Winning Space Race with Data Science

James Collier
01/10/2025

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection via API & Webscrpaing
  - Data Wrangling & Cleaning
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Dashboarding
  - Interactive Visual Analysis with maps
  - Machine Learning Prediction
- Summary of all results

Introduction

- SpaceX is one of the largest private companies in the space industry. They currently offer Falcon 9 rocket launches on their website with a cost of 62 Million dollars. Other providers cost upwards of 165 Million dollars each which is almost 300% of the cost of a SpaceX flights.
- SpaceX can offer these lower prices due to cost savings from reusing the first stage of the launch by having it successfully land
- Due to this, you can determine the cost of a rocket launch based on whether the first stage will land in a reusable state
- Problem:
  - Identifying key factors in whether the first stage will land or not
  - The relationship between the variables and how they affect the outcome and each other
  - Where & how we can get the relevant data

Section 1

Methodology

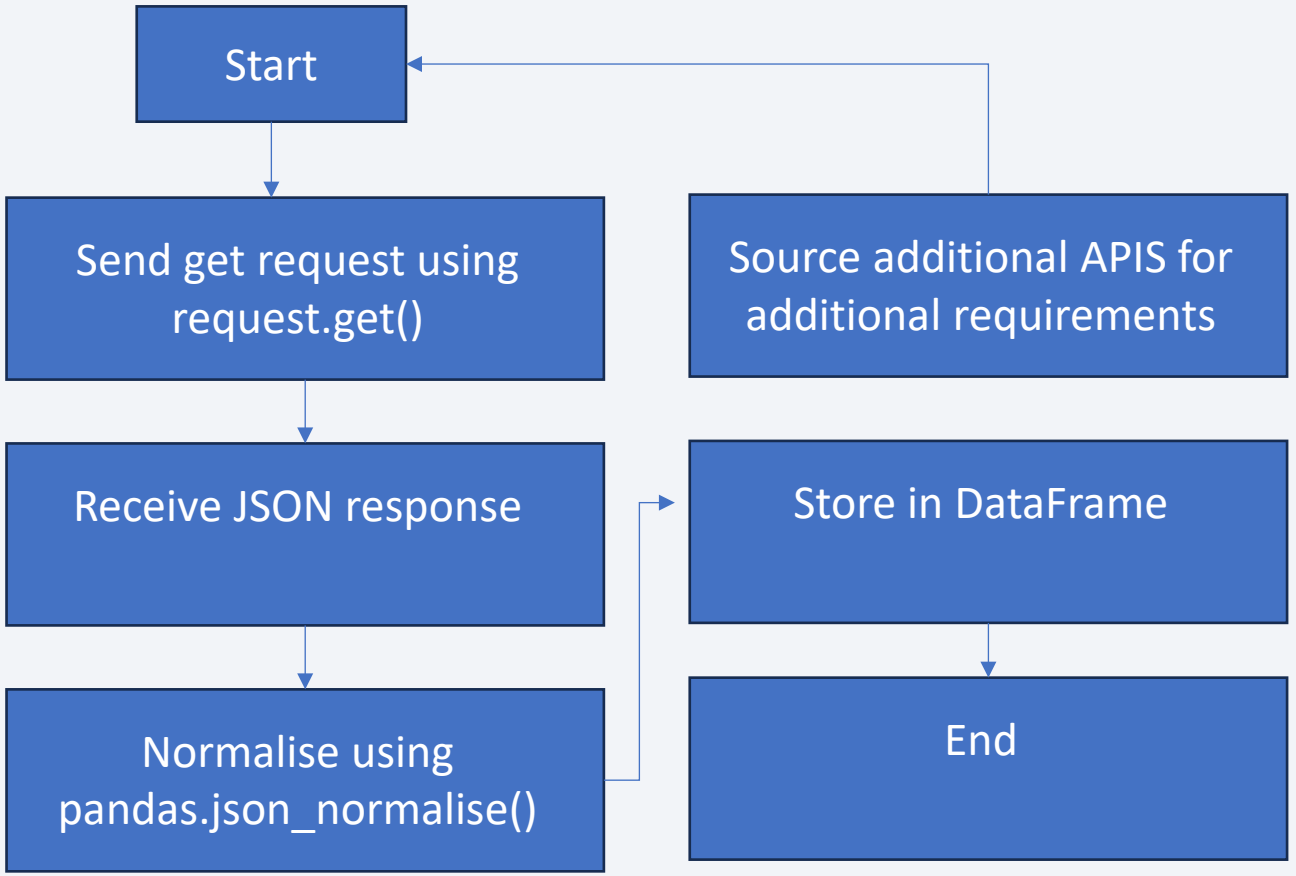
Methodology

Executive Summary

- Data collection methodology:
  - Via SpaceX REST API & Web scrapping from a website
- Perform data wrangling
  - Data was cleaned and pre-processed using classes & one hot encoding (and other standardization methods)
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models – 4 type of models with various parameters were evaluated

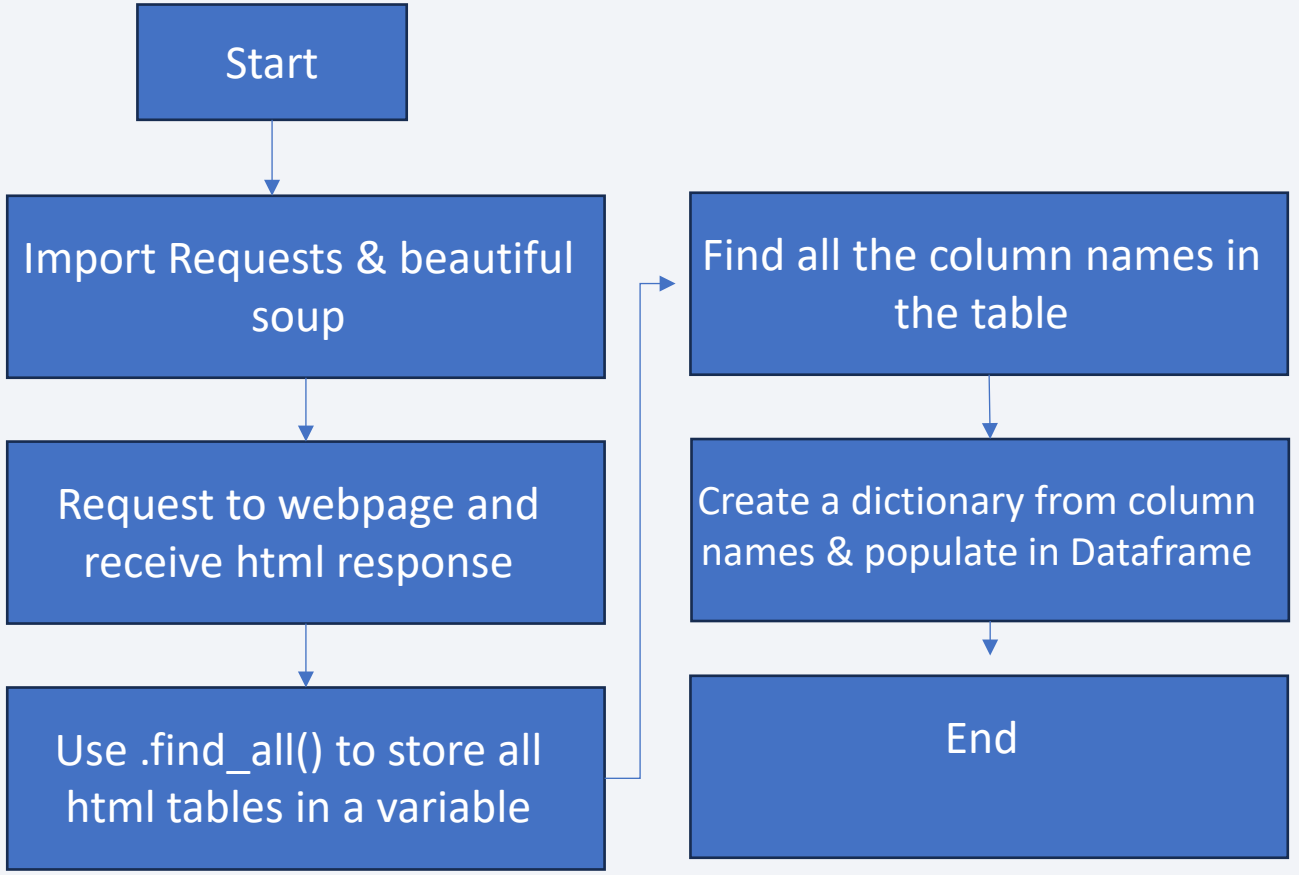
Data Collection – SpaceX API

- Collected data via a SpaceX API & performed initial data cleaning using requests, pandas and numpy
- Primary API Url used https://api.spacexdata.com/v4/launches/past
- Store the request response in a dataframe in json format using "json_normalise()"
- GitHub URL for the Data collection via API: https://github.com/James-Collier94/Applied_Data_Science_Capstone_Project/blob/main/jupyter-labs-spacex-data-collection-api_Complete.ipynb



