IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Lloyd Tripp
April 29, 2022

Outline

Executive Summary

SpaceX has achieved significant cost savings through booster recovery and reuse making them the top competitor in the space launch service industry.

- Use SpaceX API and scrape data from public websites to gather data on launches
- Select and clean data that supports analysis of booster recovery success
- Visually and statistically analyze the data
- Use machine learning to model booster recovery success
- Conclusion: SpaceX is recovering nearly 90% of boosters resulting is significant cost savings.

Introduction

Project Background

- Space launch industry is growing with new competitors
- Minimizing launch costs will attract more customers
- Booster recovery and reuse is a primary way to reduce costs
- Analyze booster recovery data published by SpaceX and on public websites

Project Goals

- Determine booster recovery rate based on payload, launch site and orbit type
- Model the booster recovery success to predict when a booster is likely to be recovered and thus the launch costs.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Use API to get data published by SpaceX
  - Use web scraping to get launch data from public websites.
- Perform data wrangling
  - Select only data that supports analysis goals.
  - Replace missing payload mass with mean.
  - One-hot encoding of categorical data
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build and evaluate Logistic, SVM, Tree, KNN models

Data Collection

Two data sources:

1. Published SpaceX launch data using API
   - Details on all of the APIs can be found at https://github.com/r-spacex/SpaceX-API
   - For the purposes of this project, these APIs will be used:
     - https://api.spacexdata.com/v4/launches/past
     - https://api.spacexdata.com/v4/rockets/
     - https://api.spacexdata.com/v4/launchpads/
     - https://api.spacexdata.com/v4/payloads/
     - https://api.spacexdata.com/v4/cores/

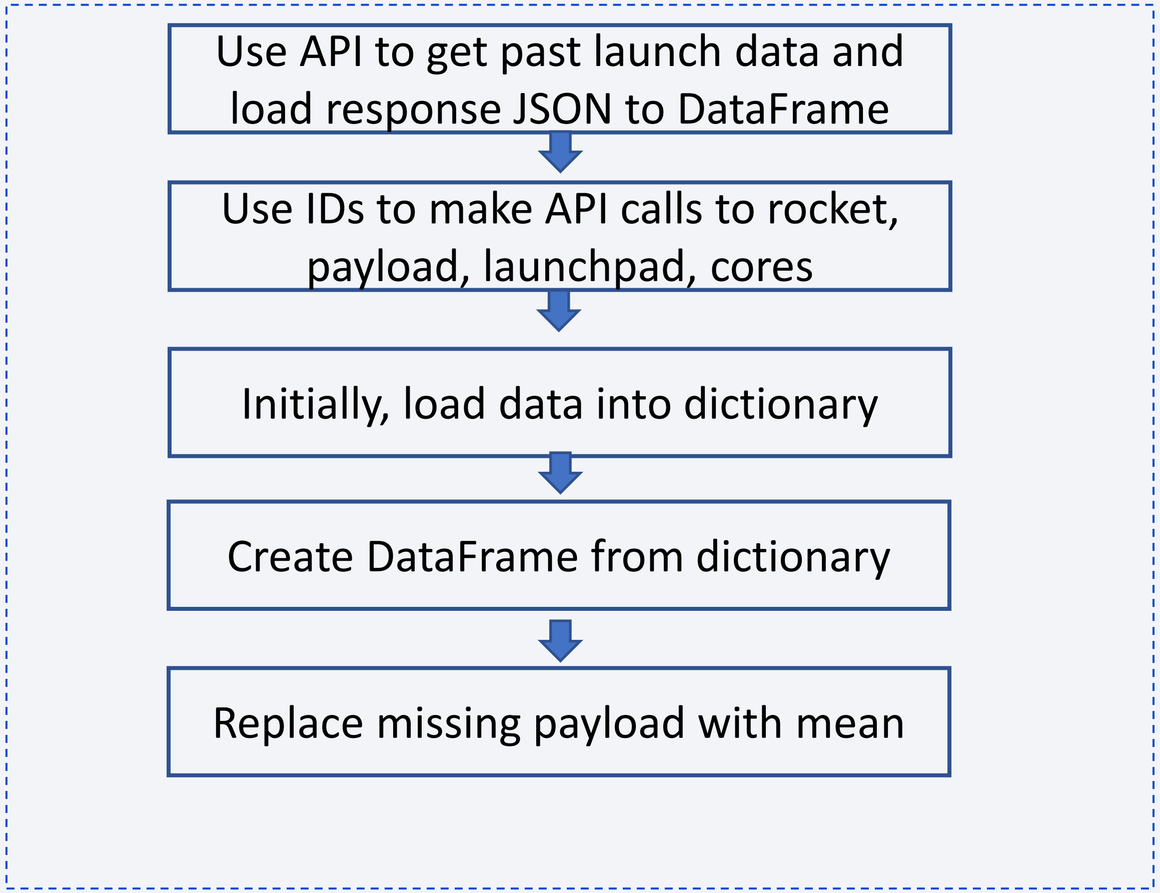
2. Scraping public SpaceX launch data on wikipedia
   - URL: https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=102768692
   - Use beautiful soup library to scrape data from launch history tables

Data Collection – SpaceX API

- Use APIs to request JSON data needed for project
- Load JSON into Pandas DataFrame for easier analysis
- Filter data to get just Falcon 9
- Replace missing payload data with mean

- SpaceX API notebook can be found at:

https://github.com/MrLRTripp/IBM-DS-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

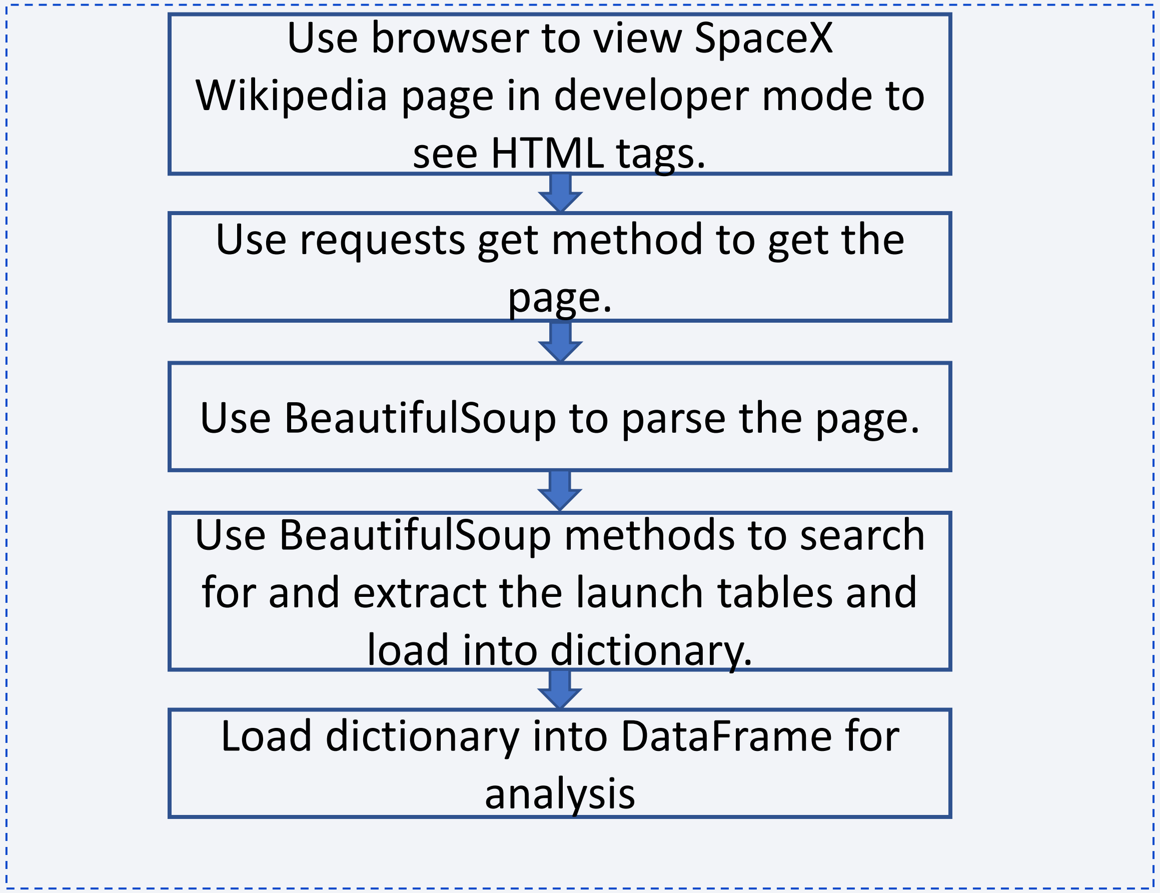


Use API to get past launch data and load response JSON to DataFrame
Use IDs to make API calls to rocket, payload, launchpad, cores
Initially, load data into dictionary
Create DataFrame from dictionary
Replace missing payload with mean

Data Collection - Scraping

- Visually examine the page that will be scraped for design and content.
- Use BeautifulSoup methods to find and extract values – there is some experimentation to get it right due to complex structures of some pages.
- Load values into DataFrame for easier analysis.
- Webscraping notebook can be found at:

https://github.com/MrLRTripp/IBM-DS-Capstone/blob/main/jupyter-labs-webscraping.ipynb

