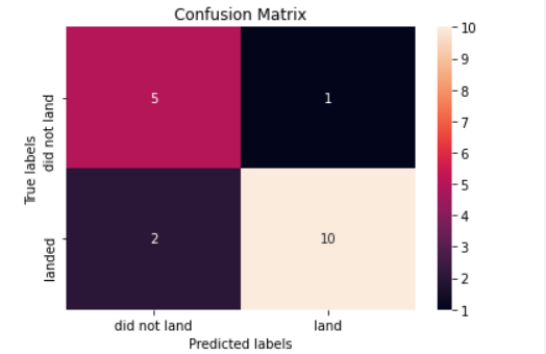






Confusion matrix of logreg, svm and knn

Sensitivity = 0.5
Specificity = 1



Confusion matrix of tree model

Sensitivity = 0.83
Specificity = 0.83

The best seems to be the TREE MODEL
All the four models have identical accuracy of 0.833 calculated with the score method on test set.
GridSearchCV application defines the best parameters for each model among the input parameters options and calculates accuracy against train set. A slightly better built-in accuracy appears in the Tree model vs the other models.
Confusion Matrix shows differences in the Tree model compared to the other three models that have all identical confusion matrix.
In particular, the total of True Positive and True Negative is the same as the other models (15), but in the tree model they are better balanced (5, 10) vs (3, 12). Also wrong predictions are more balanced between false positives and false negatives (2,1) vs (0,3). This results in better sensitivity even though a slightly worse specificity. Since we use predictions for cost evaluation we are interested in getting the most precise prediction in all situations (minimize overestimation and underestimation)
Therefore, even though the overall accuracy is the same, I believe that the Tree model has the most stable and better predicting behaviour in all situations.

Conclusions

- With this model we may predict with decent accuracy (>80%) if the first stage of the rocket will properly land with success and then be reusable or not. This information is valuable in cost determination of future launches which has impact on the project budget, sales price, investment decisions, etc…
- Further refinements can be done with a more accurate visual EDA to determine which input parameters are really determining the success of the mission and which of them only generate “entropy”.
- In addition I believe that a more accurate filtering of the model training data set would produce a more accurate result. For instance, I deem that V1 boosters are old generation and wouldn’t longer be used (but maybe I’m wrong, this is the kind of investigation and discussion that must be taken with expert stakeholders). In such case we could exclude from the model data referring to such boosters.

Appendix

Space X official page	https://www.spacex.com
Space X launch data	https://docs.spacexdata.com/
Wikipedia Space X page	https://en.wikipedia.org/wiki/SpaceX
GitHub project document repository	https://github.com/AndreaDak/Final_Capstone_Project

For specific exercise notebooks and other Python documents please also refer to the URL links already provided in the previous slides of this document

Thank you!