IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jon Houston
Oct 6th 2024

Outline

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

Executive Summary

- We were tasked with predicting the landing success of the Falcon 9 first stage thruster.
- We used the following methods to achieve that:
- Collecting data through web scraping and the SpaceX API.
- Wrangling the data and exploring it using SQL and Matplotlib.
- Creating data visualizations and predicting machine learning algorithms.
- The result was a wealth of information represented through informative graphs and analytical results.

Introduction

- Project background and context:
  - Falcon 9 rockets are significantly cheaper than competitors, with the average one costing ~33% of what other manufacturers would need.
  - A big part of this cost reduction is the ability to land and re-use the first stage of the rocket.
  - By predicting whether the first stage will land or not we can predict the cost of a flight more accurately.
- Problems you want to find answers to:
  - What are the determining factors of a successful landing?
  - Is it possible to predict a landing with 100% accuracy?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - First we collected the data. We used both SpaceX's own API and a web-scraped Wikipedia table of the launch results.
- Perform data wrangling:
  - From there the relevant data was extracted and wrangled to prepare it for analysis in Python. One-hot encoding was applied to categorical features.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - A series of classification models were used and evaluated against one another to find which one produced the best results.

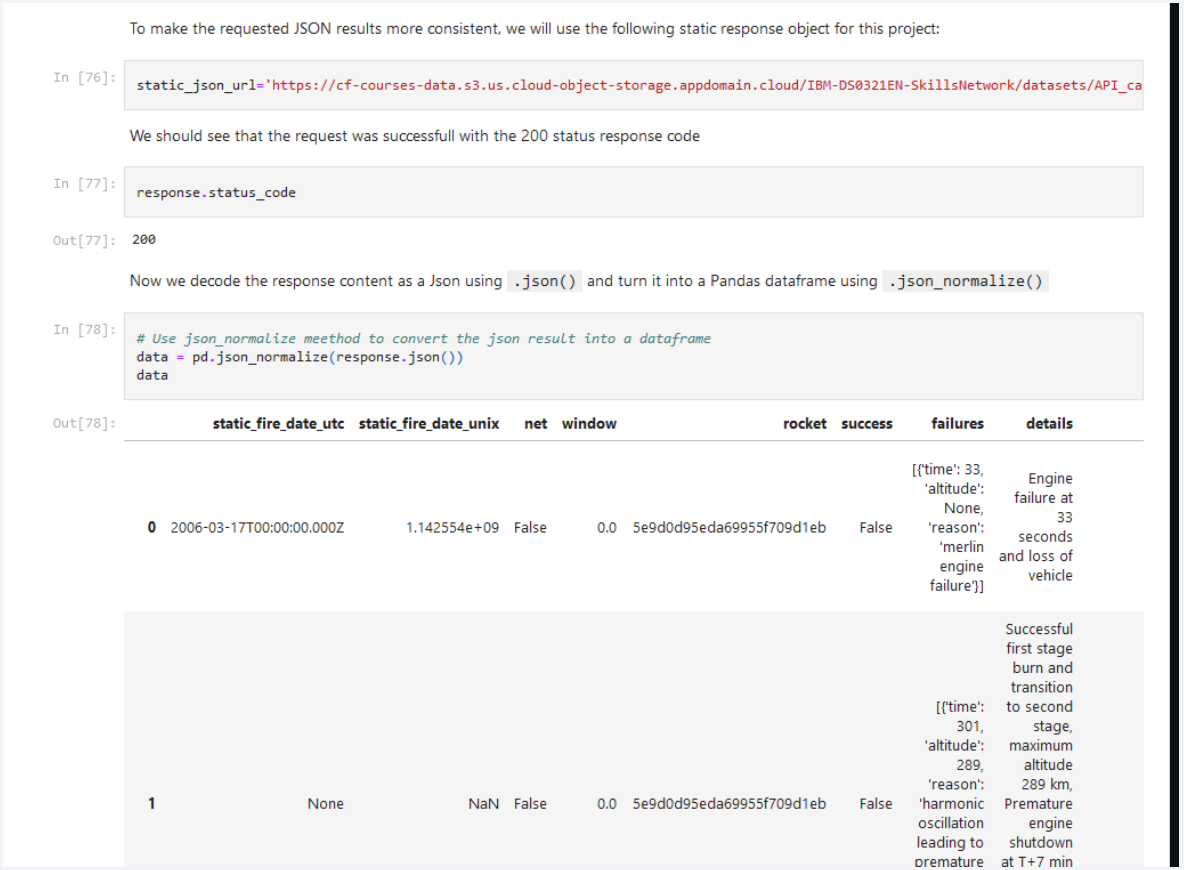
Data Collection

- Here is how we collected the data:
  - First we made requests to the SpaceX API. We used the .json() function to decode it and normalized it before transforming it into a dataframe.
  - We filled missing values with the mean value for that column where necessary.
  - We also used web scraping of the Wikipedia Falcon 9 launch records with BeautifulSoup.
  - This too was parsed into a table and converted into a dataframe for our analysis.

Data Collection – SpaceX API

- We used a GET request to the API to collect the data. We then did some basic formatting and data wrangling. We weren't able to get all the data we needed from this though.

- See this github link for the full lab.



To make the requested JSON results more consistent, we will use the following static response object for this project:

In [76]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_ca
```

We should see that the request was successfull with the 200 status response code

In [77]:
```
response.status_code
```

Out[77]: 200

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

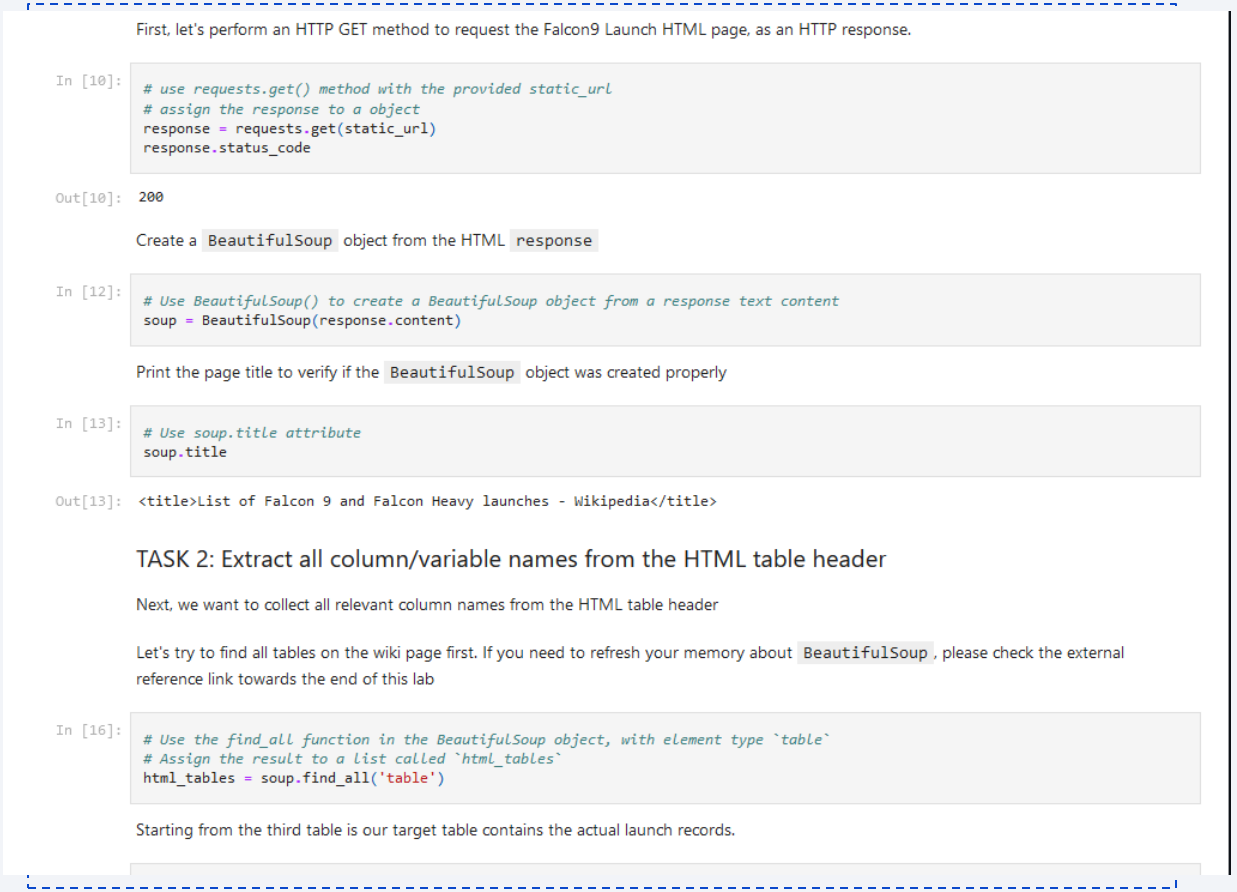
In [78]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
data
```

Out[78]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle
1	None	NaN	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to premature	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min

Data Collection - Scraping

- We used BeautifulSoup to get the records from Wikipedia.
- After extracting the relevant information from the HTML code we converted it into a dataframe.
- Click this github link for the full lab.