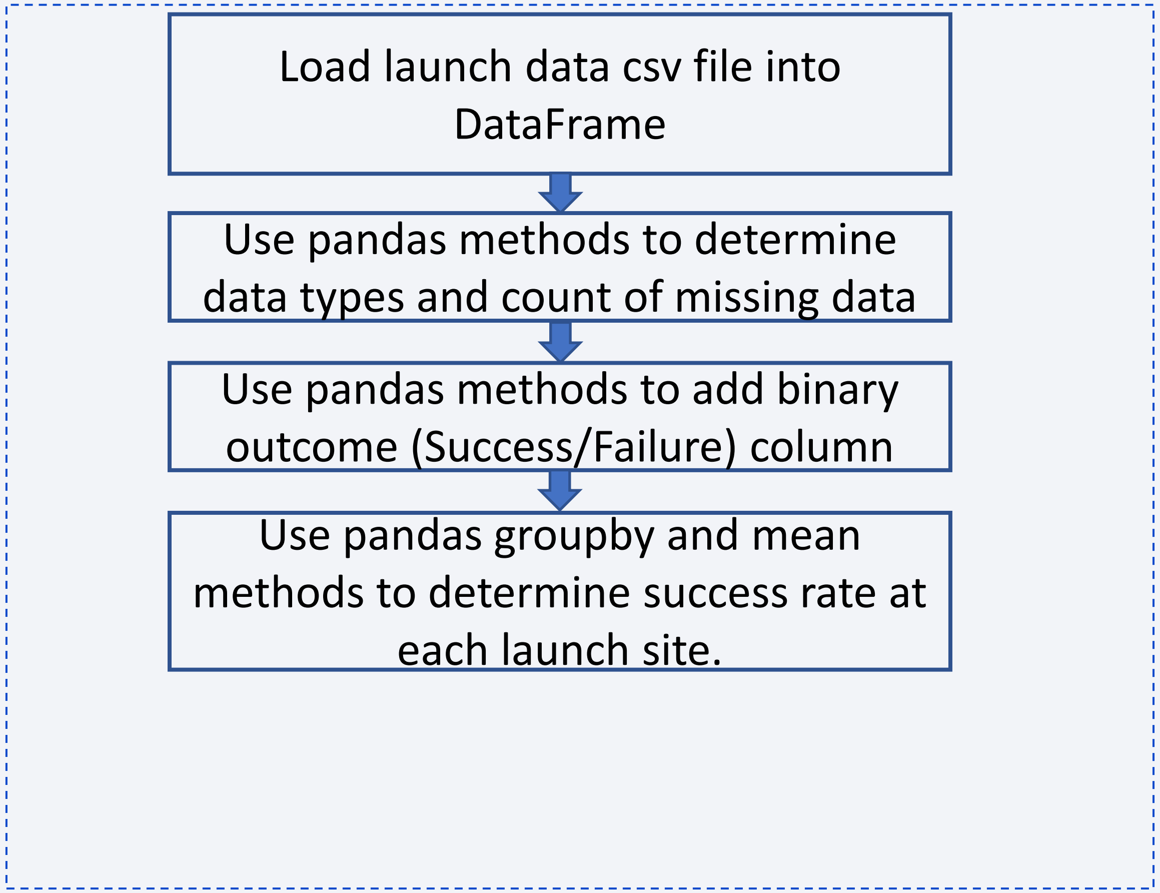


Use browser to view SpaceX Wikipedia page in developer mode to see HTML tags.
Use requests get method to get the page.
Use BeautifulSoup to parse the page.
Use BeautifulSoup methods to search for and extract the launch tables and load into dictionary.
Load dictionary into DataFrame for analysis

Data Wrangling

- After loading launch data into DataFrame, examine data types and look for missing data.
- Since the goal is to determine booster landing success rates and how to predict it, a binary booster outcome column is added to the DataFrame
- Determined landing success rate for each launch site.
- The Data Wrangling notebook can be found at:

https://github.com/MrLRTripp/IBM-DS-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb


Load launch data csv file into DataFrame
Use pandas methods to determine data types and count of missing data
Use pandas methods to add binary outcome (Success/Failure) column
Use pandas groupby and mean methods to determine success rate at each launch site.

EDA with Data Visualization

- Visualization can provide insights into the data and point to correlations that can steer our analysis.
- Plot relationships between key features such as flight number, payload, orbit type, and launch site to look for correlations and trends.
- Expand the DataFrame with one-hot encoding of categorical columns so they can later be used to develop success prediction models.
- The EDA data visualization notebook can be found at:

https://github.com/MrLRTripp/IBM-DS-Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- SQL is a powerful way to analyze tabular data especially if the data is spread across multiple tables. In this case, all the data is in one table so the benefits over a Pandas DataFrame are not as apparent.
- Queries were executed to select launch site data.
- Queries were executed to select and summarize payload data
- Queries were executed to select booster landing success based on payload size, launch site and date range.
- EDA with SQL notebook can be found at:

https://github.com/MrLRTripp/IBM-DS-Capstone/blob/main/jupyter-labs-eda-sql-coursera.ipynb

Build an Interactive Map with Folium

- Markers were added for launch site locations. The proximity to coastline is important for safety reasons. It also explains why some booster landings are on ships.
- Added markers for success/failure of booster recovery at each launch site. A visual indication of which launch sites have higher success.
- Added markers to show proximity of launch sites to nearby infrastructure such as railways, highways and cities. Launch sites need access to transportation infrastructure, but need to be distant from population centers.
- Folium map notebook can be found at:

https://github.com/MrLRTripp/IBM-DS-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

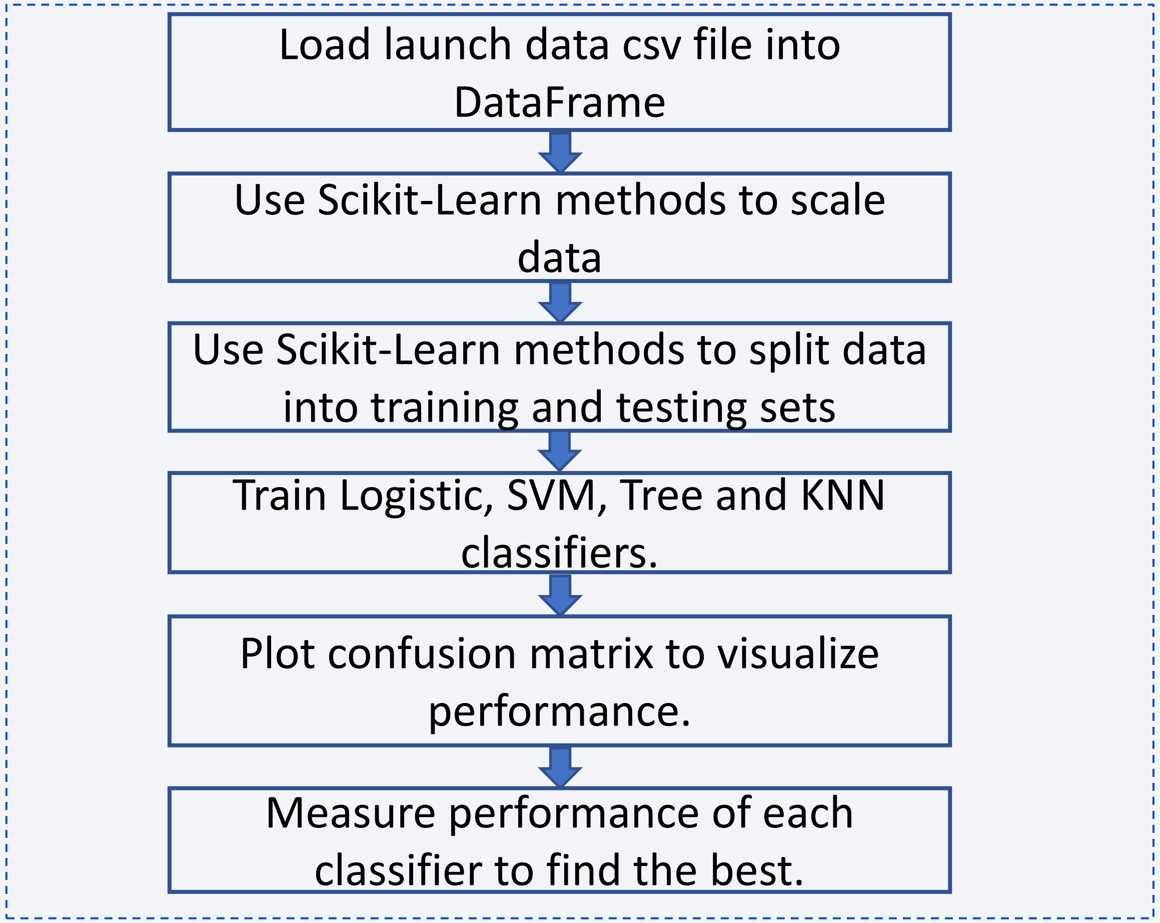
- Created a pie chart for successful booster landing rate at each launch site. A quick visual that clearly indicates some launch location are more successful than others.
- Interactively drill down to each launch site to see success and failure rates. A quick visual indication of the outcome at each site.
- Payload vs. outcome scatter plot that lets user select launch site and payload range. Payload is correlated with booster landing success for different ranges.
- The Plotly Dash lab python file can be found at:

https://github.com/MrLRTripp/IBM-DS-Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Scale and split launch data into train and test sets.
- Train Logistic, SVM, Tree and KNN models using grid search to find best hyperparameters.
- Visually evaluate each model using confusion matrix.
- Numerically evaluate each model using Accuracy, F1-Score and Jaccard Score
- Predictive analysis notebook is located at:

https://github.com/MrLRTripp/IBM-DS-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb


Load launch data csv file into DataFrame
Use Scikit-Learn methods to scale data
Use Scikit-Learn methods to split data into training and testing sets
Train Logistic, SVM, Tree and KNN classifiers.
Plot confusion matrix to visualize performance.
Measure performance of each classifier to find the best.