Data Collection - Scraping

- Collected data via a website & performed initial data cleaning using requests, & beautiful Soup
- Found relevant columns using html tags such as th
- GITHub URL for Data Collection via Webscraping: https://github.com/James-Collier94/Applied_Data_Science_Capstone_Project/blob/main/jupyter-labs-webscraping%20Complete.ipynb



Data Wrangling

- Imported Pandas & Numpy for data wrangling purposes
- Imported .csv created previously into a pandas data frame
- Checked the count of data for each launch site, and each type of orbit launched
- Counted the mission outcome of different types
- Created a landing class to determine if it was successful 1, or unsuccessful 0 so we could then calculate a mean
- 66% of launches landed successfully
- GITHUB Link for Data Wrangling: https://github.com/James-Collier94/Applied_Data_Science_Capstone_Project/blob/main/labs-jupyter-spacex-Data%20wrangling%20Complete.ipynb

EDA with Data Visualization

- Started off by using scatter charts so we could easily visualize the relation ship between different variables (using the previously defined success/fail class)
    - Payload mass & Flight number
    - Flight Number & Launch Site
    - Payload Mass & Launch Site
    - Flight Number & Orbit type
    - Payload Mass & Orbit Type
- Also used bar charts to visualize the success rate of each Orbit type
- A line chart was used to visualize success rate over time
- During the data visualization we also used feature engineering to perform one hot encoding object data
- Github link for EDA Notebook: https://github.com/James-Collier94/Applied_Data_Science_Capstone_Project/blob/main/edadataviz%20Complete.ipynb

EDA with SQL

- I used SQL to gain a better understanding of the dataset and performed the following queries
  - Displaying a list of unique launch sites
  - Displaying figure records from launch sites beginning with CCA
  - Showing the total pay load mass launched by NASA (CRS) site
  - Calculating the average payload mass from booster version f9 v1.1
  - Retrieving the first date a successful landing took place
  - Displaying a list of unique boosters than have succeeded in Drone Ship landings with payload masses between 4000 & 6000
  - Showing the total number of success & failure outcomes
  - Displaying a unique list of booster versions that have carried the maximum payload mass
  - Retrieving the months with failure outcomes on drone ships, & the booster version used
  - Ranking the failure outcomes based on the “Landing_Outcome” between 2010/06/04 & 2017/03/20
- GITHUB Link for the notebook: https://github.com/James-Collier94/Applied_Data_Science_Capstone_Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20Complete.ipynb

Build an Interactive Map with Folium

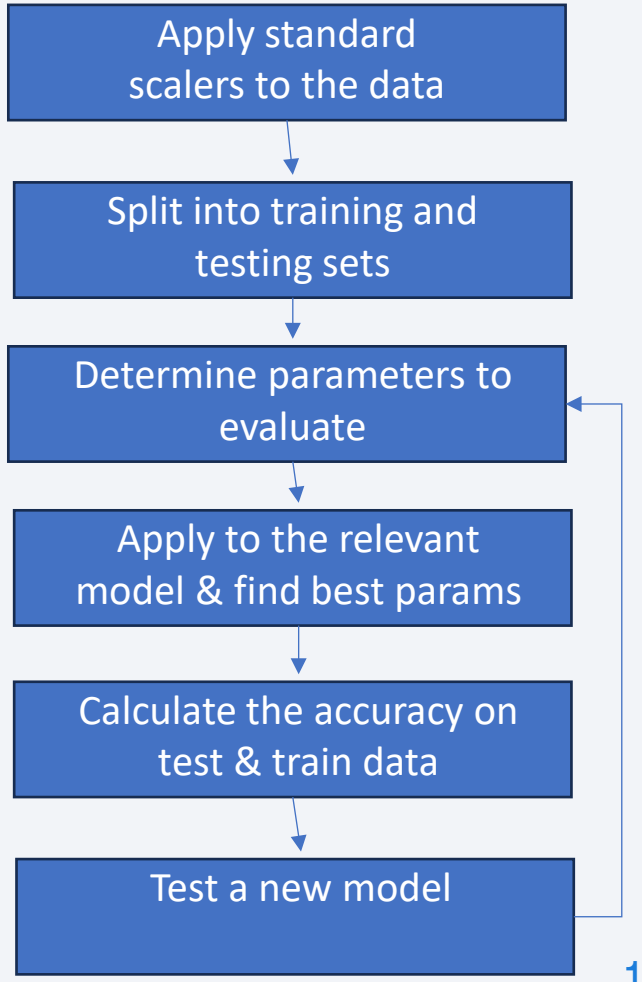
- I started by adding a marker to all launch sites on a map, using the relevant latitude and longitude.
- I then marked all the success/failed launches for each site on the map. Naturally this was quite crowded
- To make it less crowded I created a marker cluster and added each success/failure marker inside a cluster
- The map was then interactive so you could click on a cluster, zoom in and see additional details about the individual launches
- Red markers were launches with class 0 (failures) and green with class 1 (success)
- We then calculated the distance to other landmarks by creating new markers & running a function. We did this on:
  - Cities
  - Coast Lines
  - Train stations
- Github link to completed workbook: https://github.com/James-Collier94/Applied_Data_Science_Capstone_Project/blob/main/lab_jupyter_launch_site_location_complete.ipynb

Build a Dashboard with Plotly Dash

- I created an interactive dashboard with:
  - A pie chart showing total successful launch percentage per site
  - A scatter chart showing Payload Mass range for each booster version category & if it succeeded or not.
  - You could select the site, and the pie chart would update showing the percentage of successful and failed launches at that site
  - You could also adjust the payload mass range to show a more specific scatter chart (which was also filtered by the site selection)
- These plots are very useful in seeing which sites had good success rates and contribute to the majority of the successful laucnes.
- The payload mass scatter graph is very useful in showing which booster version are most successful and how the mass affects sucess
- Dashly python code link: https://github.com/James-Collier94/Applied_Data_Science_Capstone_Project/blob/main/spacex-dash-app.py

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model
- Building the Model
  - Import relevant modules, numpy, pandas, sklearn
  - Import the data into a dataframe
  - Collect the “class” column in a separate dataframe
  - Applied standard scalers to the full data frame
  - Split the data frame into training and testing sets
  - Apply training data to the relevant model
- Evaluating the Model
  - Applied the model with multiple parameters using the GridSearchCV()
  - Fitted the data to the model and calculated the best parameters using .best_params_
- Improving the model
  - This was done by applying the above steps to different types of models/machines
- Finding the best performing classification model
  - All models were reviewed using a confusion matrix, and the accuracy on both the test set & training data



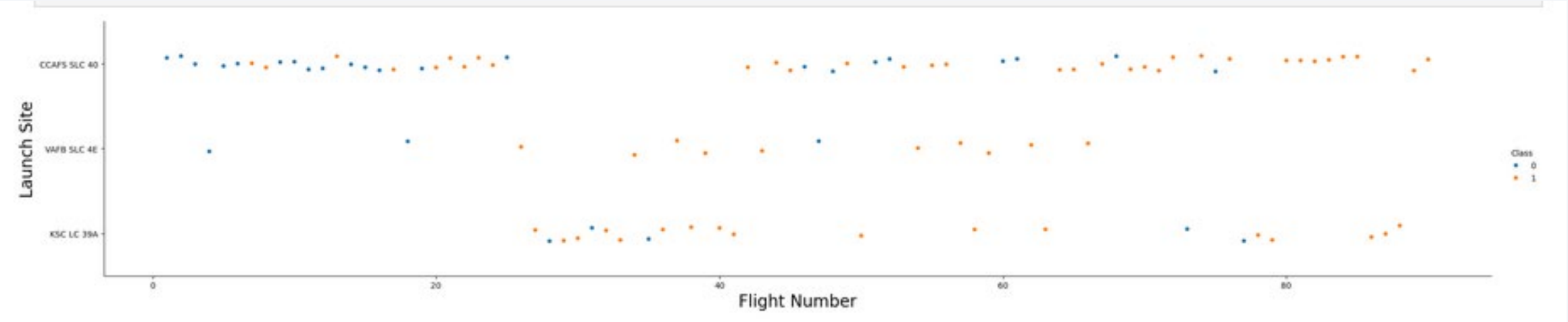
Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