First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

In [10]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
response.status_code
```

Out[10]: 200

Create a `BeautifulSoup` object from the HTML `response`

In [12]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content)
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [13]:
```
# Use soup.title attribute
soup.title
```

Out[13]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

In [16]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

Data Wrangling

- We performed exploratory data analysis to determine training labels.
- We then calculated the number of launches at each site and orbit. We also properly labelled the outcomes.
- Click this github link to learn more.

Out[8]:

Orbit	
GTO	27
ISS	21
VLEO	14
PO	9
LEO	7
SSO	5
MEO	3
HEO	1
ES-L1	1
SO	1
GEO	1

Name: count, dtype: int64

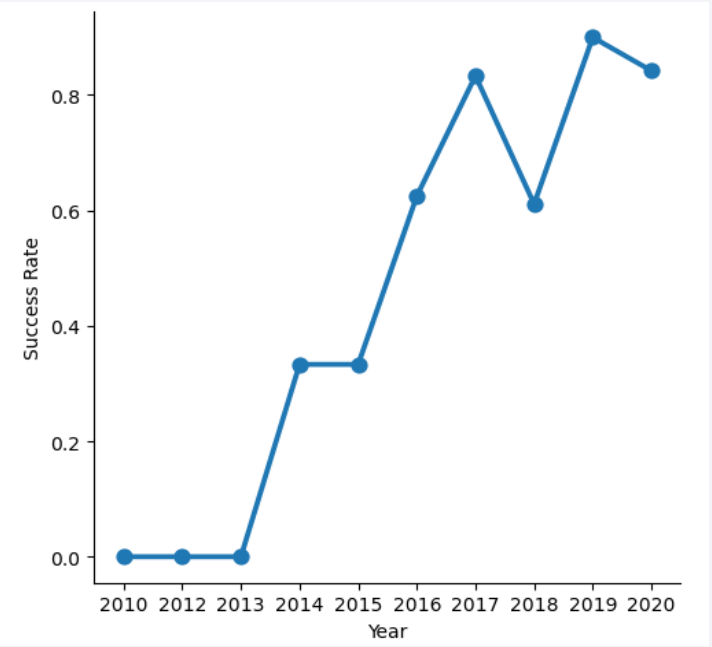
In [18]:

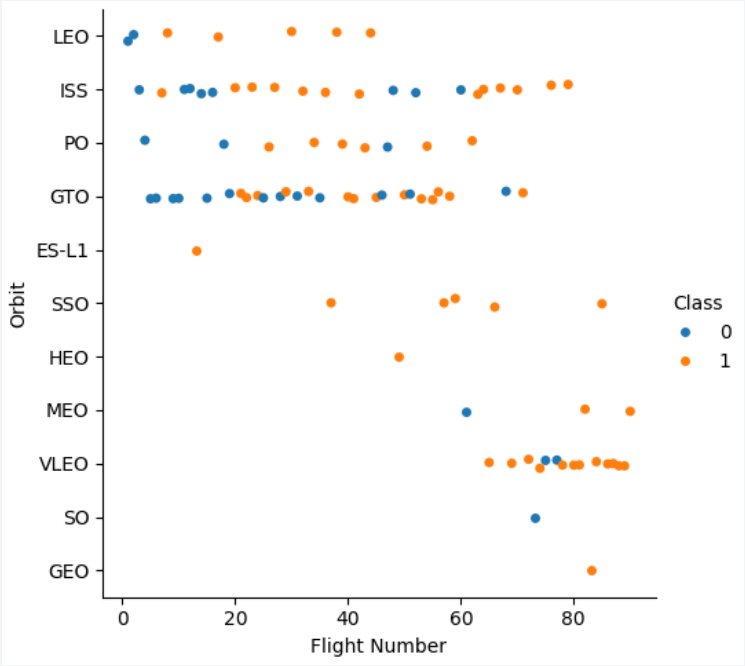
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []

for i in df['Outcome']:
    if i in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)

print(landing_class)
```

```
[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1]
```

EDA with Data Visualization

- We visualized the relationship between many different aspects during the exploratory stage including:
  - Flight # and launch site
  - Success rate and orbit type
  - Payload and launch site
  - Flight # and orbit type
  - Successful launches as the years went by
  - And many more
- Click this github link for more info on the lab





EDA with SQL

- We explored the data looking for insights using SQL. We wrote queries to gather information such as:
  - Names of unique launch sites
  - Payload mass carries by NASA boosters
  - The avg. Payload mass carried by the F9
  - Total successful missions/failures
- Once again, click this github link for the full lab.

Build an Interactive Map with Folium

- We marked the launch sites and added map objects like markers, circles, and spirals to mark the successes and failures of each site.
- The launch outcomes were assigned either a 1 for success or a 0 for failure which was used to calculate the above.
- Using these clusters we are able to show which site is the most successful.
- All of these links are available on the project page of my github but allow me to tediously link every project through these slides. Click here.

Build a Dashboard with Plotly Dash

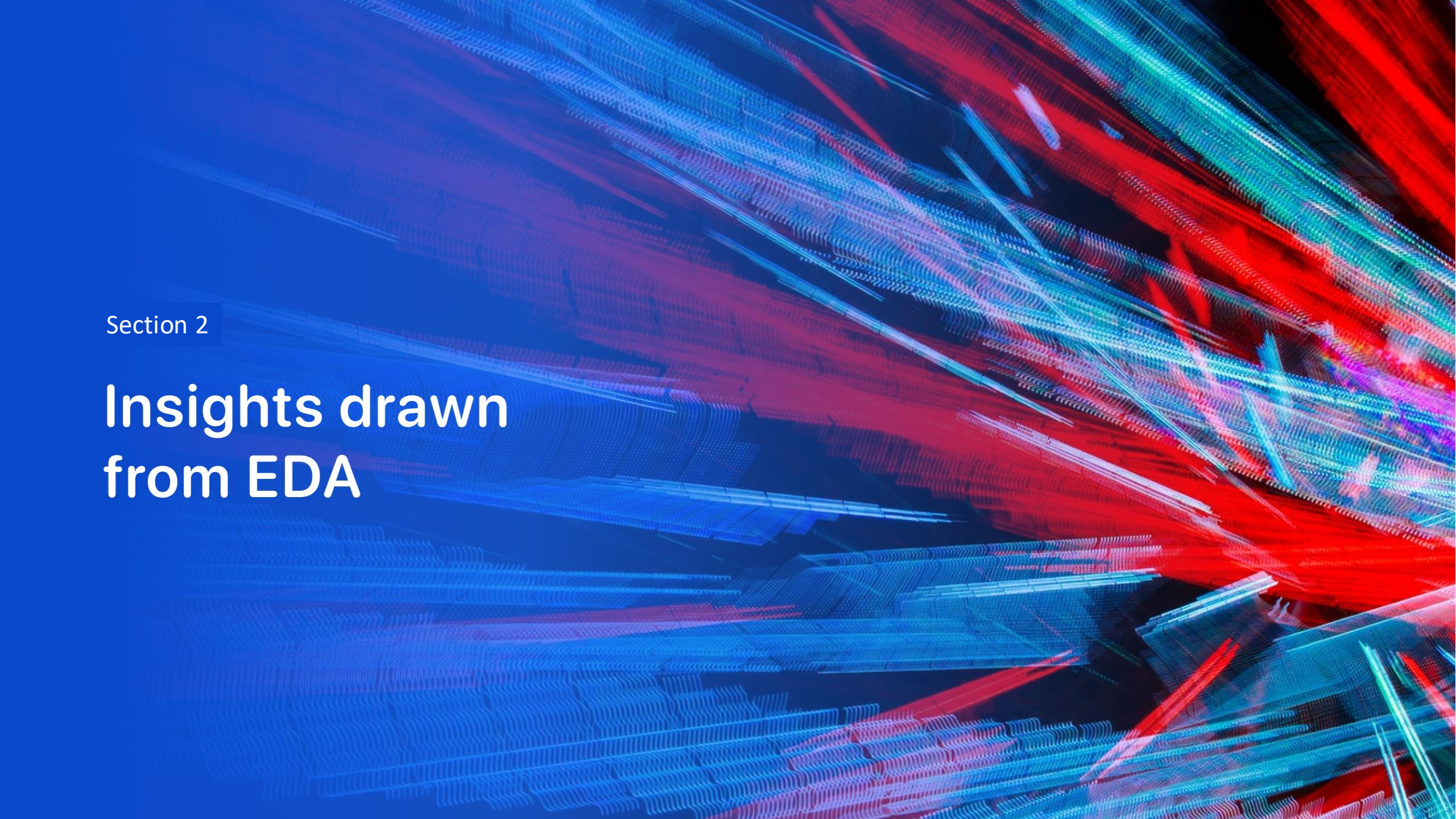
- Using Dash we created interactive charts that show the success rate vs. Payload mass for all sites or individual ones. We made a similar pie chart that shows total successes for each site and can also show successes/failures for specific sites.
- This is important data to consider for our final analysis and model building, and having it in an interactive form makes it easier to decipher.
- See this lil link here for the code that makes this possible.