Results

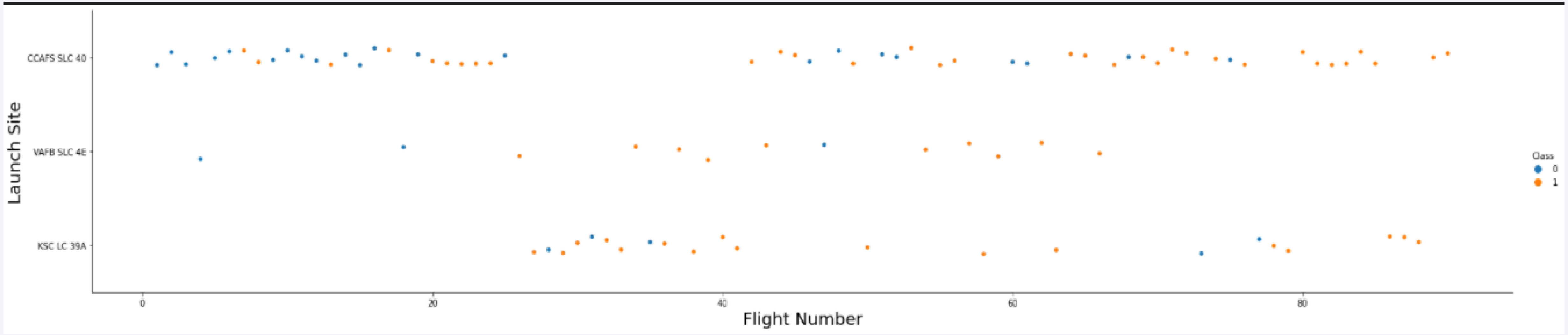
- Using EDA we can narrow down the features to the ones with most predictive power.
- Interactive dashboards is a quick way to visualize different aspects of the data without writing new code.
- Using the trained Tree model, we can predict successful booster landing with about 85% accuracy.

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

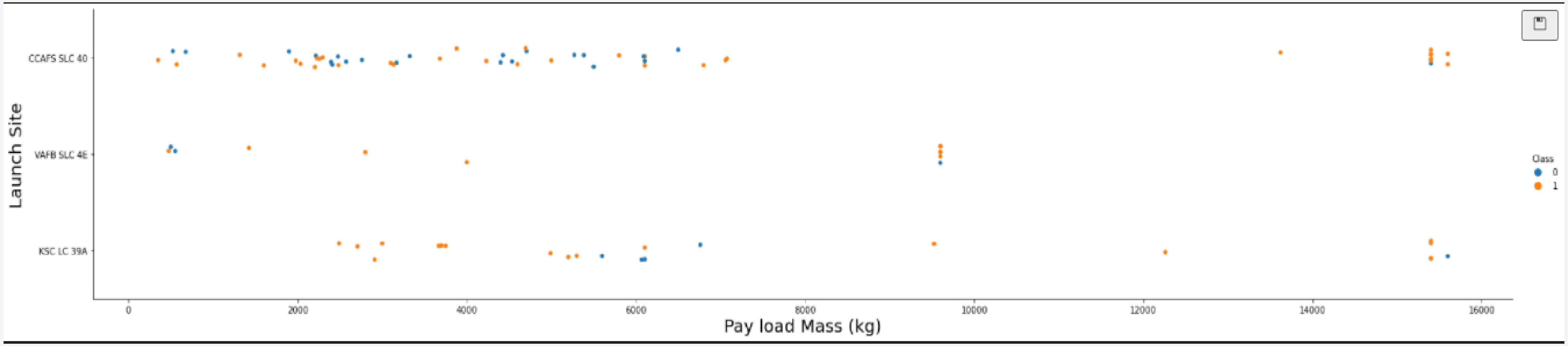
Scatter plot of Flight Number vs. Launch site



This scatter plot shows that successful booster landings improved at each site. In other words, as SpaceX gained experience, then there were more successful booster landings.

Payload vs. Launch Site

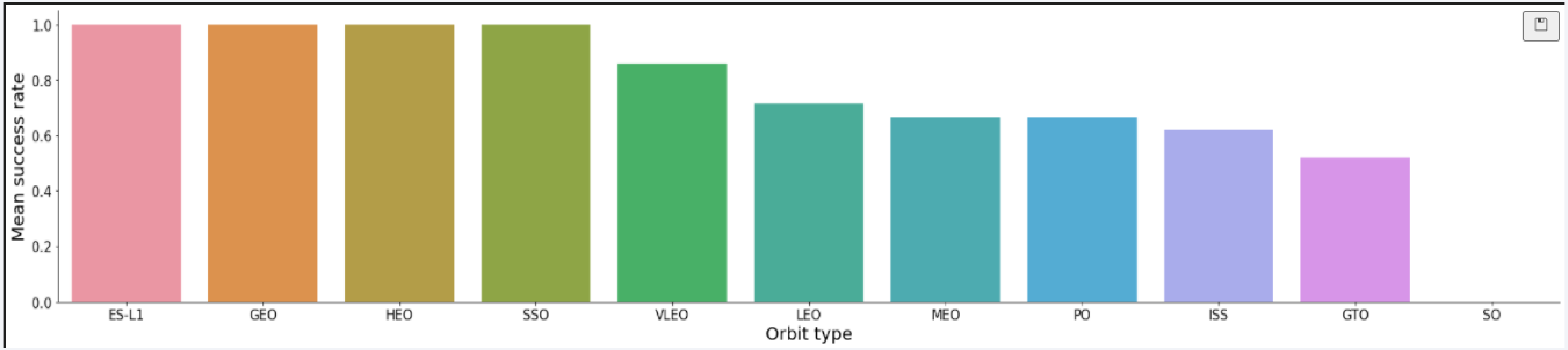
Scatter plot of payload vs. launch site



There are some clear groupings of successful booster landings based on payload mass. Smaller payloads at KSC are very successful as are larger masses at all sites.

Success Rate vs. Orbit Type

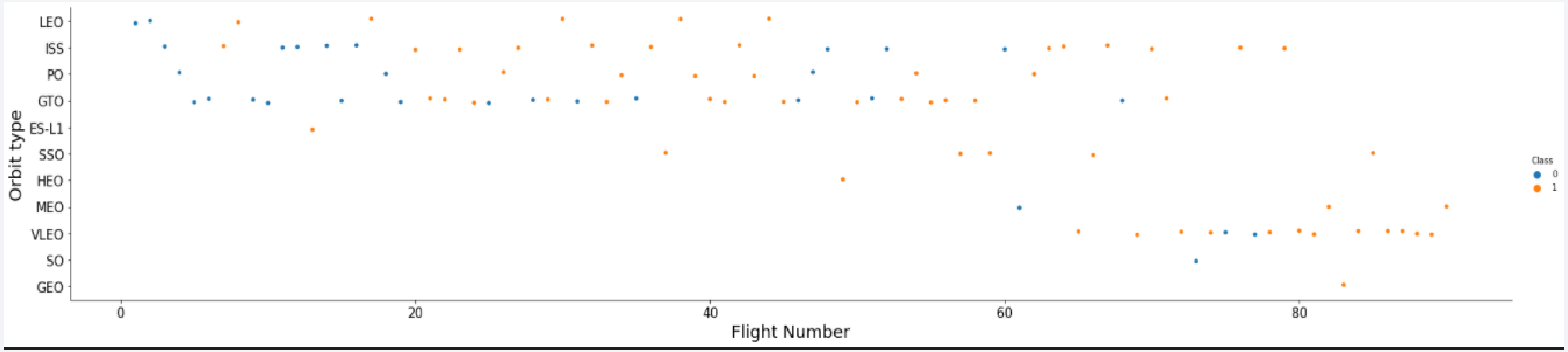
Bar chart of success rate for each orbit type.



Four orbit types have perfect booster landing success. Others are considerably lower. The next chart helps us understand if this is due to launch experience or not.

Flight Number vs. Orbit Type

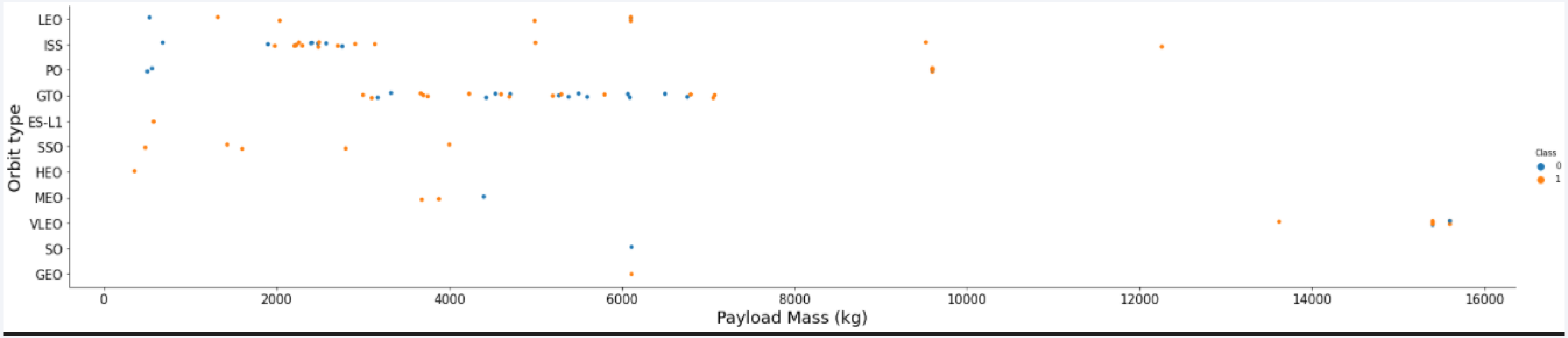
Scatter plot of Flight number vs. Orbit type



Now we can see one factor as to why some more successful than others – there are very few launches to some orbit types and those did have successful booster landings.
Later flight numbers were more successful for all orbit types which shows the benefit of experience.