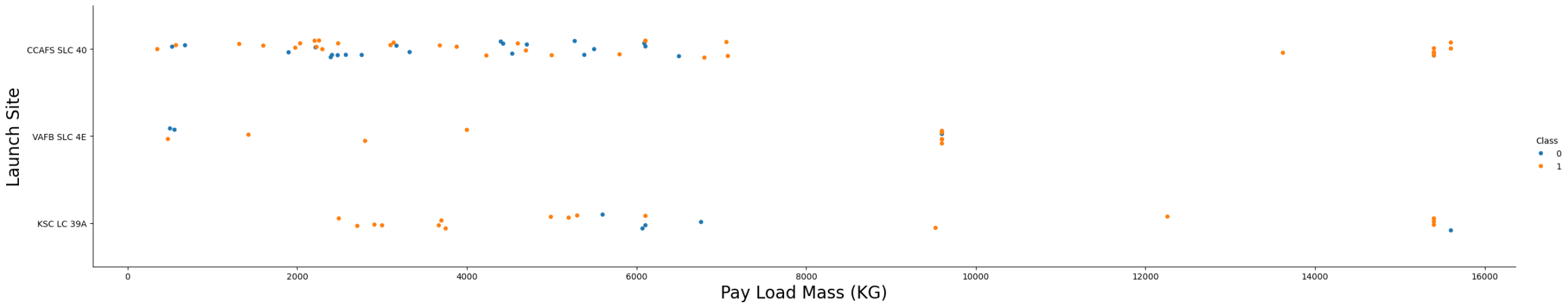
Insights drawn from EDA

Flight Number vs. Launch Site



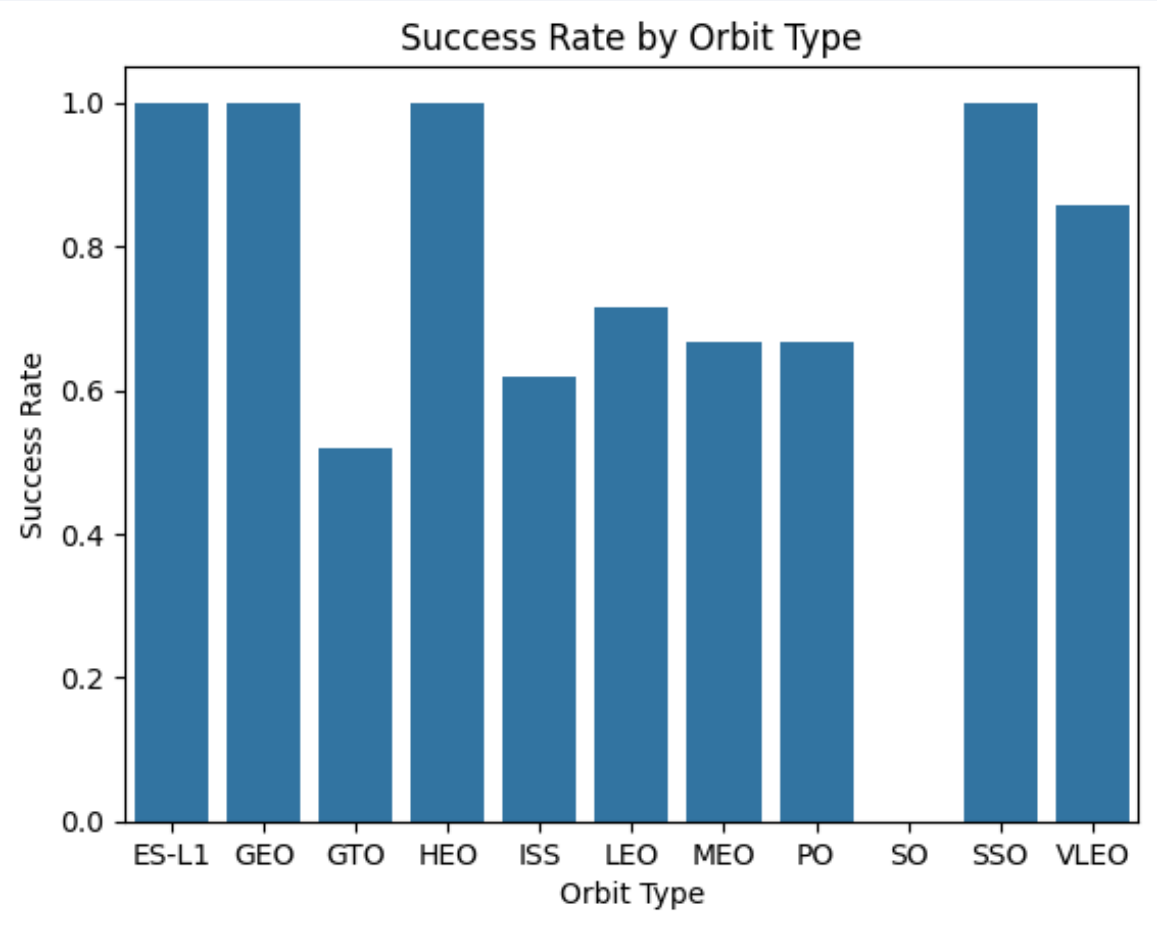
As with the first plot we can still see that flight number is a factor and the later flight numbers are more likely to suceed. The launch sites also seem to be a factor with CCAFS SLC 40 being more likely to have a failed landing, however the site itself seems to be approving with late flight numebrs

Payload vs. Launch Site



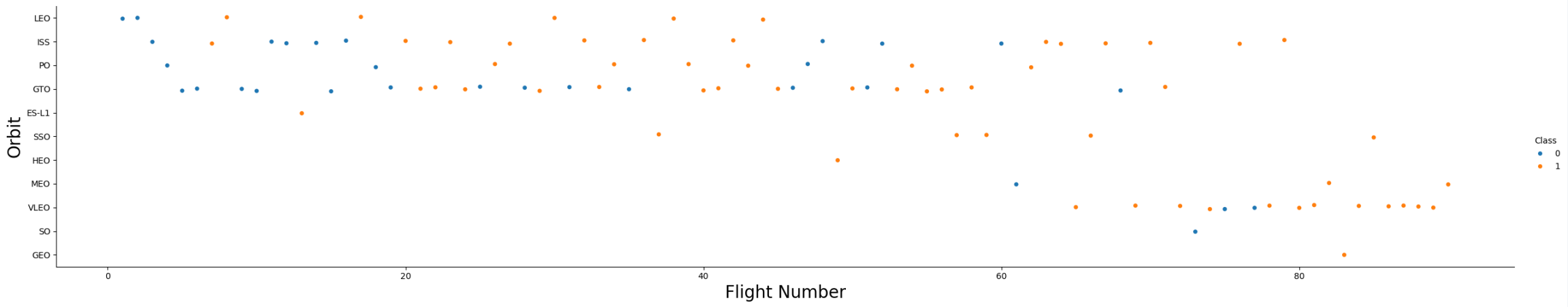
Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type