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas before splitting it into a training/testing set using an 80:20 split.
- We then built different machine learning pipelines with GridSearchCV. They used:
  - Logistic regression
  - Support vector machine
  - And a decision tree classifier
- We improved models using feature engineering and went for the ones with the most accuracy.
- As usual, github link here.

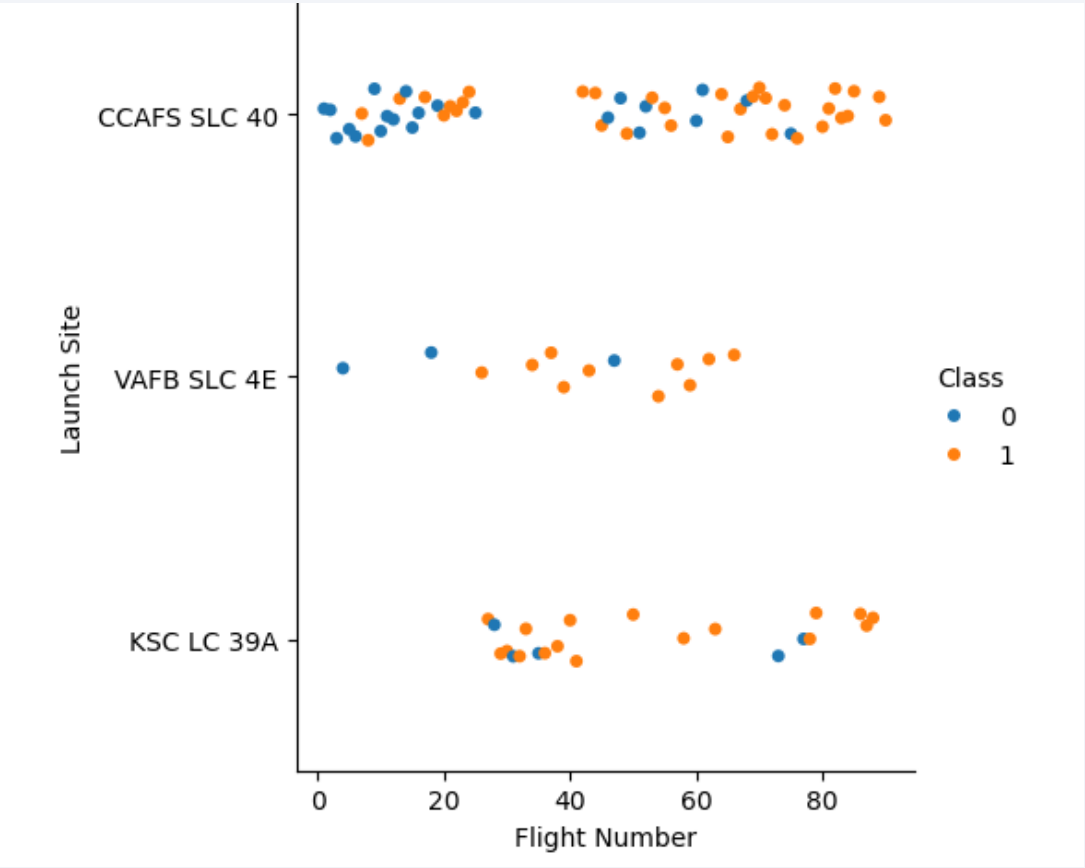
Results

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


Section 2
Insights drawn from EDA

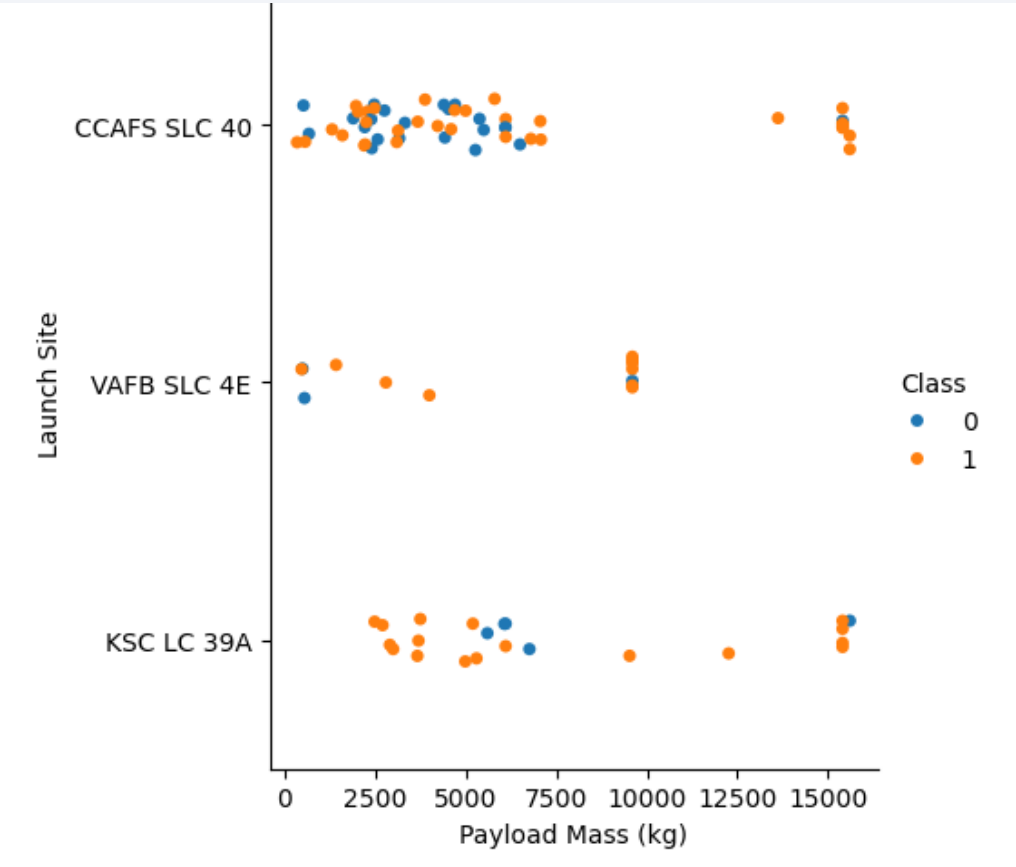
Flight Number vs. Launch Site

From this we were able to gather that as the number of flights increased so did the success rate for each site.



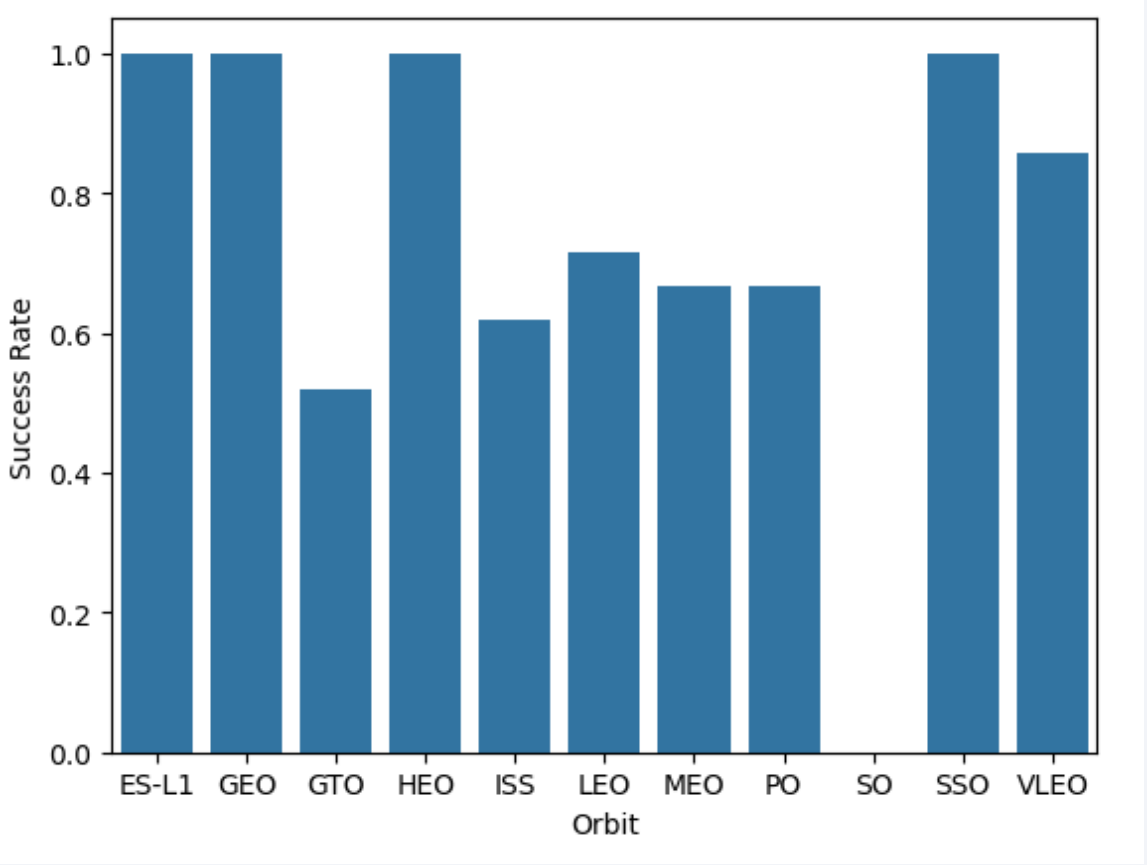
Payload vs. Launch Site

If you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launch site there are no rockets launched for heavy payload mass (greater than 10000).