Payload vs. Orbit Type

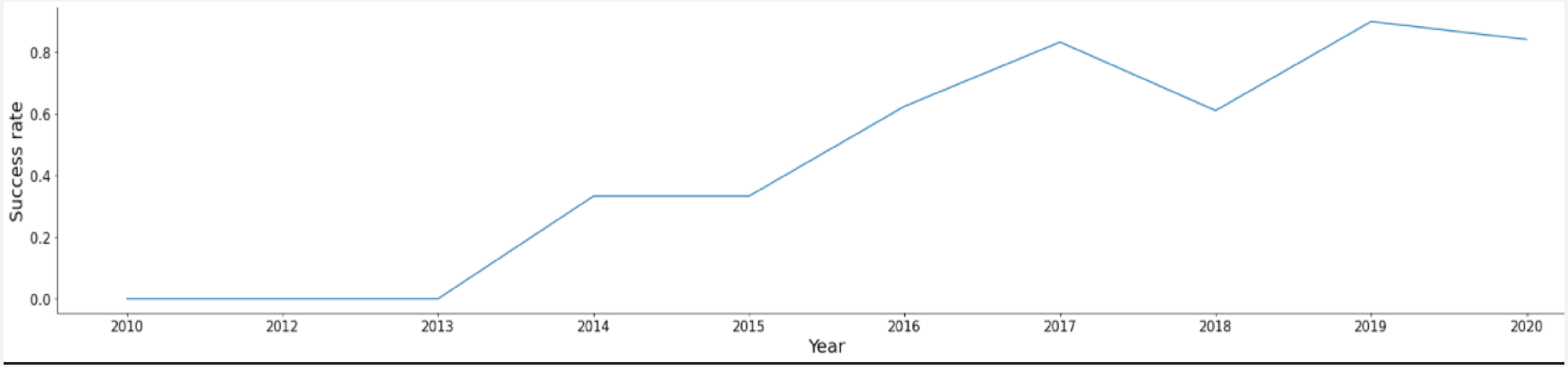
Scatter plot of Payload vs. Orbit type



Most of the payloads are below 8000 kg. However, there is a higher booster landing success rate for heavier payloads.

Launch Success Yearly Trend

Line plot of average successful landings for each year



This clearly indicates that as SpaceX gained more experience, the landing success rate improved. There appears to be an asymptote at about 85%.

All Launch Site Names

- Find the names of the unique launch sites:

```
select DISTINCT(LAUNCH_SITE)
from SPACEXDATASET
```

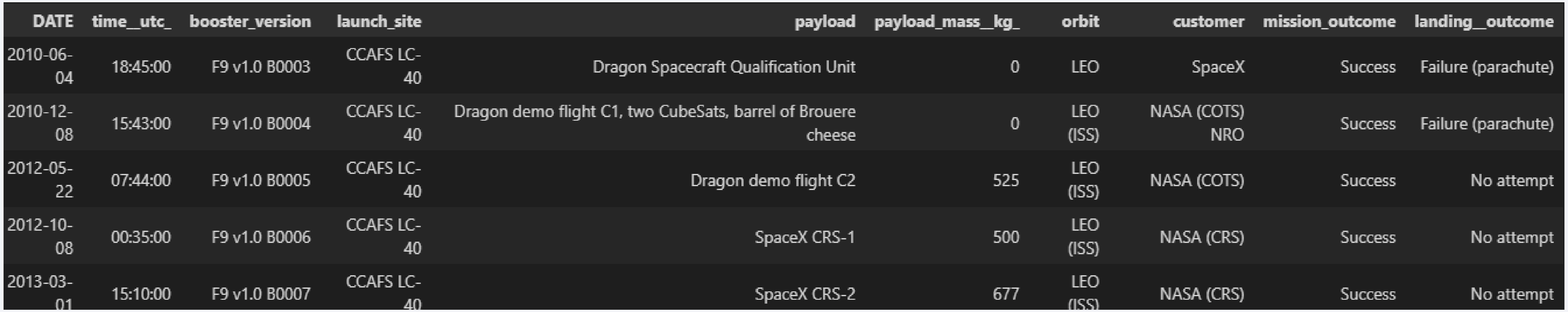
```
  launch_site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E
```

The Distinct qualifier will eliminate duplicates

Launch Site Names Begin with 'CCA'

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

```
select *
from SPACEXDATASET sx
where sx.launch_site like 'CCA%'
limit 5
```



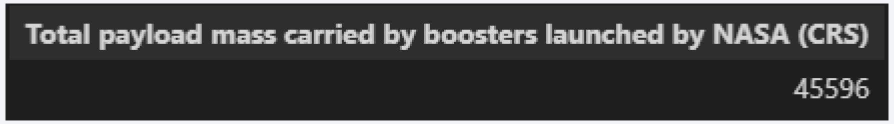
DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	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 % sign is the wildcard character in SQL.
The limit statement limits the number of records returned.

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

```
select SUM(sx.PAYLOAD_MASS__KG_) AS "Total payload mass carried by boosters launched by NASA
(CRS)"
from SPACEXDATASET sx
where sx.customer = 'NASA (CRS)'
```



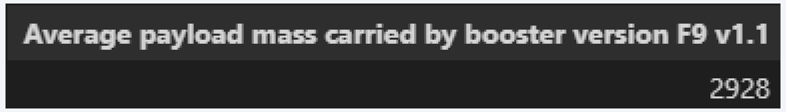
Total payload mass carried by boosters launched by NASA (CRS)
45596

Using the Sum operation, SQL can aggregate values.
The Where statement selects the specific customer we are interested in.

Average Payload Mass by F9 v1.1

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

```
select AVG(sx.PAYLOAD_MASS__KG_) AS "Average payload mass carried by booster version F9
v1.1"
from SPACEXDATASET sx
where sx.BOOSTER_VERSION = 'F9 v1.1'
```



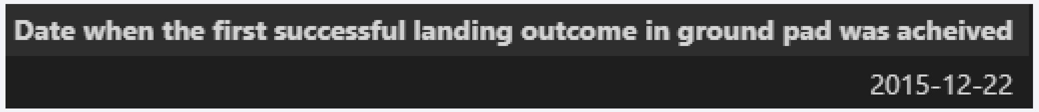
Average payload mass carried by booster version F9 v1.1
2928

The AVG operator computes the average value of a column.
The Where statement limits it to F9 v1.1

First Successful Ground Landing Date

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

```
select MIN(sx.DATE) AS "Date when the first successful landing outcome in ground pad was
acheived"
from SPACEXDATASET sx
where sx.LANDING__OUTCOME = 'Success (ground pad)'
```



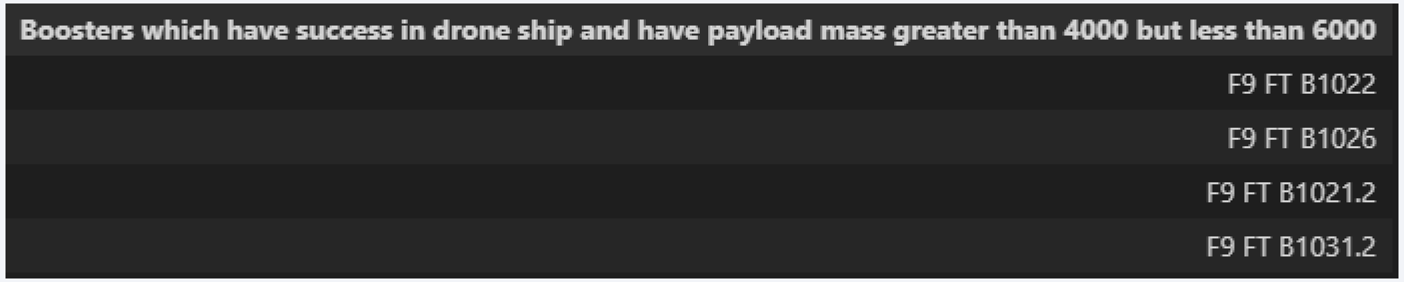
Date when the first successful landing outcome in ground pad was acheived
2015-12-22

Use the Min operator on Date to find oldest date.
The Where statement selects the successful ground pad landing

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

```
select sx.BOOSTER_VERSION AS "Boosters which have success in drone ship and have payload
mass greater than 4000 but less than 6000"
from SPACEXDATASET sx
where sx.LANDING__OUTCOME = 'Success (drone ship)’
AND sx.PAYLOAD_MASS__KG_ between 4000 and 6000
```



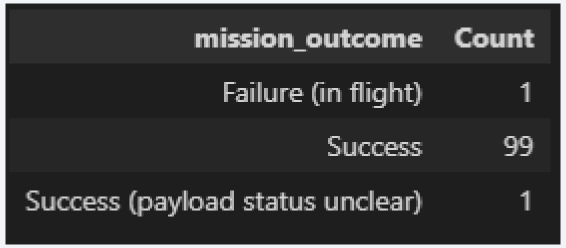
Boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