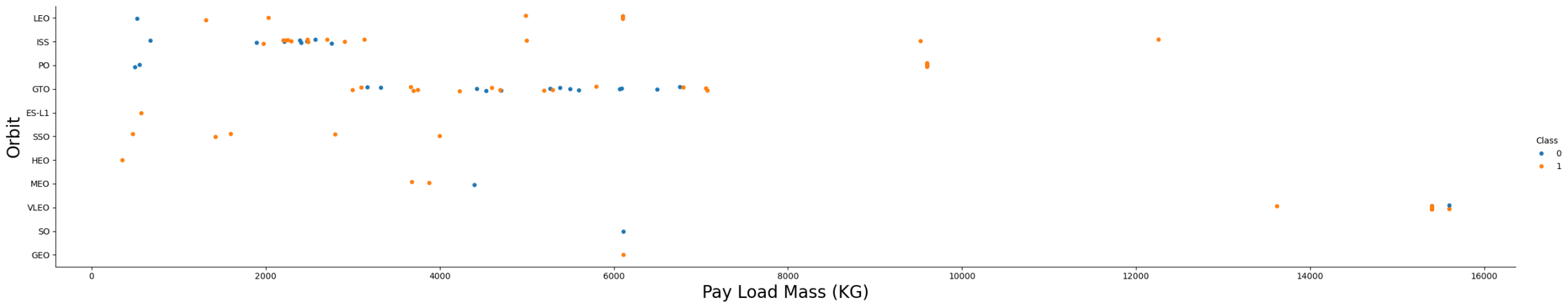
- Certain orbit types have a guaranteed success rate based on the data, such as ES-L1, GEO, HEO and SS
- The bar chart shows a pretty obvious potential relationship between orbit type and if phase 1 successfully lands
- This should be taken with a grain of salt, as potentially the 100% success or failure rates could be due to lack of data

Flight Number vs. Orbit Type



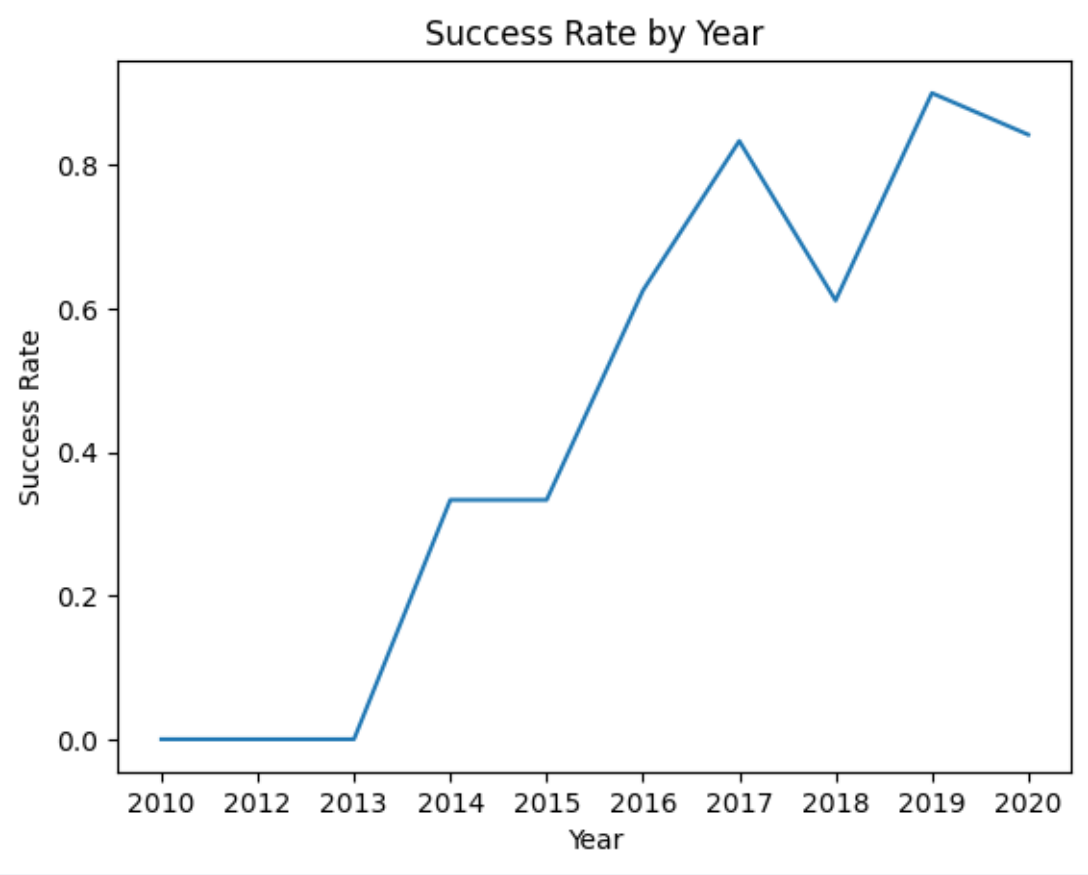
You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend



From the graph it is very clear that the success rate keeps increasing overtime, with significant jumps between 2015-2017 perhaps due to technological enhancements.

This increase can be seen to carry on to 2020

All Launch Site Names

```
%sql select distinct Launch_Site from SPACEXTBL
```

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

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

Distinct was used to only show unique launch sites

Launch Site Names Begin with 'CCA'

- Like was used to find launch sites with a specific string in it
- % was used as a wildcard to show any text could come after cca

```
%sql select * from SPACEXTBL where Launch_Site like "CCA%" limit 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (paracl
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 (paracl
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No att
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No att
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No att

Total Payload Mass

- Sum was used to calculate the total payload mass
- Where was used to restrict to purely one customer

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer = "NASA (CRS)"
```

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

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- Avg was used to calculate the mean of payload mass
- Where was used to select the booster version

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version = "F9 v1.1"
```

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

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- Min was used on the date function to find the oldest (first) date
- Where was used to specify the landing outcome we wanted

```
%sql select min(Date) from SPACEXTBL where Landing_Outcome = "Success (ground pad)"
```

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

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
%sql select distinct Booster_Version from SPACEXTBL where Landing_Outcome = "Success (drone ship)" and PAYLOAD_MASS__KG_ be
```

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

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

- Distinct is used to select unique values from the booster version
- Where is used to to specify the landing outcome, and “and” is used to specify a type of payload mass
- Between is used to specify which results I want to return for the payload mass
- %sql select distinct Booster_Version from SPACEXTBL where Landing_Outcome = "Success (drone ship)" and PAYLOAD_MASS__KG_ between 4000 and 6000

Total Number of Successful and Failure Mission Outcomes

- Successful missions displayer by using the like function and searching for success

```
%sql select count(Mission_Outcome) from SPACEXTBL where Mission_Outcome like "Success%"
```