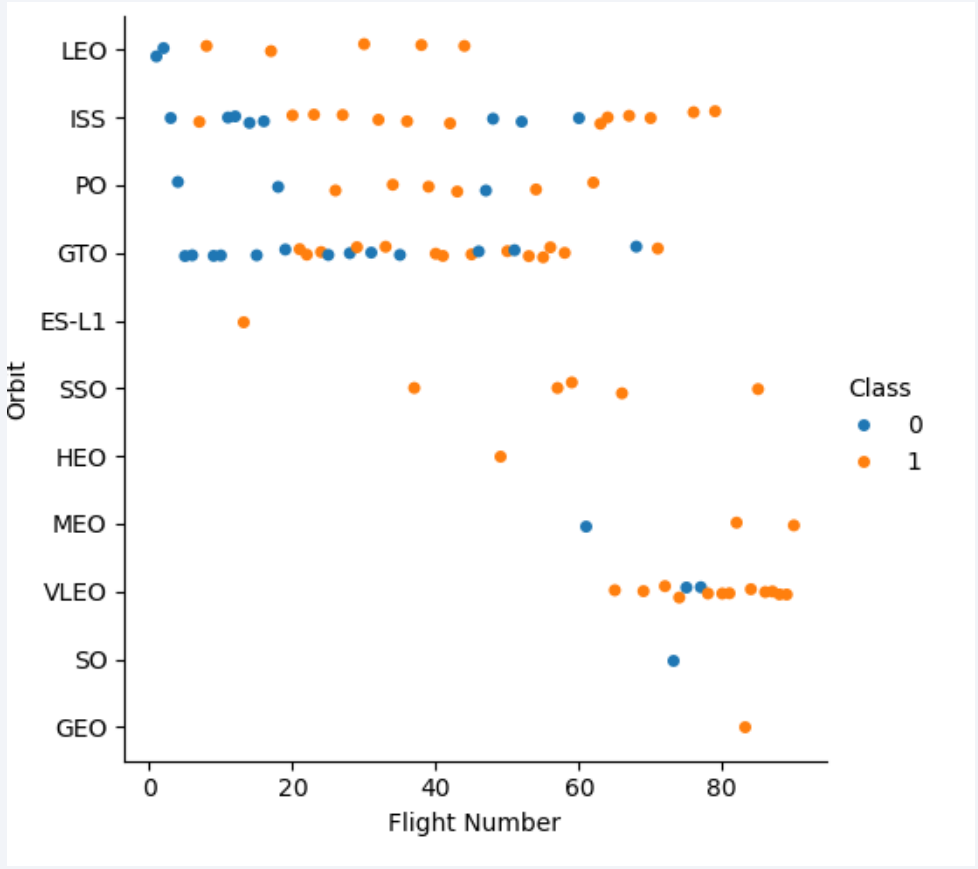
Success Rate vs. Orbit Type

This chart shows us that ES-L1, GEO, HEO and SSO have the highest rates of success.



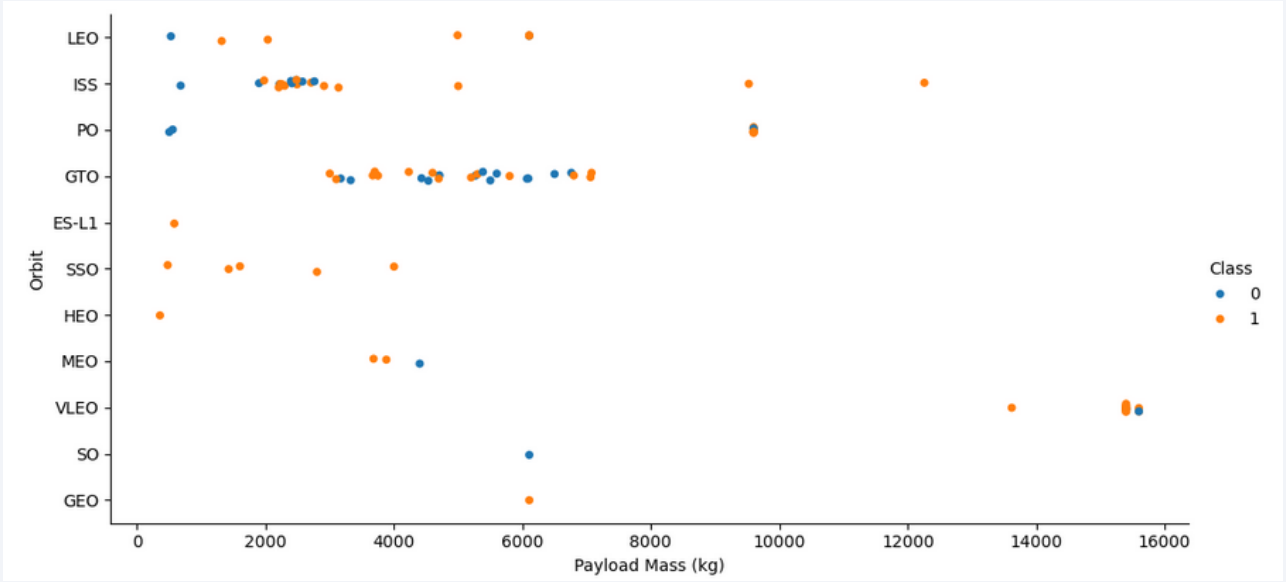
Flight Number vs. Orbit Type

This chart shows us that during the first two dozen flights there were rarely any flights above ES-L1. As the number of flights increases the focus also appears to shift to higher orbits.



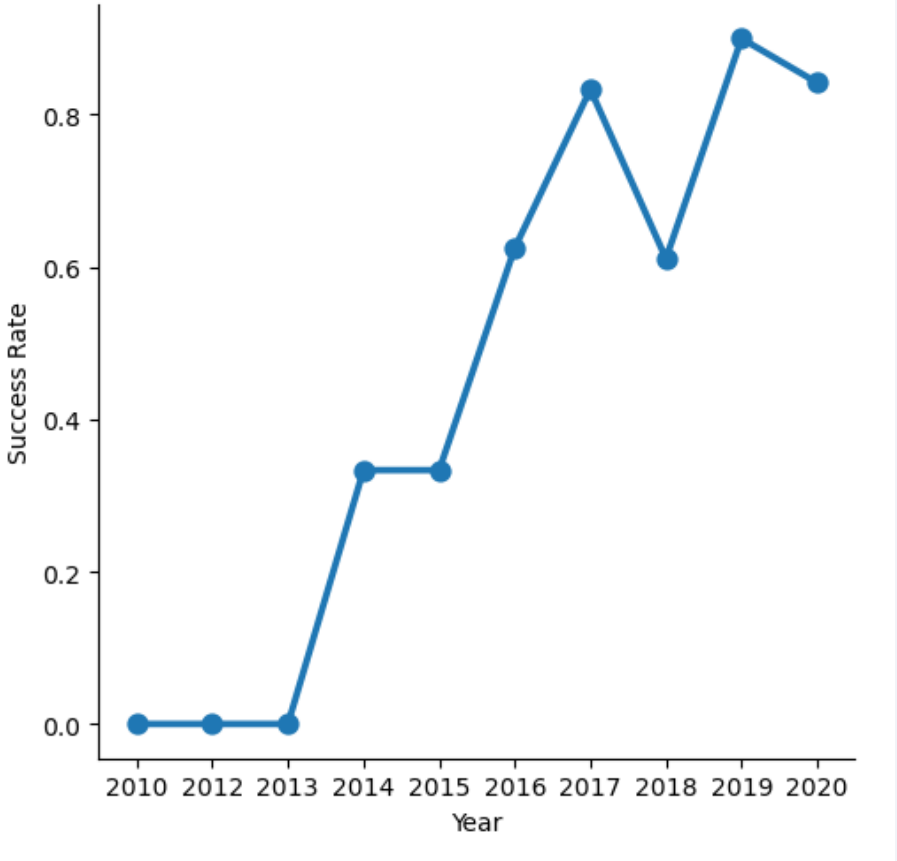
Payload vs. Orbit Type

This chart shows us that the highest payload masses are usually present in MEO and above.



Launch Success Yearly Trend

We already showed this on a previous slide, but clearly the success rate has gone up in a way that strongly correlates with the number of years the company operated.



All Launch Site Names

Here you will find the basic SQL query used to find distinct launch sites. It uses the DISTINCT keyword. I feel like this and most of the slides after this go against the advice of not overwhelming the viewer with useless data but here you go. This is how I did it. Such a magical feat I know.