The Where has an AND condition to limit the payload range

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
select sx.MISSION_OUTCOME, COUNT(sx.MISSION_OUTCOME) as "Count"
from SPACEXDATASET sx
group by sx.MISSION_OUTCOME
```



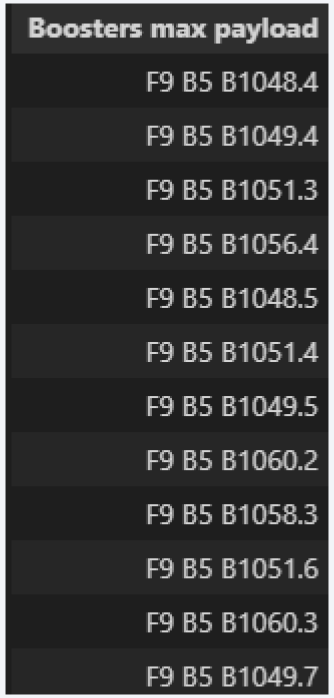
mission_outcome	Count
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

The group by statement shows group results for values in a column
The Count operation simply counts the values in a column

Boosters Carried Maximum Payload

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

```
select sx.BOOSTER_VERSION AS "Boosters max payload"
from SPACEXDATASET sx
where sx.PAYLOAD_MASS__KG_ = (select MAX(sy.PAYLOAD_MASS__KG_) from SPACEXDATASET sy)
```



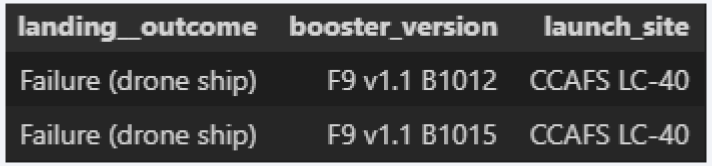
Boosters max payload
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

Use a subquery to determine max payload

2015 Launch Records

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

```
select sx.LANDING__OUTCOME, sx.BOOSTER_VERSION, sx.LAUNCH_SITE
from SPACEXDATASET sx
where sx.LANDING__OUTCOME = 'Failure (drone ship)'
and DATE_PART('YEAR',sx.DATE) = 2015
```



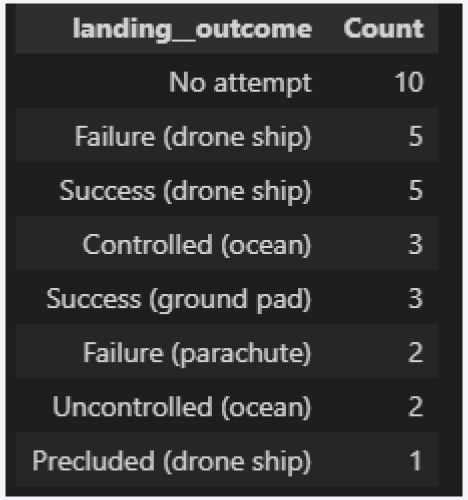
landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

The Date_Part operator is used just to select the year out of the DATE column

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