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

count(Mission_Outcome)
100

Boosters Carried Maximum Payload

- Distinct is used to return unique values from the booster version only
- Where is used to specify the type of payload mass I want
- The subquery is used in conjunction with the where function to specify I only want the maximum payload

```
%sql select distinct Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

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

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

2015 Launch Records

- %sql select substr(Date, 6, 2) as month, Booster_Version, Launch_Site, Landing_Outcome from SPACEXTBL where substr(Date, 0, 5) = "2015" and Landing_Outcome = "Failure (drone ship)"

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

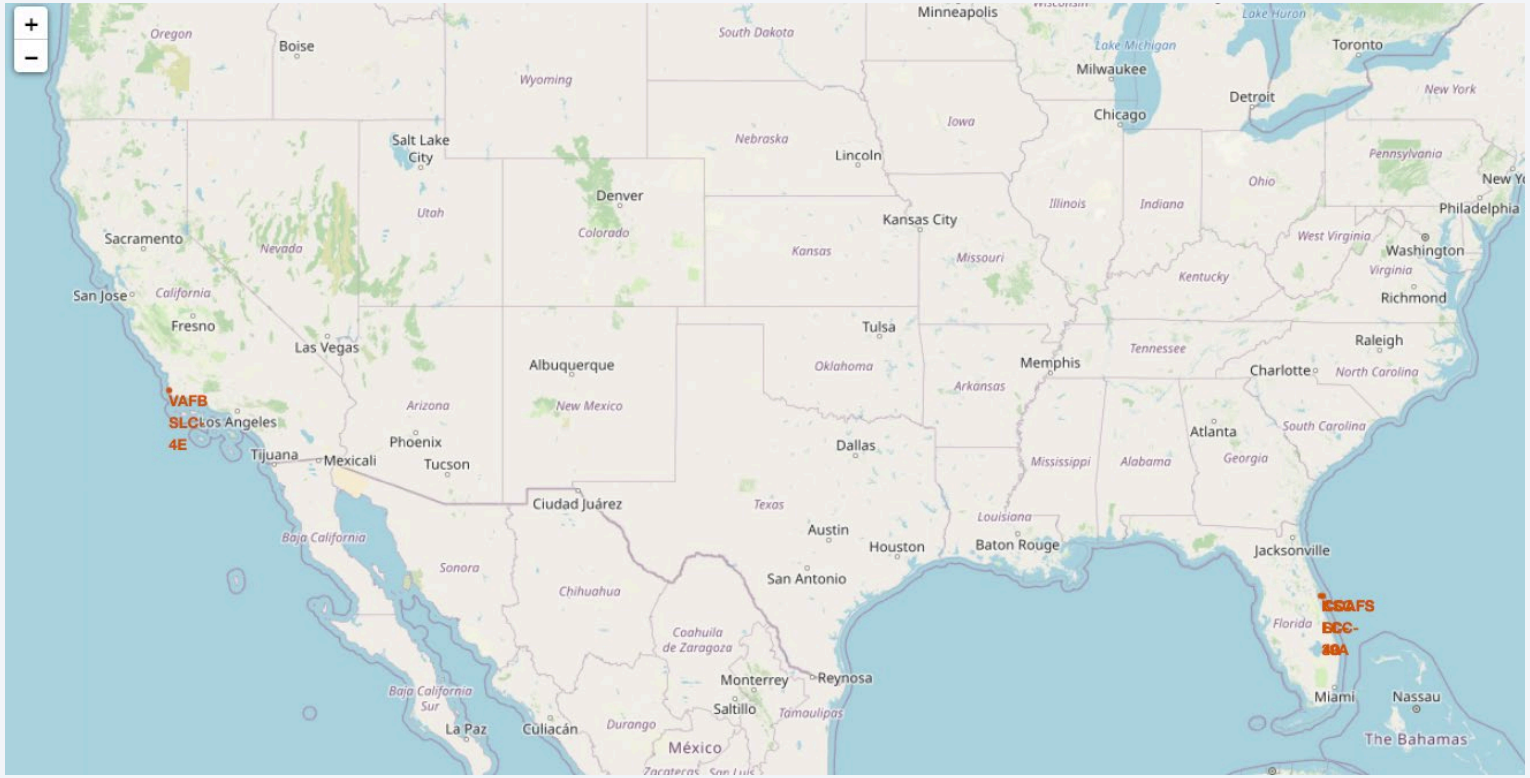
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- %sql select Landing_Outcome, count(*) as Outcome_Count from SPACEXTBL where Date Between "2010-06-04" and "2017-03-20" group by Landing_Outcome Order by Outcome_Count desc
- Created a new column using the as function
- Where function used to determine the dates I was interested in
- Grouped by the landing outcome and then ordered by the count with highest first

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

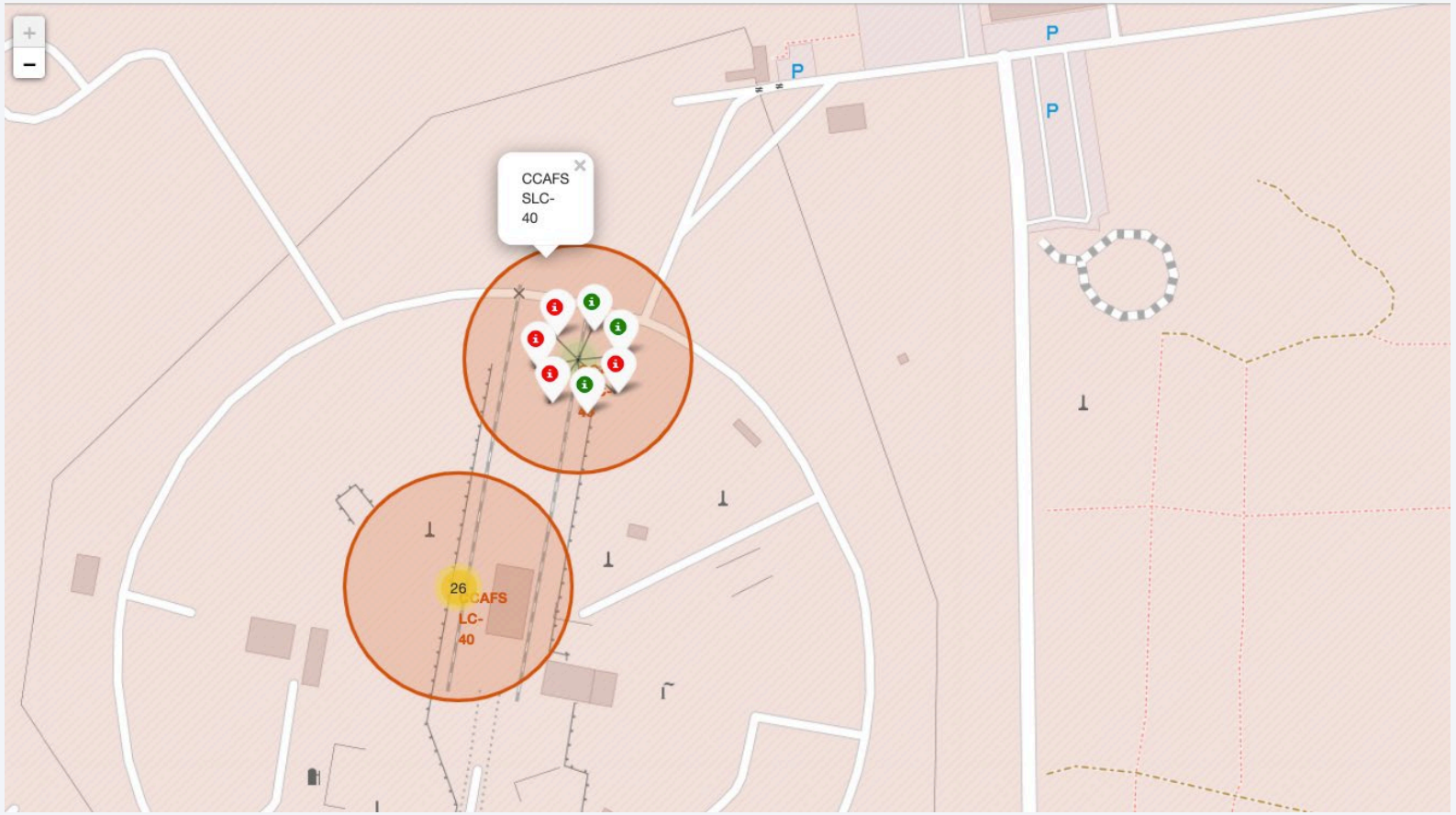

Section 3
Launch Sites
Proximities Analysis

Launch Site Location



All launch sites are situated on the south of the US, and near the coast.

Launch Markers – With Success or Failures



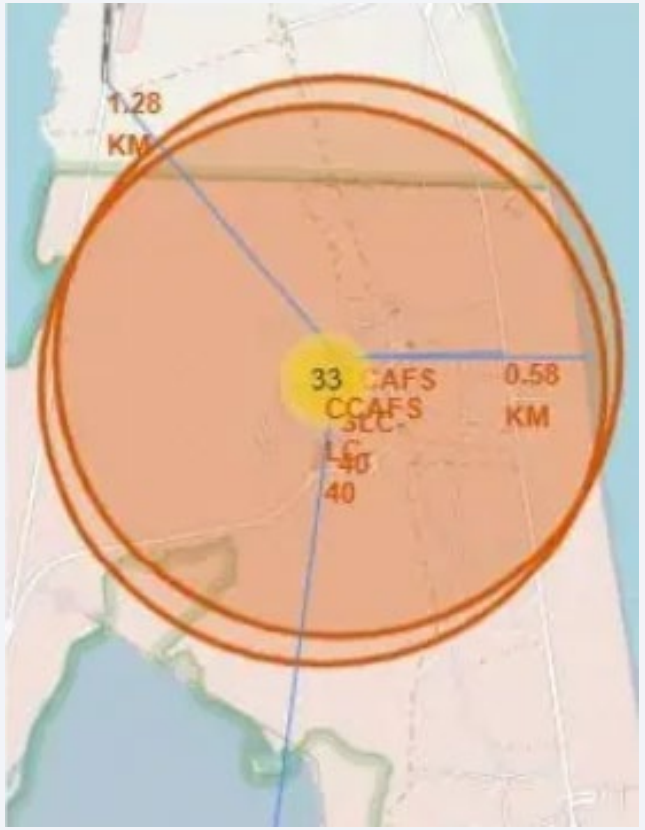
Quite a significant number of launches at each different launch site

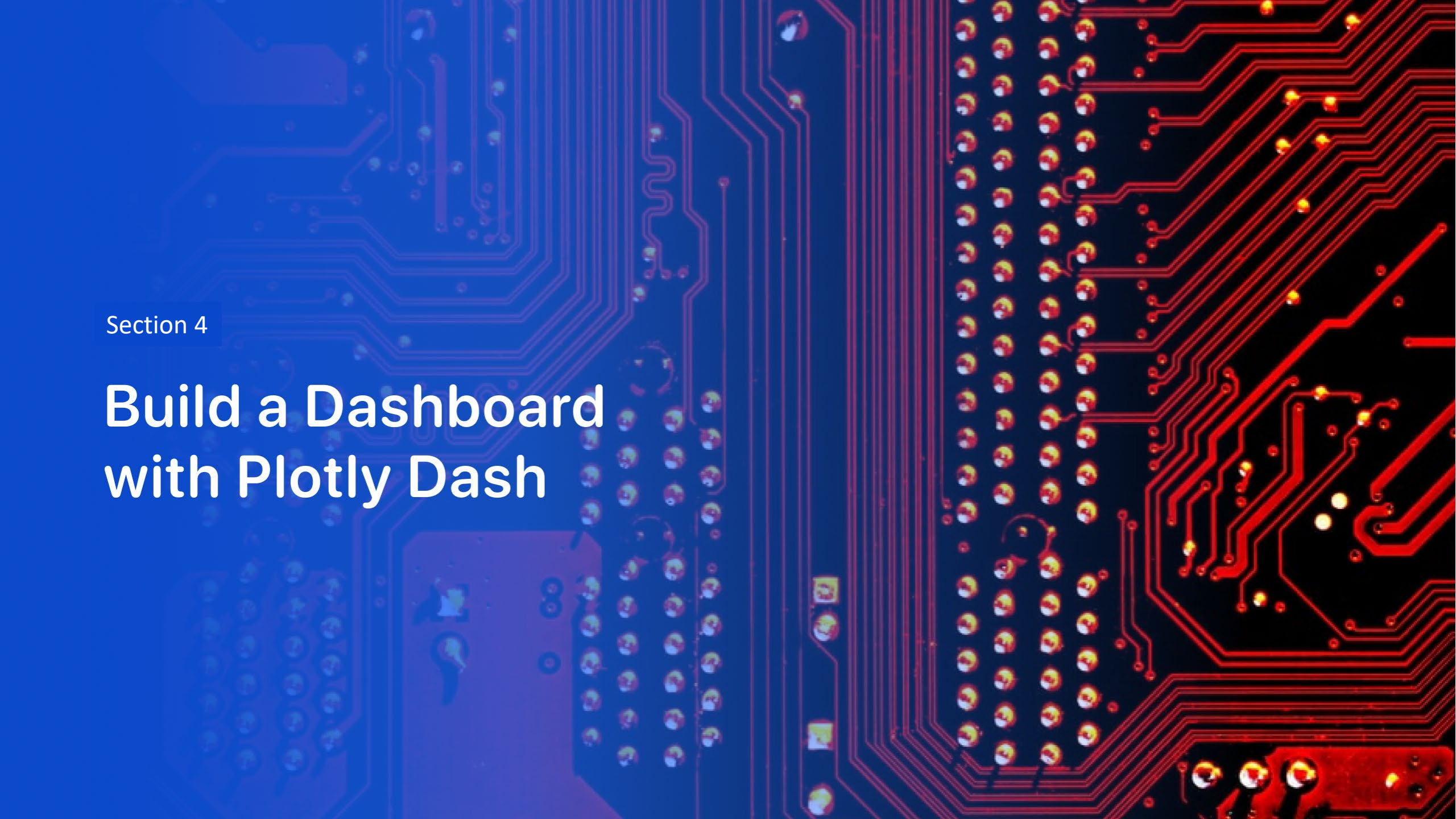