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

```
%sql select distinct "Launch_Site" from SPACEXTABLE
```

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

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

Launch Site Names Begin with 'CCA'

This is growing more tedious by the second. This picture shows the query I used to get sites with 'CCA' in their names. LIKE is a magical keyword.

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

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

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parac
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 (parac
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

Lord have mercy on my soul. We can see here a sum is easy to calculate. It's 45,596 kg.

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

```
%sql select sum("payload_mass__kg_") from SPACEXTABLE where "Customer" = "NASA (CRS)"
```

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

sum("payload_mass__kg_")
45596

Average Payload Mass by F9 v1.1

There has to be a better way for y'all to grade this. This is torture. Here's my query for finding the avg payload mass. It uses AVG().

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

```
%sql select avg("payload_mass__kg_") from SPACEXTABLE where "Booster_Version" like "F9 v1.1%"
```

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

avg("payload_mass__kg_")
2534.6666666666665

First Successful Ground Landing Date

I just did what the hint said.

Why did I even link this lab in a previous slide when I was gonna paste screenshots of every single answer here?

First success was December 22nd 2015. Dang.

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

Hint:Use min function

```
%sql select min("date") from SPACEXTABLE 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

Kill me. What do you mean present the results with a short explanation? Read the screenshot it's all there.

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

```
%sql select "booster_version" from SPACEXTABLE where "landing_outcome" like "%drone ship%" and "payload_mass__kg_" between 400
```

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

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

Total Number of Successful and Failure Mission Outcomes

Are any of the stakeholders even awake anymore? Such boring data. Many successes, hard query to make. Wildcards were used.

List the total number of successful and failure mission outcomes

```
%sql SELECT count(*), COUNT(case when "mission_outcome" like "S%" then 1 else null end) as success, COUNT(case when "mission_o
```

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

count(*)	success	failure
101	100	1

Boosters Carried Maximum Payload

This is the Coursera equivalent of uploading your resume and having to type it out again on the next page of the application. This one was a lil messy.

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

```
%sql select "payload_mass__kg_", "booster_version" from SPACEXTABLE where "payload_mass__kg_" = (select max("payload_mass__kg_
```

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

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

2015 Launch Records

At last, my suffering nears its end. Used WHERE, LIKE and the fabled BETWEEN keywords.

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

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

```
%sql select substr(date, 6,2) as "month (num)", landing_outcome, booster_version, launch_site from SPACEXTABLE where landing_o
```

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

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

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

Sweet, sweet rapture. The highest outcome is no attempt.

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
%sql select landing_outcome, count(*) as tally from SPACEXTABLE where date between "2010-06-04" and "2017-03-20" group by land
```

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

Landing_Outcome	tally
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

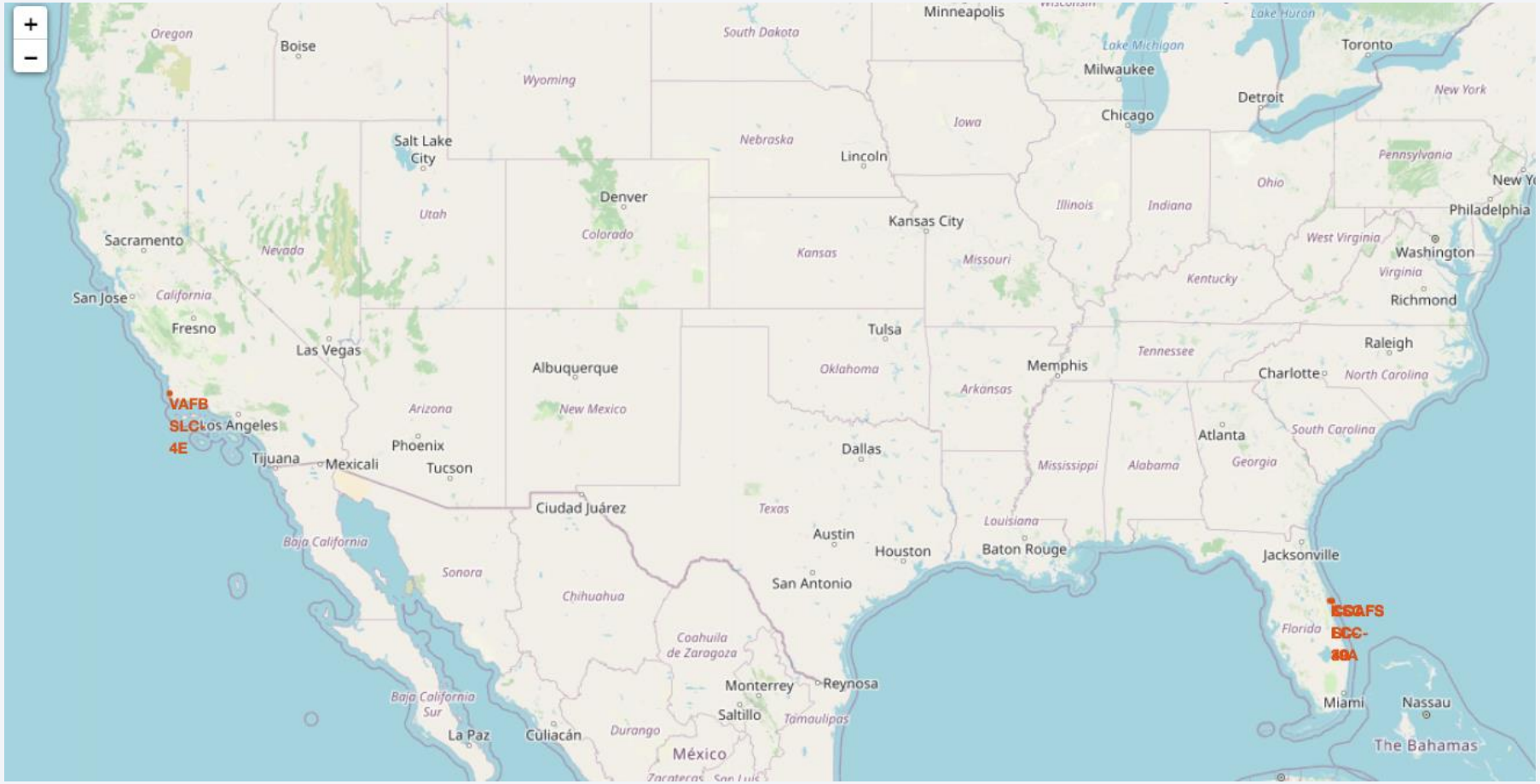
Y'all wanna watch Megalopolis later?

Section 3

Launch Sites Proximities Analysis

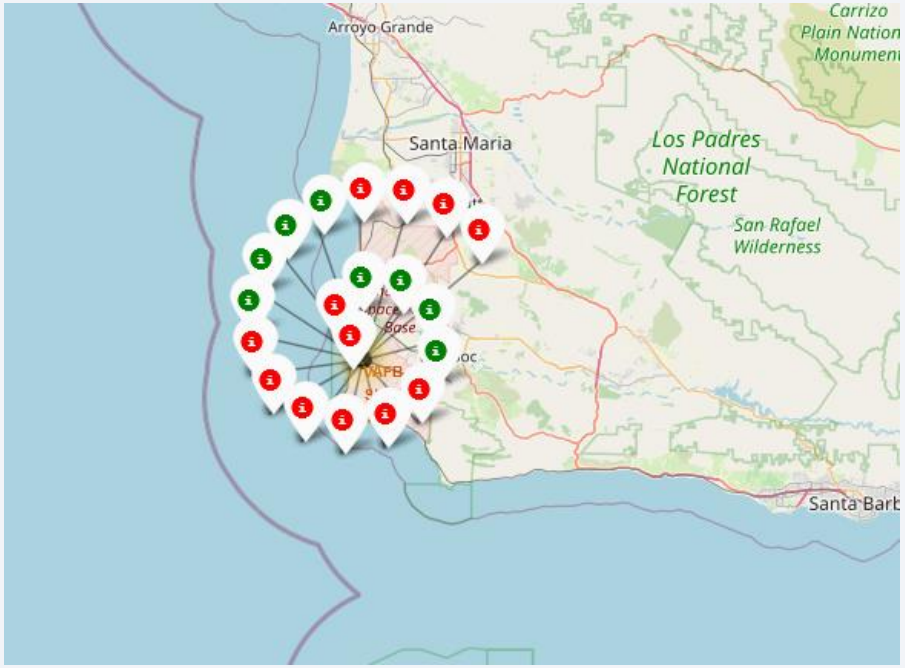
All SpaceX launch sites

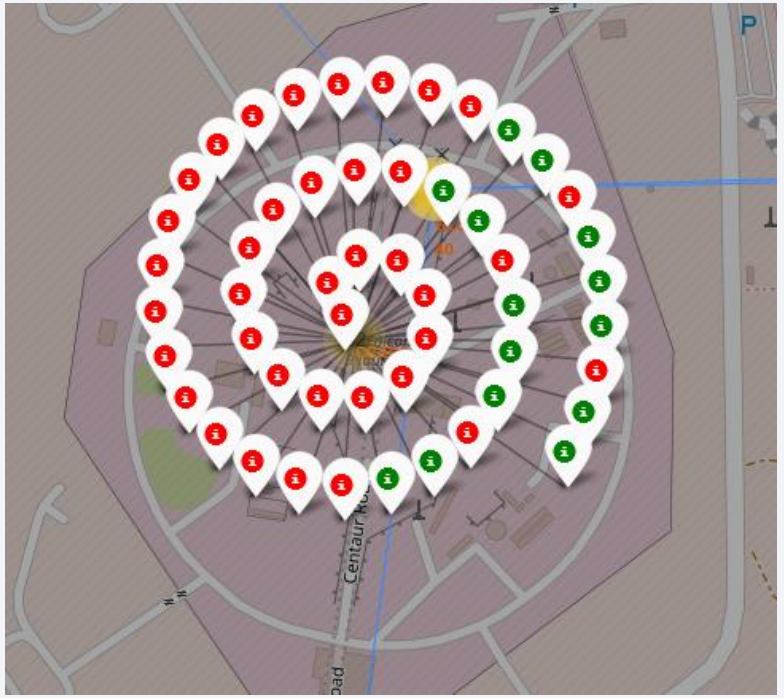
All launch sites are on the coast of the USA located in Florida and California away from any major cities.