```
select sx.LANDING__OUTCOME, COUNT(sx.LANDING__OUTCOME) AS "Count"
from SPACEXDATASET sx
where sx.DATE between '2010-06-04' and '2017-03-20'
group by sx.LANDING__OUTCOME
order by COUNT(sx.LANDING__OUTCOME) desc
```



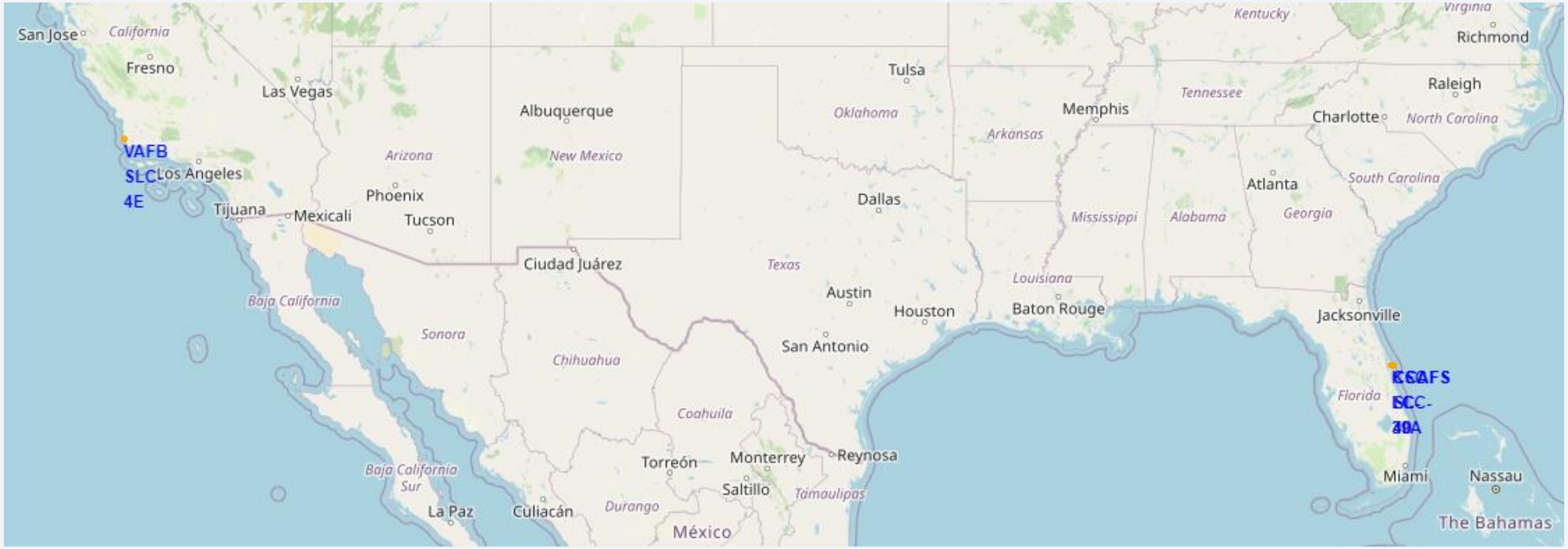
landing__outcome	Count
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Using the group by and order by statements, we can group landing outcomes and display in descending order

Section 3

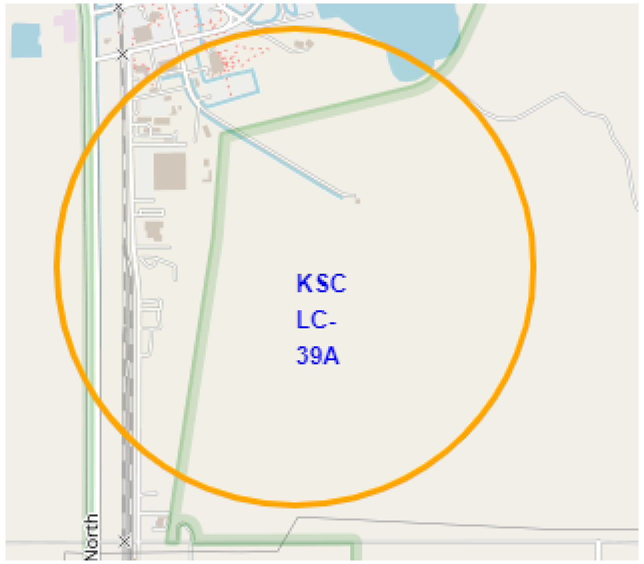
Launch Sites Proximities Analysis

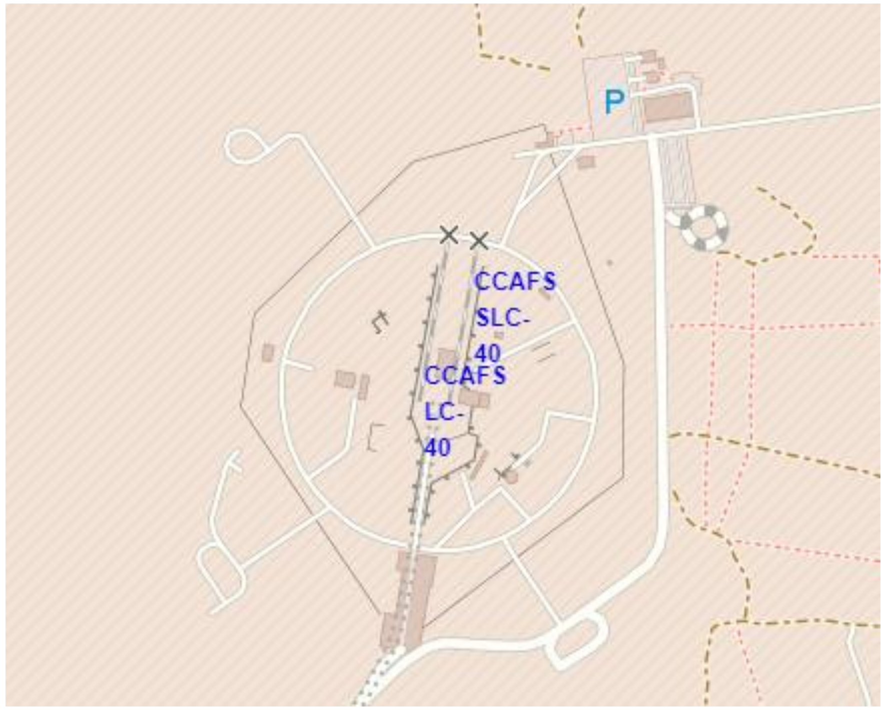
Launch Site Locations



There are four launch site location – one in California and three in Florida. Since the Florida locations are all close together, a zoom of the locations is on the next slide.

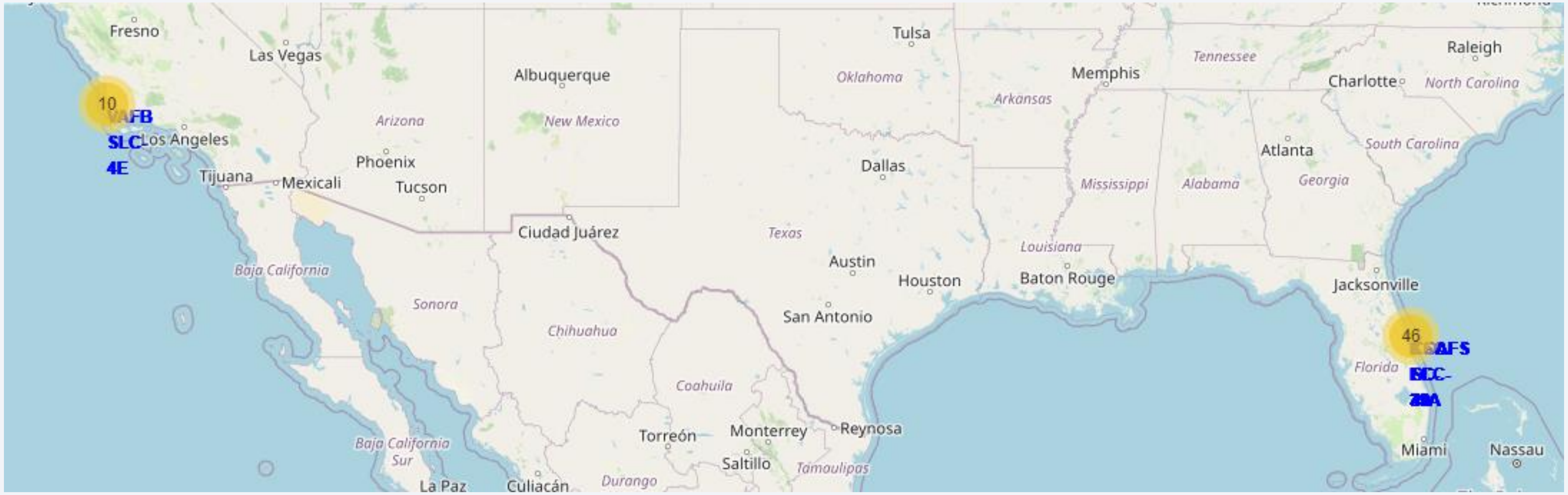
Florida Launch Site Locations





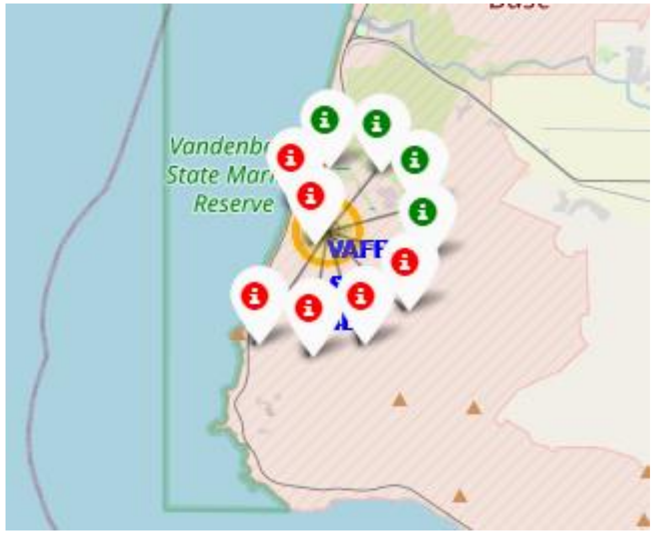
Florida launch locations.

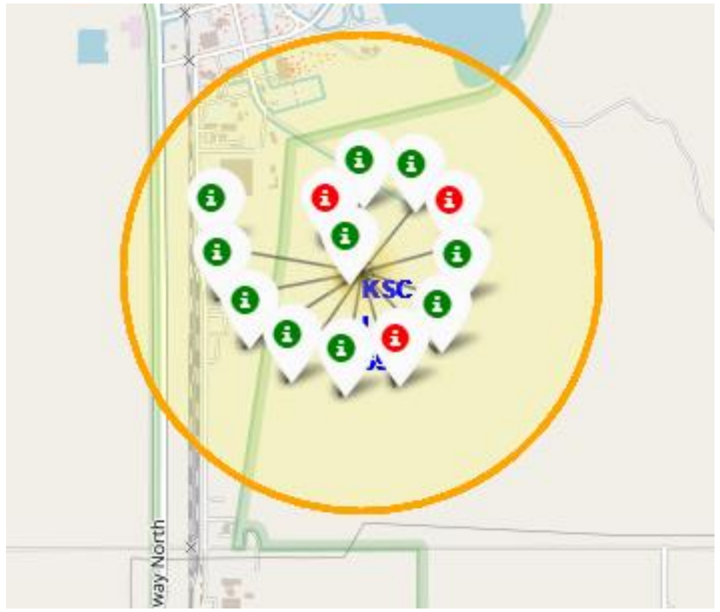
Launch Outcomes at All Launch Sites

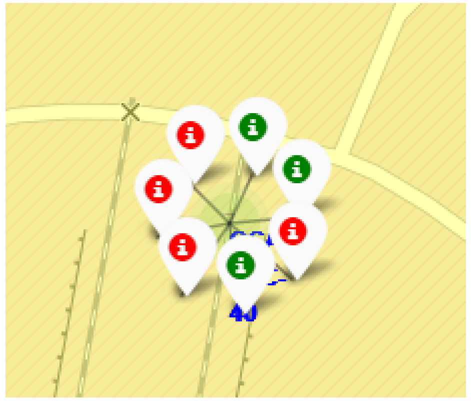


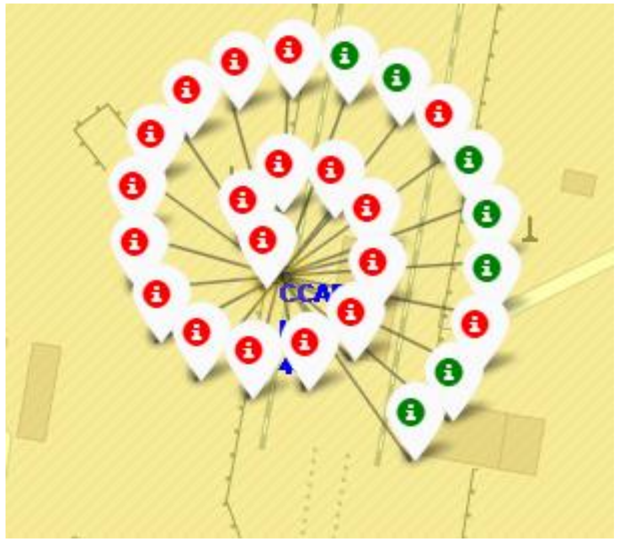
Launch outcome markers added to all sites. Next slide zooms into each site to show detail.

Launch Outcomes at All Launch Sites (Zoom)



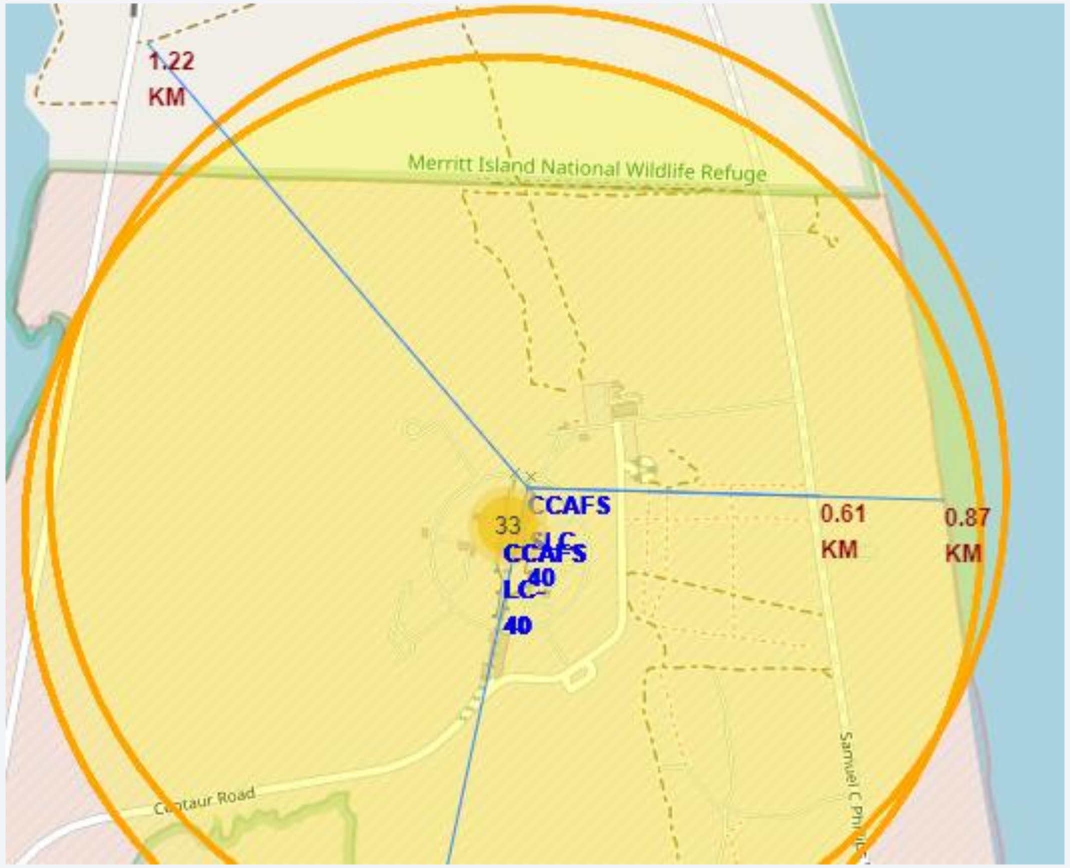




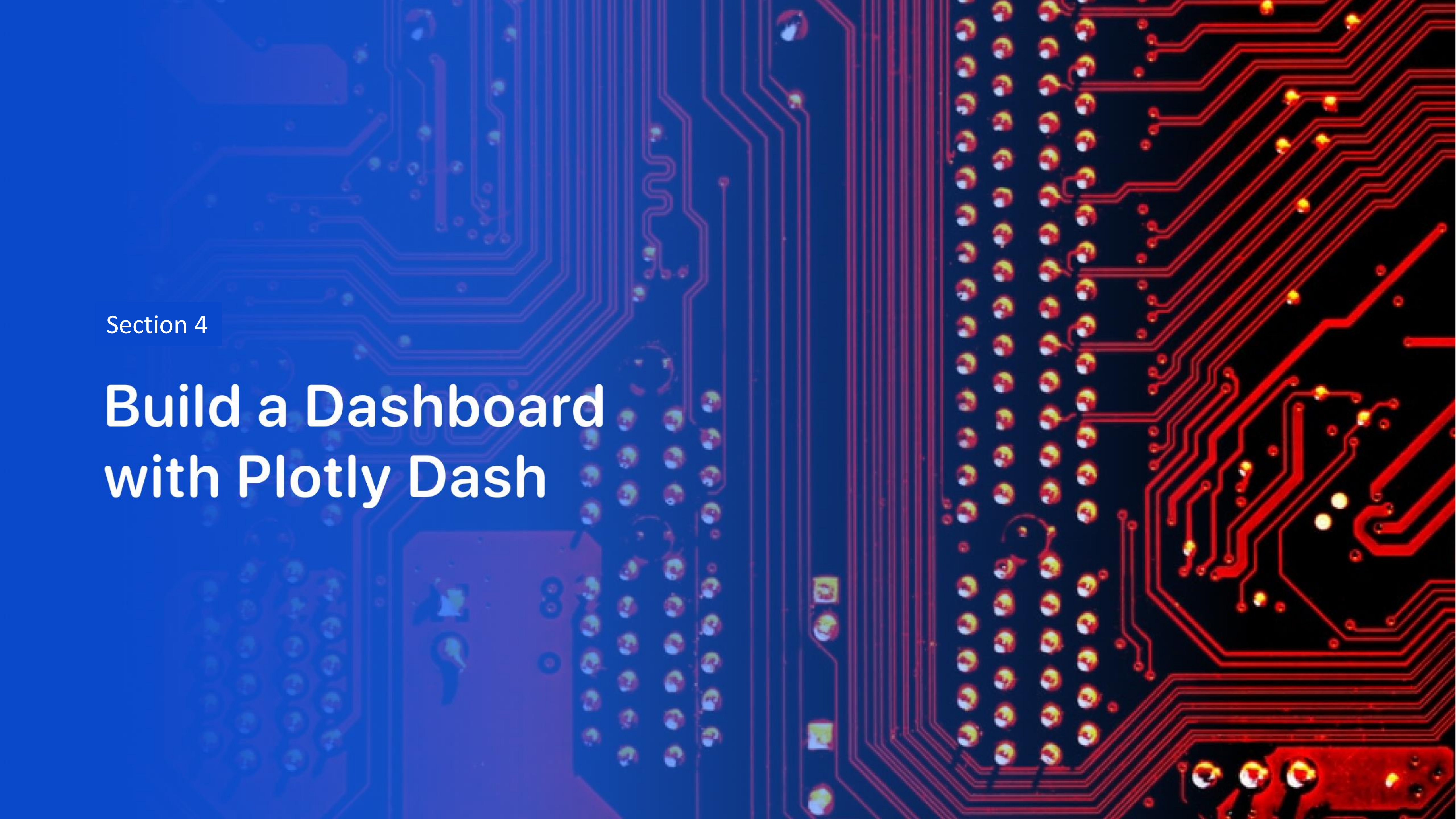


Zoom into each site to show launch outcomes.

Distance from CCAFS SLC-40 to Infrastructure



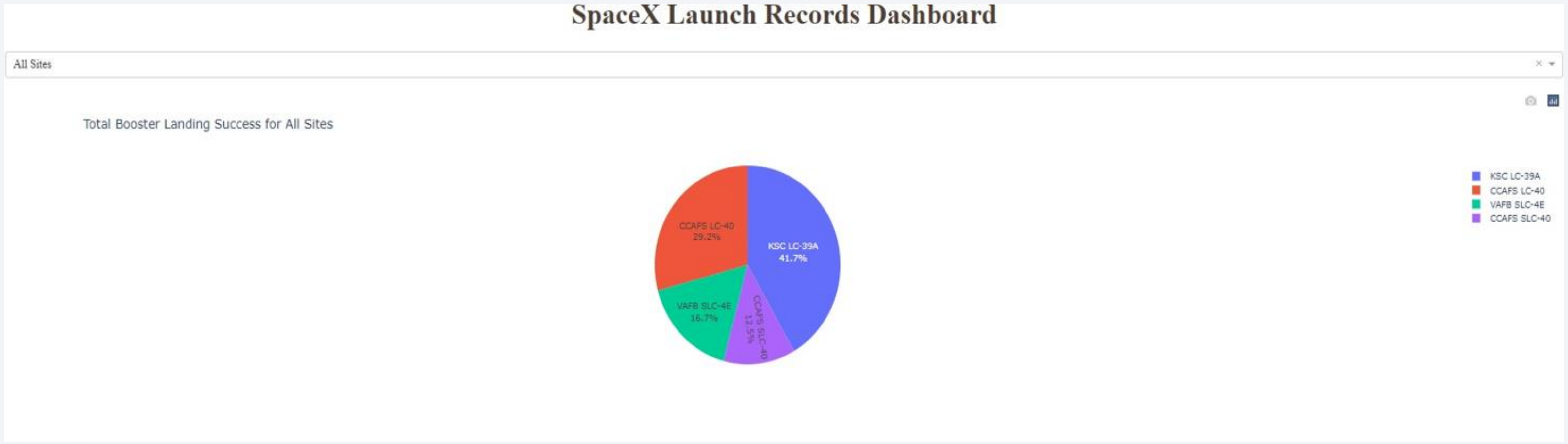
Distance markers to Railway, Highway and Coast.



Section 4

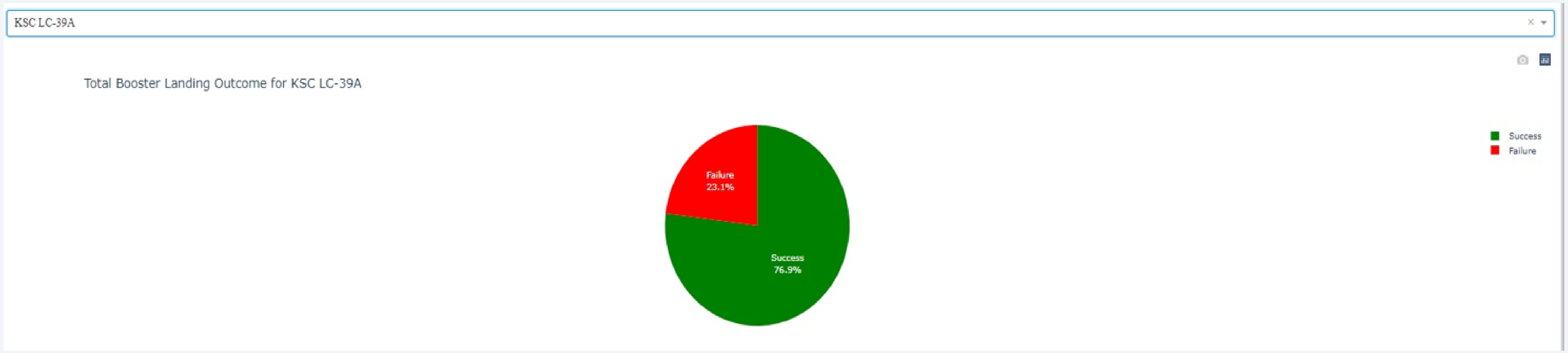
Build a Dashboard with Plotly Dash

Launch success for all sites



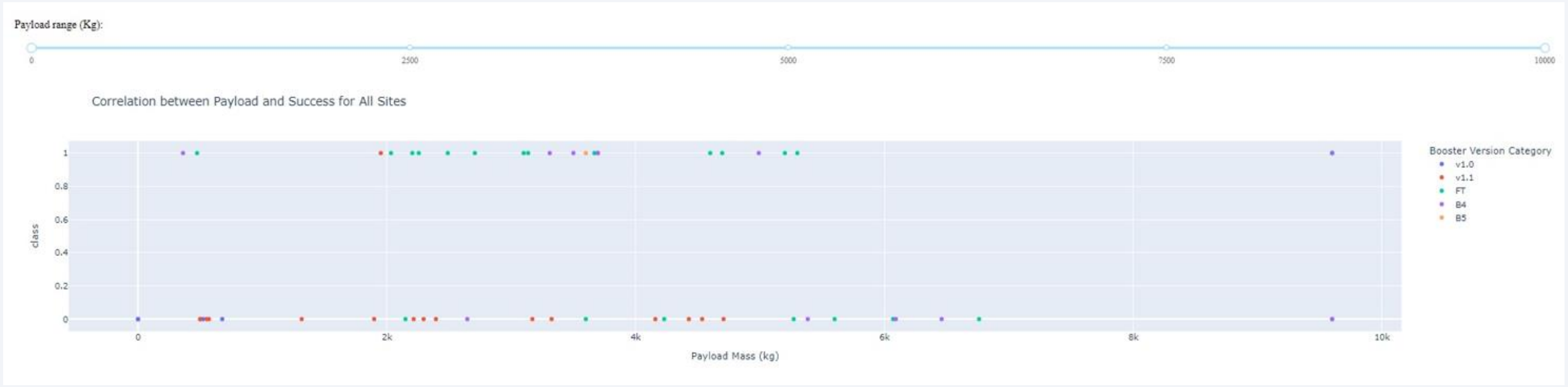