Map is easy to use to specify where success rate is higher

Launches all occur in roughly the same location

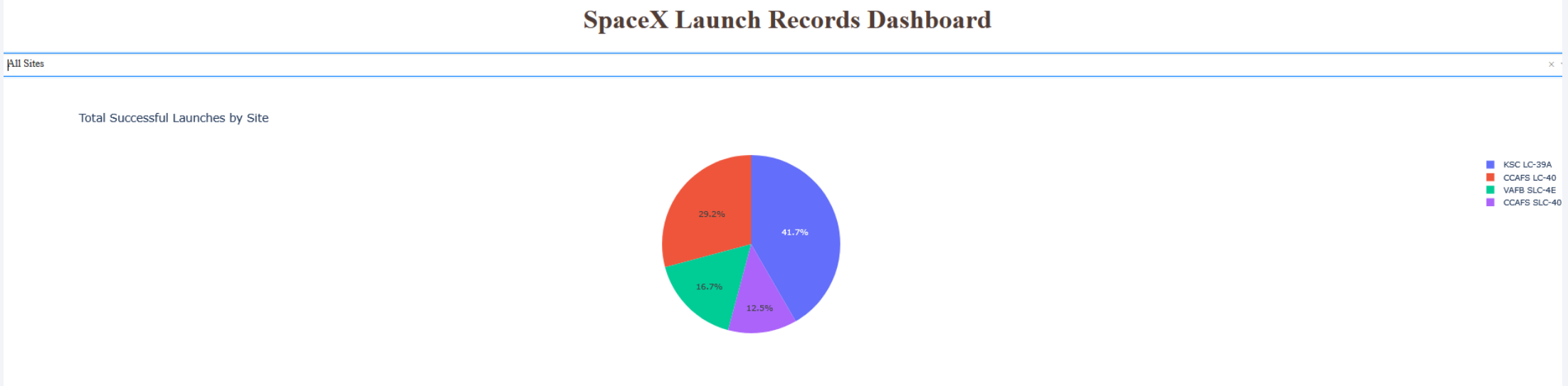
Distance from Coast & Trainlines etc

- Close to the coast line for specific types of successful and failed landings
- Near highways & trainlines, likely for logistic purposes
- Not close to significantly populated areas



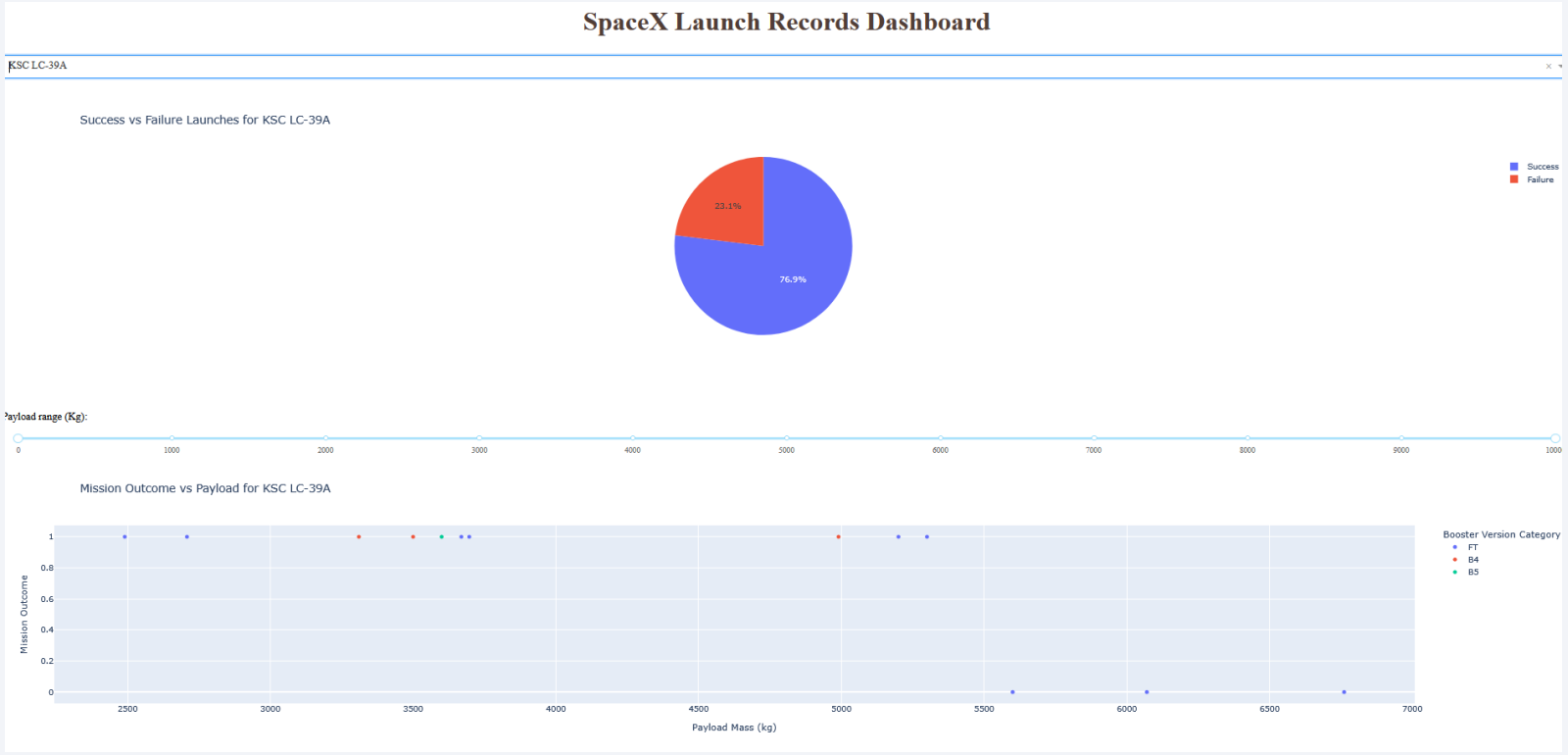

Section 4
Build a Dashboard with Plotly Dash

Successful Launches by Site



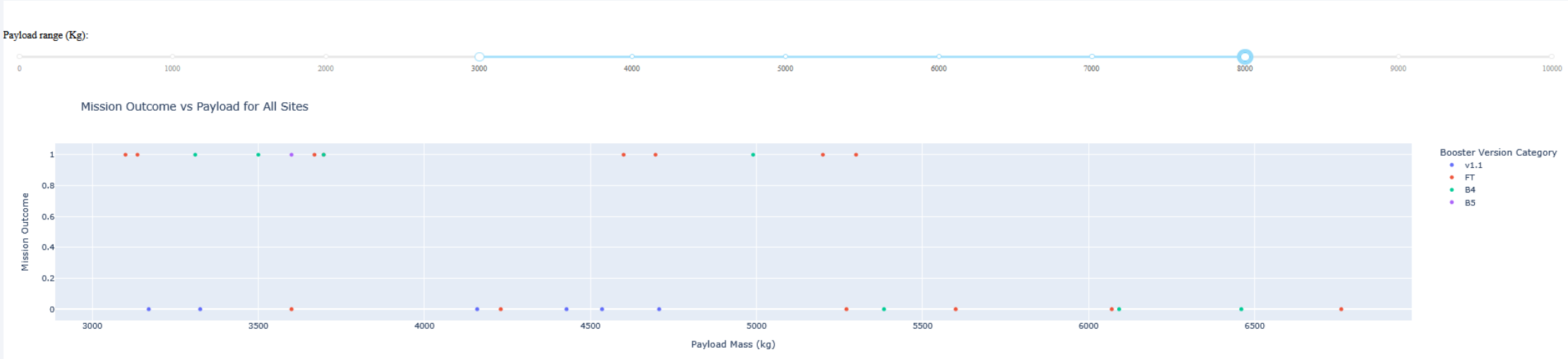
- The majority of successful launches come from sites KSC LC-39A & CCAFS LC-40
- These two sites make up roughly half of the overall launches but over 70% of the successful launches.
- This indicates the success rates at the remaining two sites is not particularly high.

Site with the Highest Success Ratio



- As you can see in the Pie Chart on the left, the launch site with the highest success ratio was KSC LC-39A
- The success rate was 76.9% which is not significantly high, however due to the nature of the launches & cost saving impact is still quite significant
- As you can see based on the scatter plot below, the failures at this site were primarily for higher Payload Masses, indicating success was much more likely for the lower payload masses

Payload vs Launch Outcome



- Payload range at first glance doesn't have a direct impact on success rate.
- However you can clearly see in the above that the success rate of Payload Masses over 5000 has a significantly worse rate
- You can also use the above to get a rough idea of which booster version performs the best, for example at lower Payload ranges the B4 is incredibly successful

Section 5

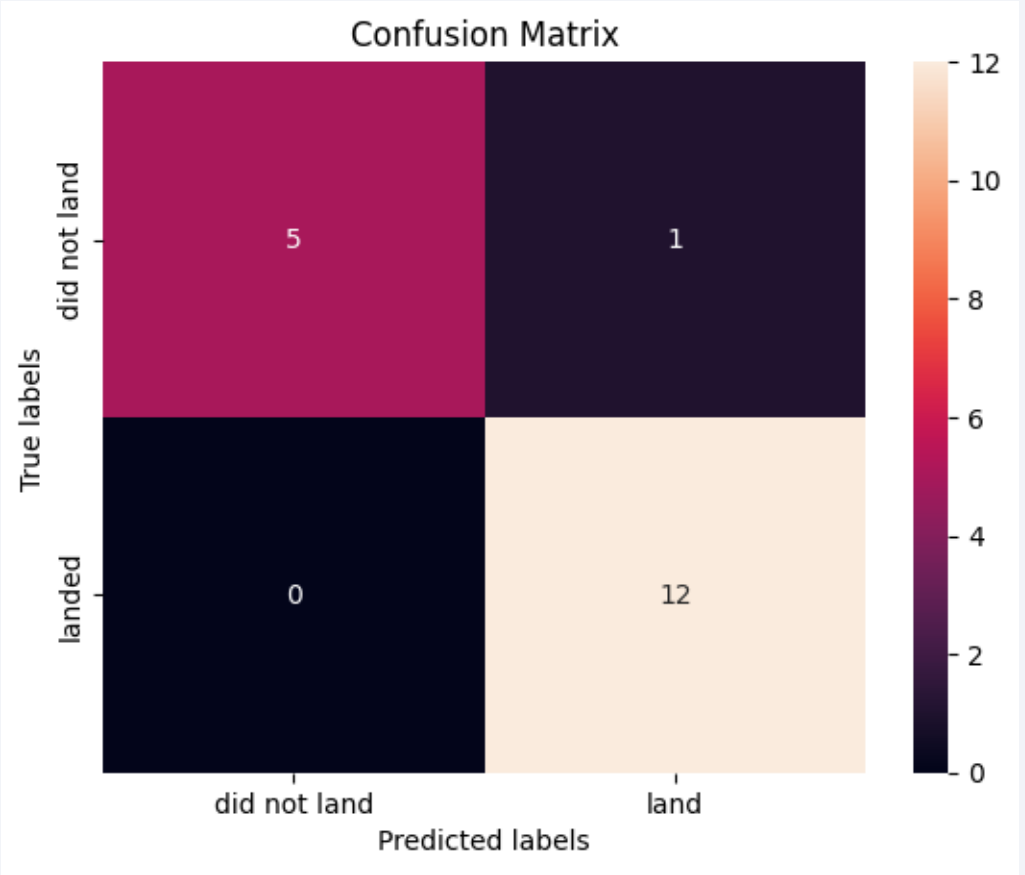