Launch Site Markers with Colored Labels

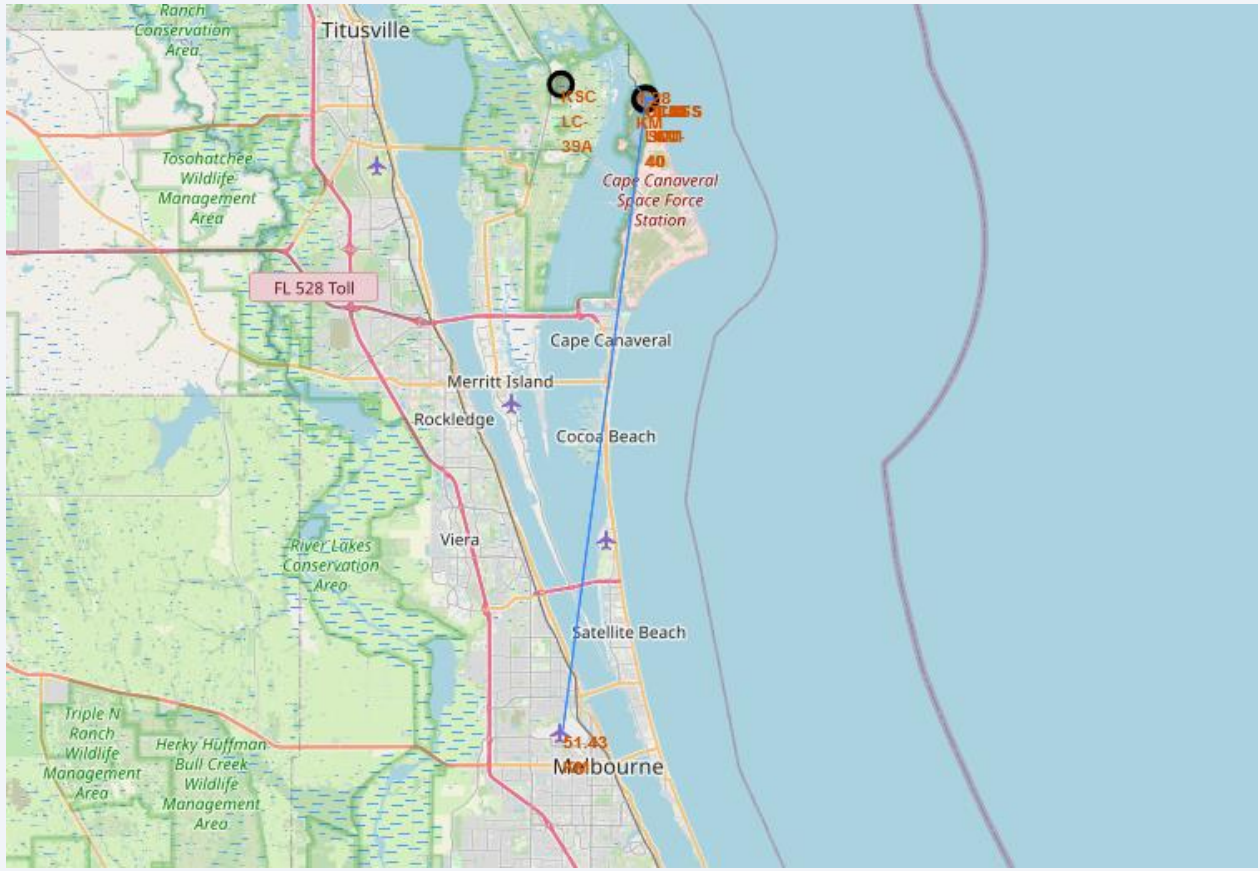
This interactive visualization allows us to see at a glance the number of successes (green) and failures (red). At each launch site.