The size of the wedge is the percentage of successful booster landings at that launch site.

Launch outcomes at KSC LC-39A


KSC LC-39A

Select KSC LC-39A to show percentage of successes and failures.

Booster outcome for various payloads

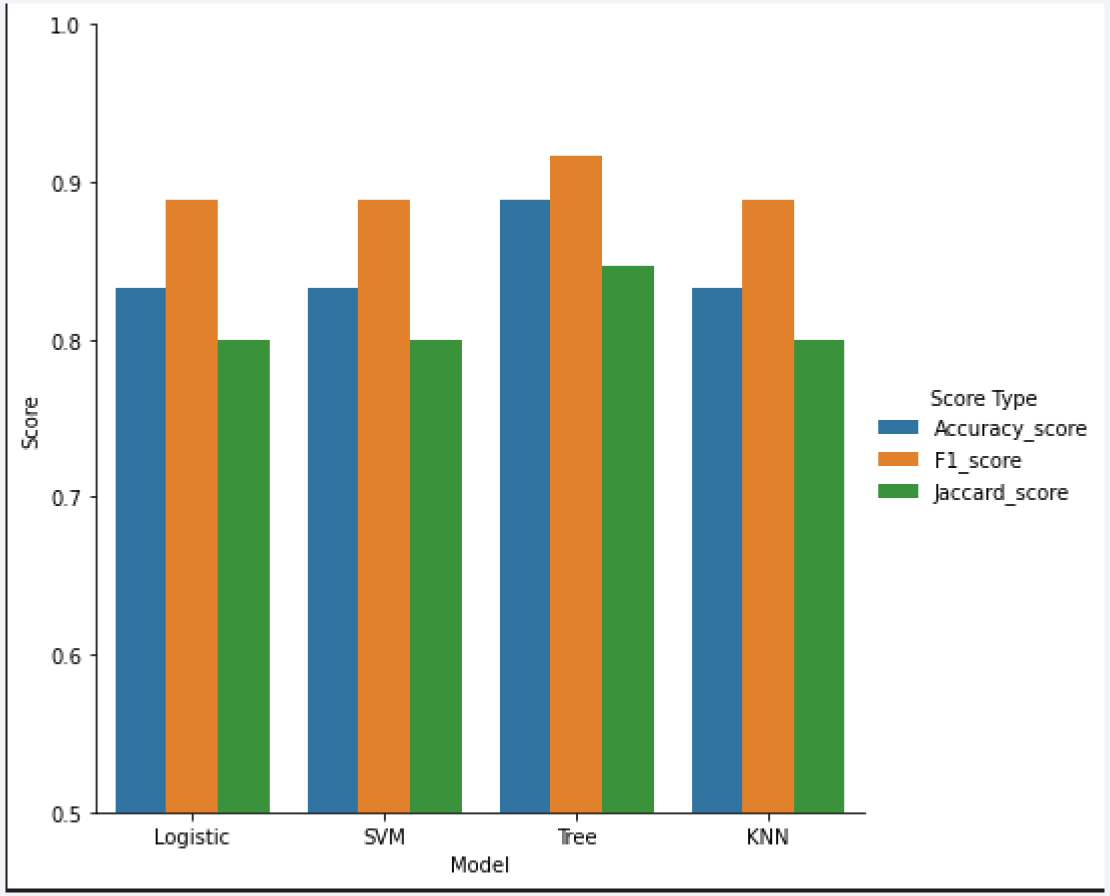


This scatter plot shows booster landing outcome for various payloads (Class 1 = Success). The marker colors identify the booster version so we can visually identify that version FT is the most successful.

Section 5

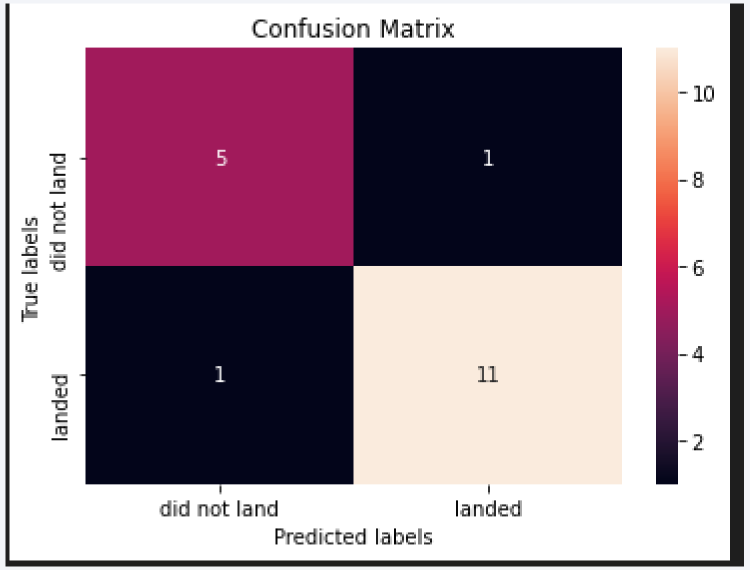
Predictive Analysis (Classification)

Classification Scores



Score the models using Accuracy, F1 and Jaccard.
The Tree model has the best scores.

Confusion Matrix for Tree model



The Tree model did very well in predicting both True Positive and True Negative. Only two cases were incorrectly predicted. This visually explains the high scores on the previous slide.

Conclusions

- The large number of booster landing outcomes (both success and failure) provided a rich data set to build prediction models.
- The Tree model performed slightly better than the other models, but all of them had accuracies above 80%. This means the features selected for the models are a good indicator of outcome.
- SpaceX has learned from experience how to successfully land boosters better than 85% of the time.
- There is a high barrier for a new launch service company to enter the market because of the time and the cost of launches to gain experience.

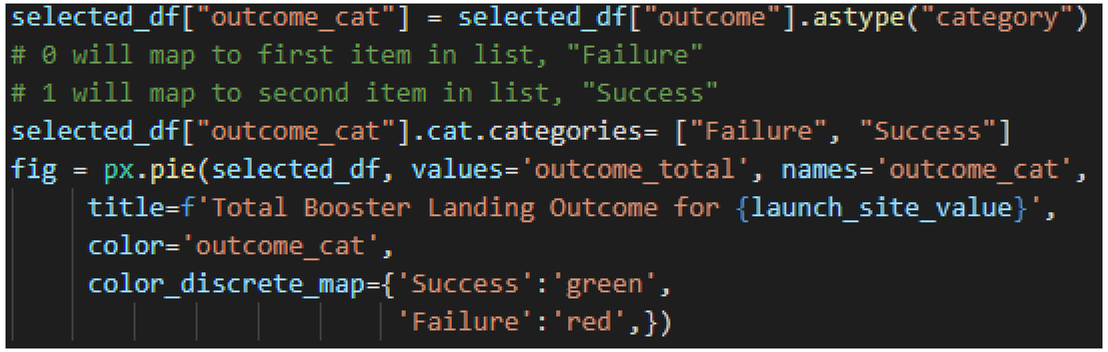
Appendix

- I used VS Code to develop and test code. A very good integrated environment that has helpful tooltips and notebook debugging.
- Pandas melt method is a flexible way to rearrange columns