Predictive Analysis (Classification)

Classification Accuracy

- As you can see in the chart to the right, KNN had the best accuracy with 84% in the training data and 94% in the testing data

Confusion Matrix

- As you can see from the confusion matrix on the right, KNN showed a significant amount of true positives and true negatives with only one error (true negative)



Conclusions

- In conclusion:
  - Time has a big impact in the chances of a successful landing, which was clearly displayer in our earlier line chart showing an increase in success chance between 2013 & 2020
  - Lower payload ranges tend to have a higher success chance than the larger payload rangones
  - KSC LC-39A has the most successful launches of any site, as well as the highest percentage of success when compared to total launches
  - The KNN Algorithm was the best machine learning approach for this data set, it provided the highest accuracy with the best confusion matrix
  - The tree classifier algorithm was the worst showing multiple false positives and having an overall lower testing score

Appendix

- Github link for the project https://github.com/James-Collier94/Applied_Data_Science_Capstone_Project

- Error in given code in web scraping exercise
- ---
- AttributeError Traceback (most recent call last)
- /tmp/ipykernel_922/1669576390.py in <module>
- 68 # Customer
- 69 # TODO: Append the customer into launch_dict with key `Customer`
- ---> 70 customer = row[6].a.string
- 71 launch_dict['Customer'].append(customer)
- 72 print(customer)

- AttributeError: 'NoneType' object has no attribute 'string'

Confusion matrixs

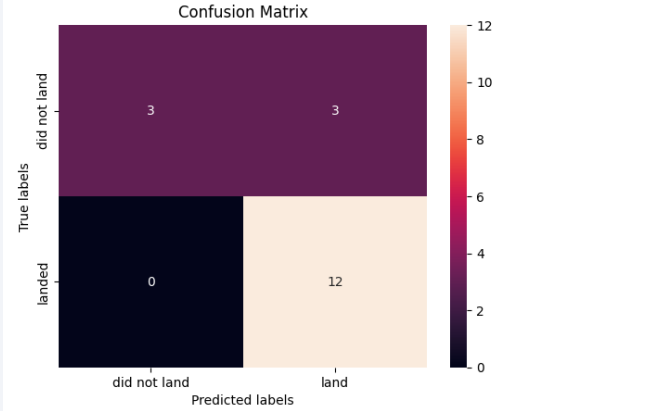
Calculate the accuracy of tree_cv on the test data using the method `score`:

```
tree_cv.score(X_test, Y_test)
```

```
0.8333333333333334
```

We can plot the confusion matrix

```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

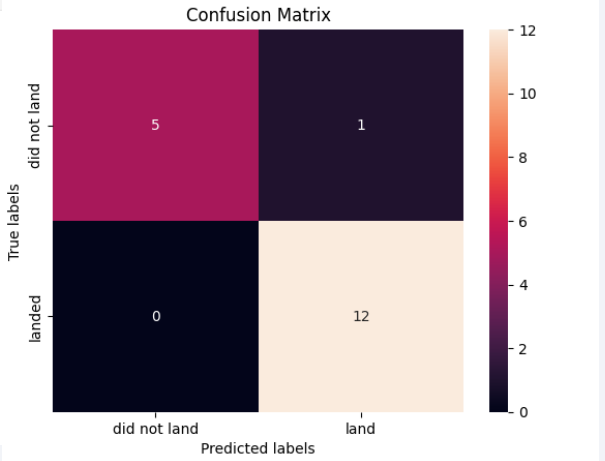



```
knn_cv.score(X_test, Y_test)
```

```
0.9444444444444444
```

We can plot the confusion matrix

```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

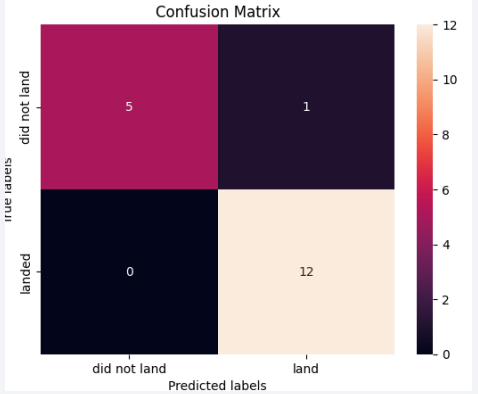



```
logreg_cv.score(X_test, Y_test)
```

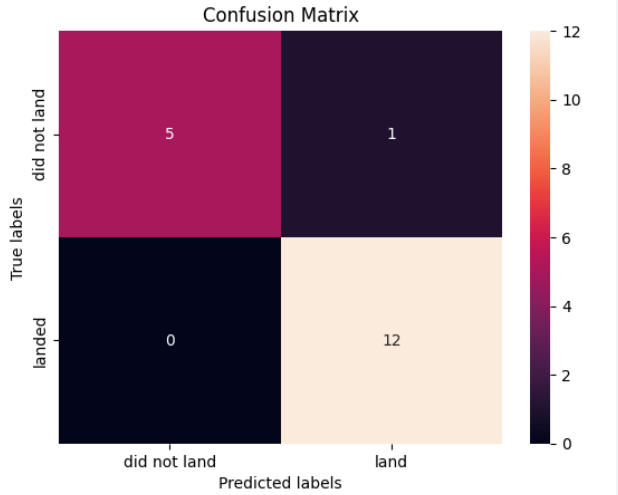
```
0.9444444444444444
```

Lets look at the confusion matrix:

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

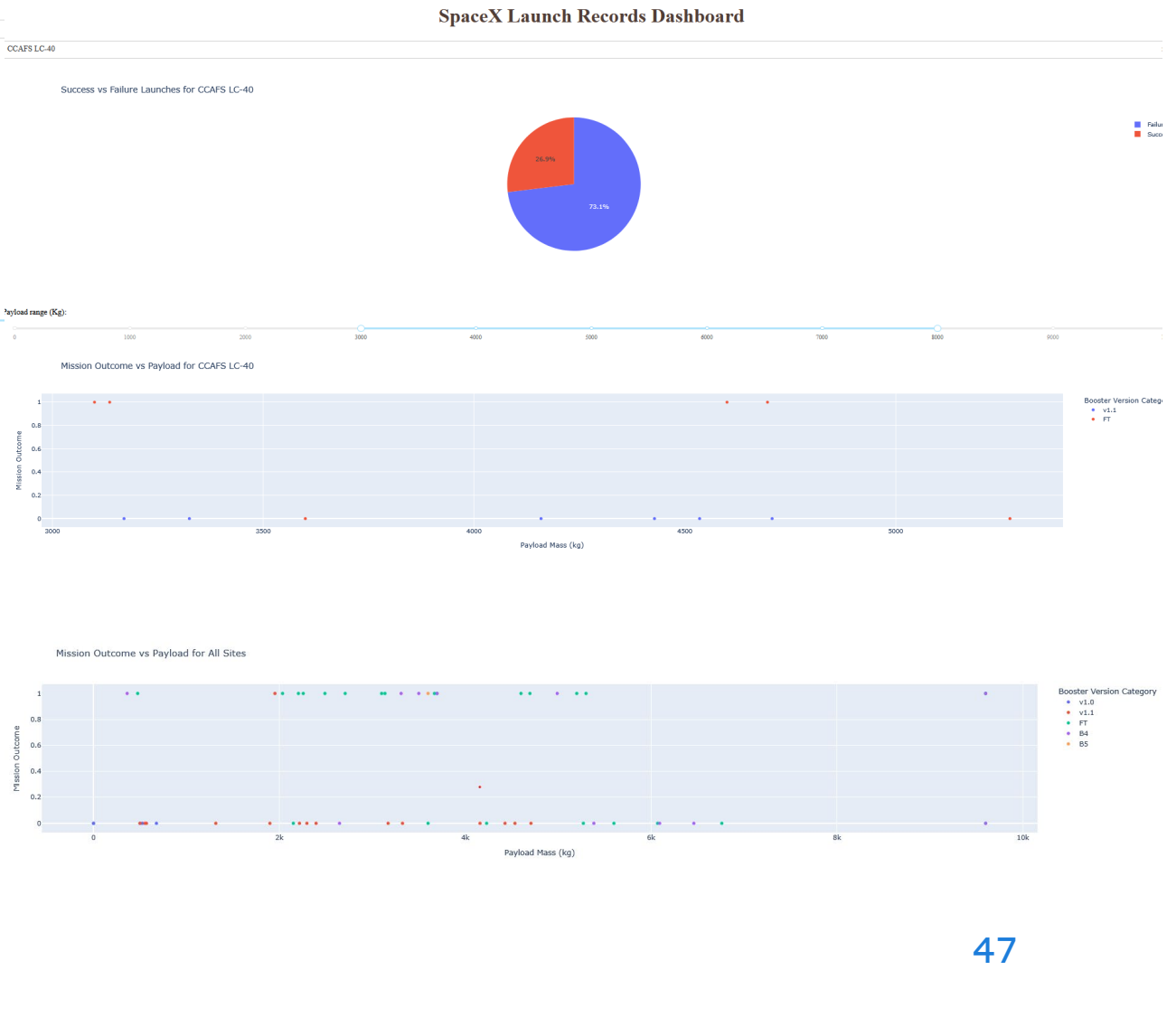



```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 8

Appendix – Dashboard Screenshots



Thank you!