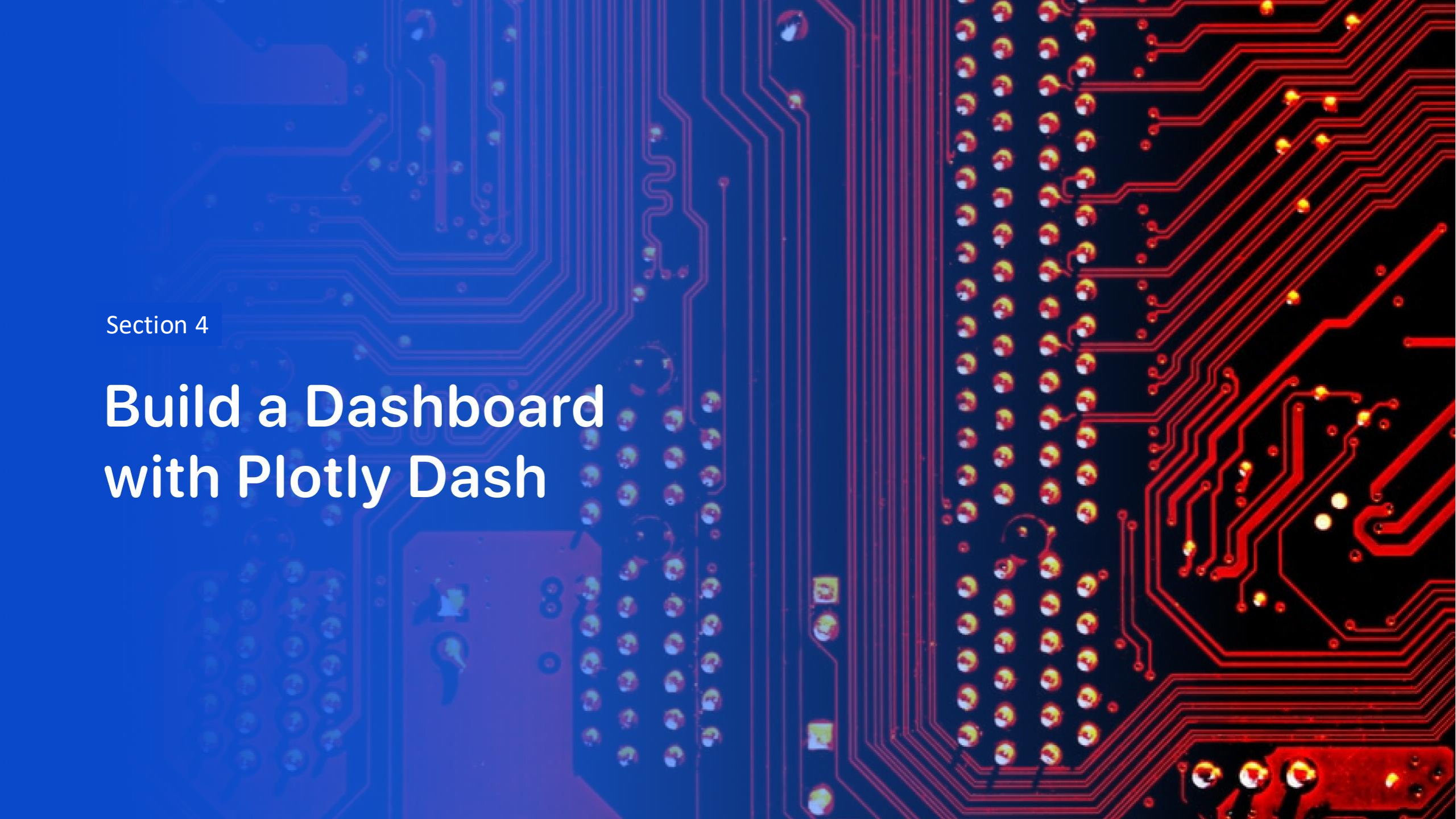




Landing Site Proximity to Infrastructure and Cities

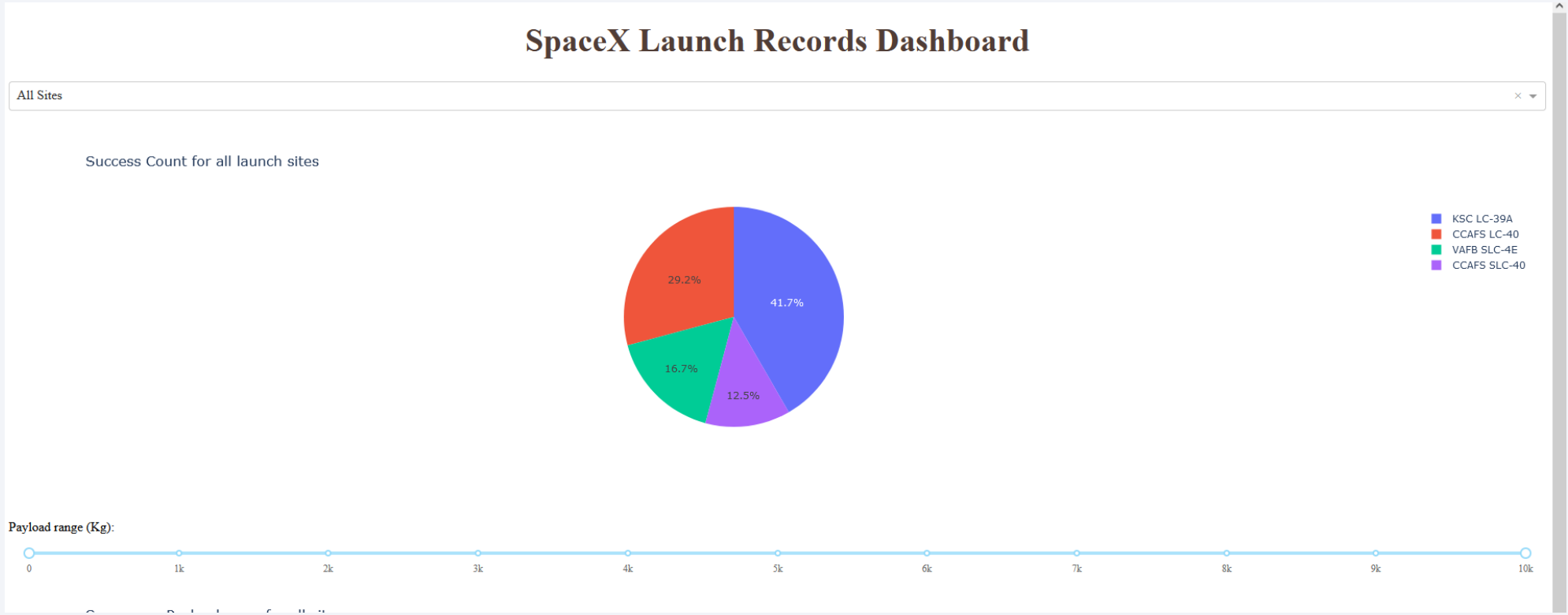
Here we are able to show, visually, how launch sites are close to highways and railways but far from major cities, likely for logistics re. the former and safety the later.




Section 4
Build a Dashboard with Plotly Dash

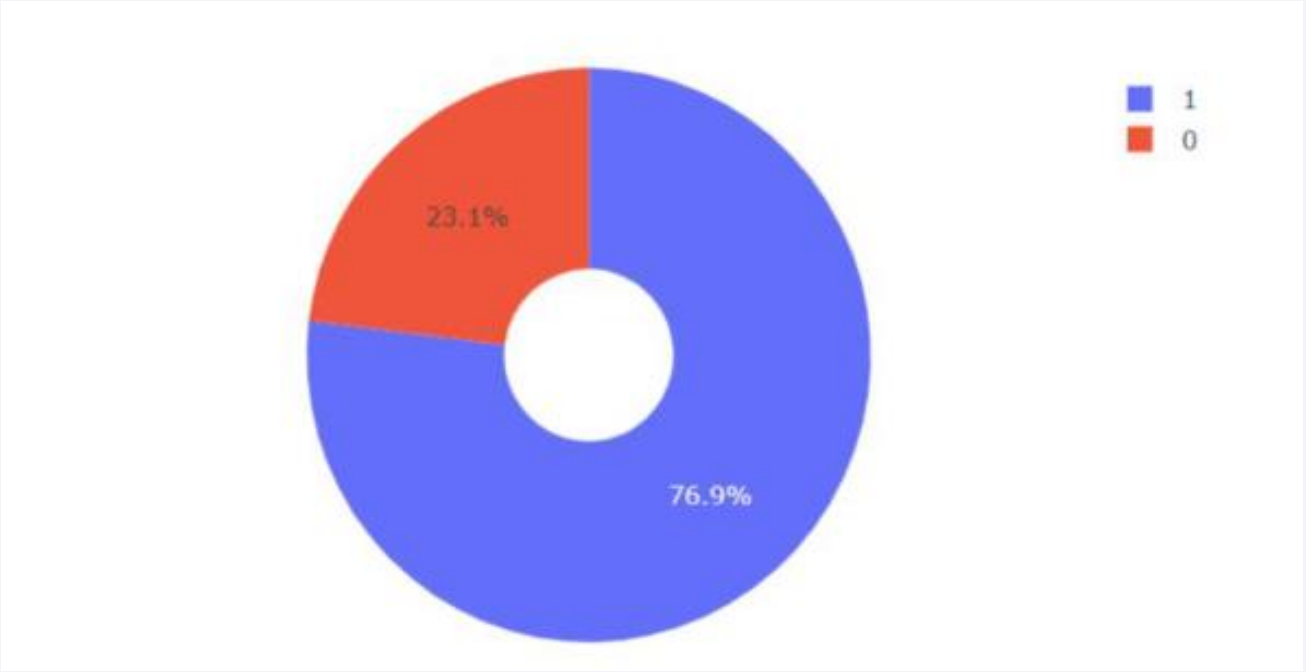
A Pie Chart Demonstrating Successes/Launch Site

KSC LC-39A is the leader with 41.7% of successes reported



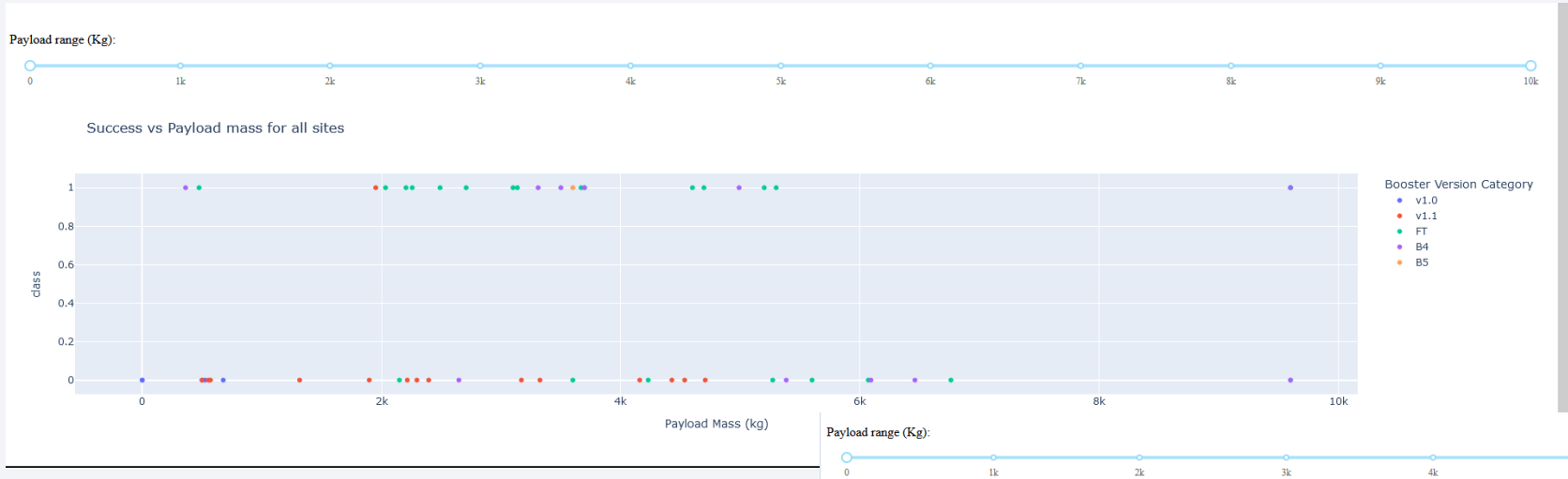
Launch Site with the Highest Success Rate

KSC LC-39A was the best performing landing site.



Scatter Plots for Payload vs Launch Outcome for All Sites + Just KSC LC-39A

Generally the success rate is higher for lower payloads.