```
score_df = pd.DataFrame({'Model': [model_name for model_name in model_list],
                            'Accuracy_score':[accscore for accscore in Accuracy_scores_list],
                           'F1_score':[f1score for f1score in F1_scores_list],
                            'Jaccard_score' : [jscore for jscore in Jaccard_scores_list]})
# Put all the scores into one column and corresponding score types in another column.
# This format is needed for the plot style
score_df = pd.melt(score_df, id_vars=['Model’],value_vars=['Accuracy_score’,
'F1_score','Jaccard_score'])
score_df.rename(columns={'variable':'Score Type','value':'Score'}, inplace=True)
```

Appendix

- For pie chart dashboard, added a category column to data to provide more meaningful labels of Success/Failure instead of 1/0. Also, specified the colors to be green for Success and red for Failure rather than default colors.



```
selected_df["outcome_cat"] = selected_df["outcome"].astype("category")
# 0 will map to first item in list, "Failure"
# 1 will map to second item in list, "Success"
selected_df["outcome_cat"].cat.categories= ["Failure", "Success"]
fig = px.pie(selected_df, values='outcome_total', names='outcome_cat',
        title=f'Total Booster Landing Outcome for {launch_site_value}',
        color='outcome_cat',
        color_discrete_map={'Success':'green',
                            'Failure':'red',})
```

Thank you!