Section 5

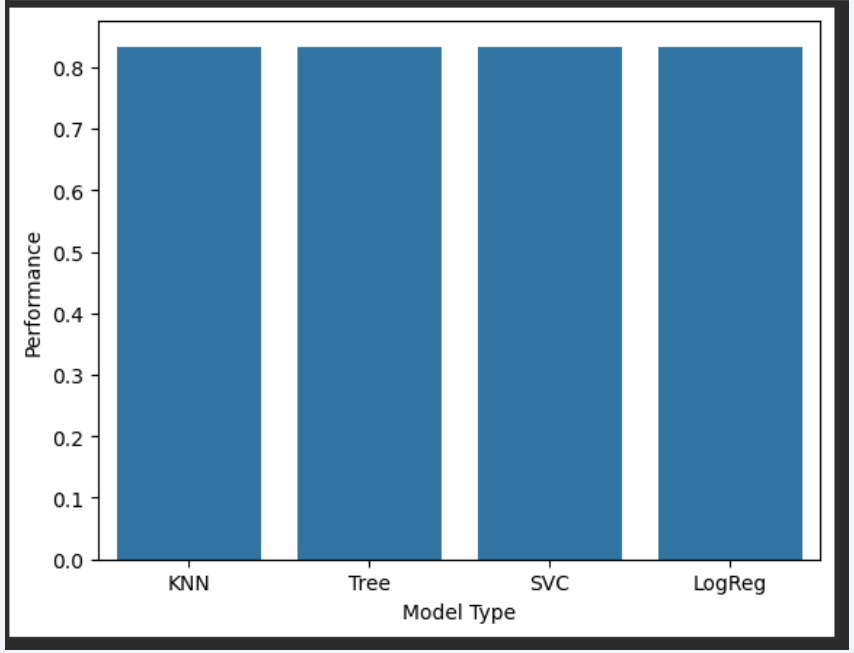
Predictive Analysis (Classification)

Classification Accuracy

We visualized the accuracy score of each different model and found...they're all the same. My theory is that this has to do with the low number of test samples.

When running the entire data set through each model we found the tree to be the most accurate.

```
methods = ['KNN', 'Tree', 'SVC', 'LogReg']
values = [knn_cv.score(X_test, Y_test),
          tree_cv.score(X_test, Y_test),
          svm_cv.score(X_test, Y_test),
          logreg_cv.score(X_test, Y_test)]

perfDF = pd.DataFrame({
    'Model': methods,
    'Performance': values
})

#perfDF.head()

sns.barplot(x='Model', y="Performance", data=perfDF)
plt.xlabel("Model Type")
plt.ylabel("Performance")
plt.show()
```

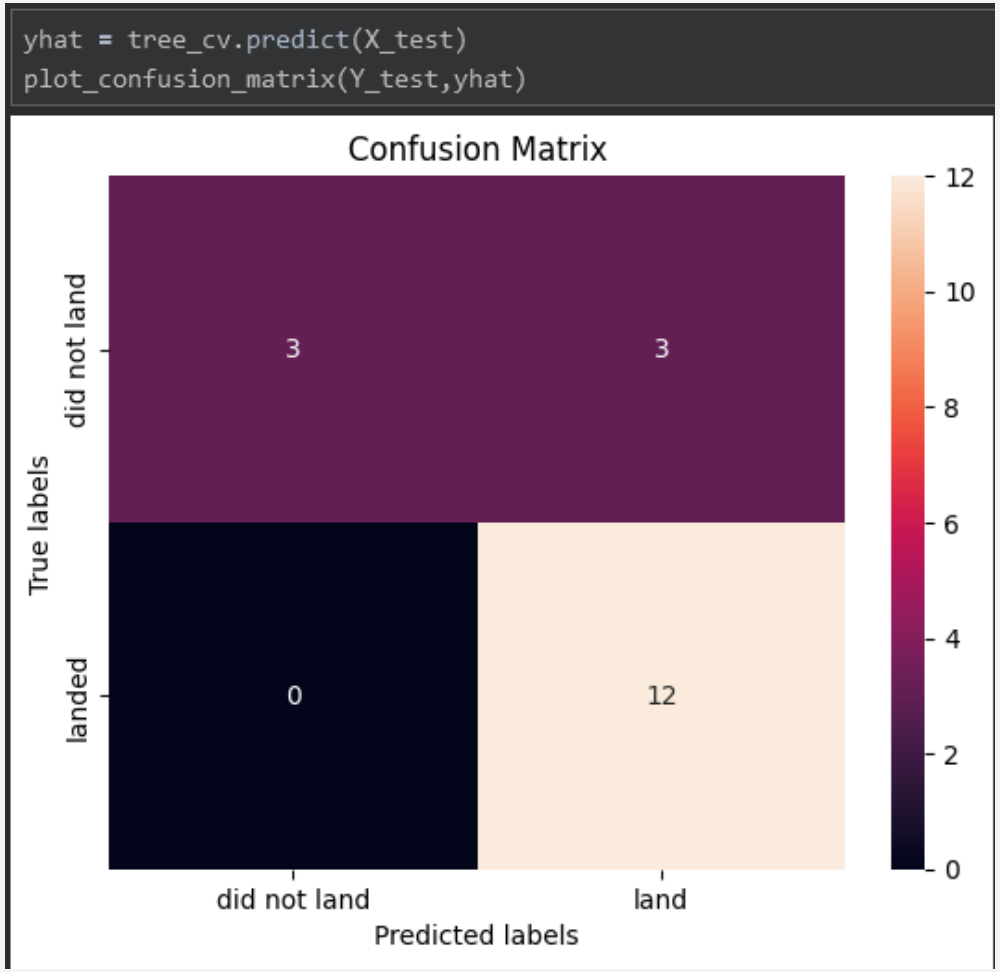


Confusion Matrix

The tree was the best performing model. This confusion matrix shows that its accuracy for the failed landings is perfect, with no false negatives.

However, the accuracy for landings isn't as good, with 3 false positives accounting for a 20% error rate.

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



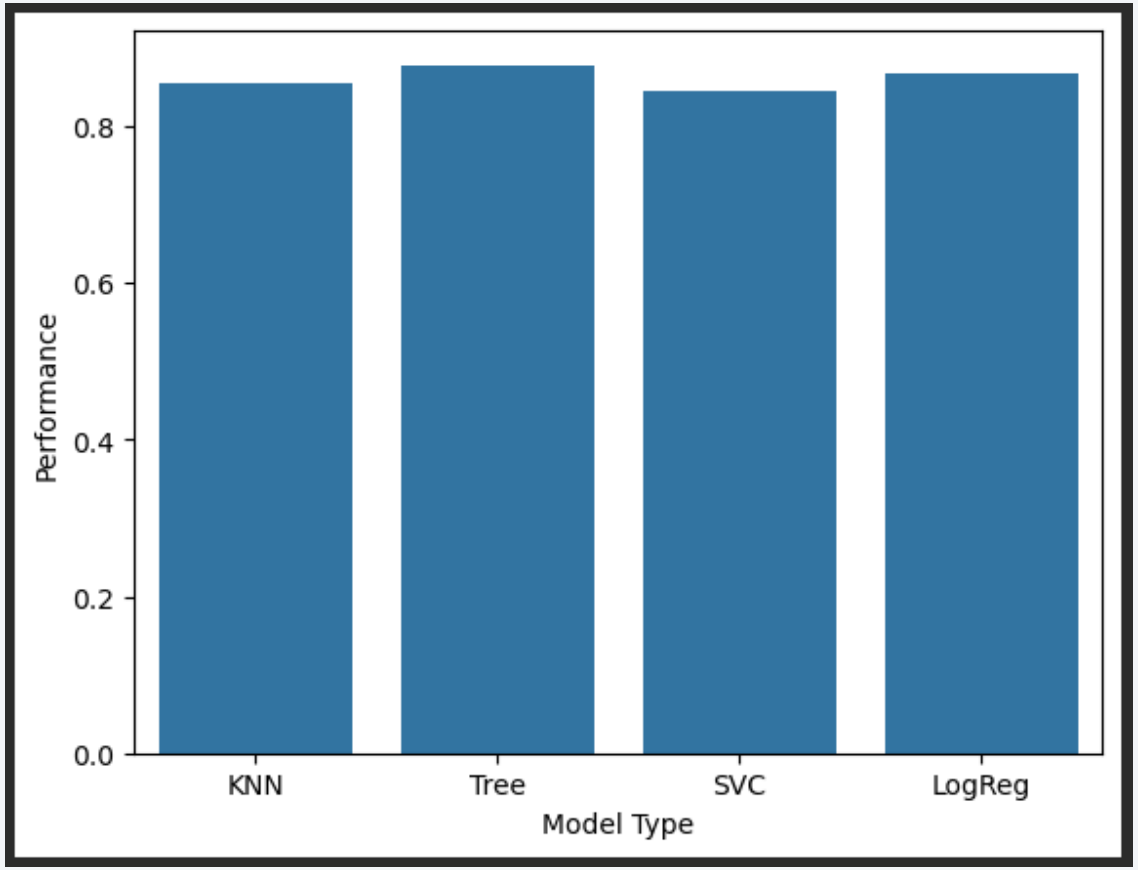
Conclusions

From our findings we can conclude:

- As SpaceX launches more rockets the overall success rate will increase for any launch site.
- Similarly, from the first launch in 2013 to 2020 the success rate increased.
- KSC LC-39A has had the highest number of successful launches.
- The most successful launch orbits are ES-L1, GEO, SSO, HEO, and VLEO.
- Overall the decision tree model performed the best and should be used to predict the success of launches.

Appendix

Here is the other bar chart showing accuracy for the original dataset.



Thank you!