

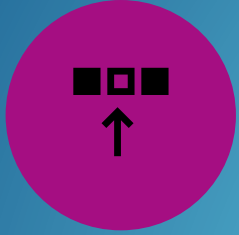


IBM Developer  
SKILLS NETWORK

# Winning Space Race with Data Science

Adam Thornill  
01/01/2026





EXECUTIVE  
SUMMARY



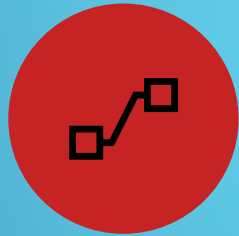
INTRODUCTION



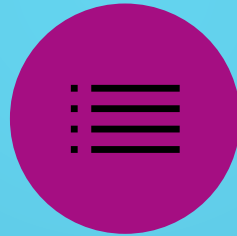
METHODOLOGY



RESULTS



CONCLUSION



APPENDIX

# OUTLINE

## Summary of methodologies

- Data Collection through API
- Data Collection with Web Scraping
- Data Wrangling
- Exploratory Data Analysis with SQL
- Exploratory Data Analysis with Data Visualization
- Interactive Visual Analytics with Folium
- Machine Learning Prediction

## Summary of all results

- Exploratory Data Analysis Result
- Interactive Analytics in screenshots
- Predictive Analytics Result

# EXECUTIVE SUMMARY

- ▶ Project Background and Context

- ▶ This project aims to predict the likelihood of a successful landing of the Falcon 9 first stage. SpaceX advertises Falcon 9 launches at a cost of \$62 million, significantly lower than other providers, which charge upwards of \$165 million per launch. Much of this cost advantage stems from SpaceX's ability to reuse the first stage. Therefore, accurately predicting the first stage's landing success can help estimate the overall launch cost. This insight is valuable for competitors seeking to bid against SpaceX for rocket launch contracts.

- ▶ Key Questions to Address

- ▶ Which factors influence the successful landing of the rocket?
  - ▶ How do various features interact to affect the probability of a successful landing?
  - ▶ What operational conditions are necessary to ensure a reliable and successful landing programme?

# INTRODUCTION

Section 1

# Methodology



## Executive Summary



## Data collection methodology:

Data for this project was collected using SpaceX API and web scraping from Wikipedia



## Perform data wrangling

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

How to build, tune, evaluate classification models

# METHODOLOGY

# DATA COLLECTION

- ▶ The data for the Project was collected with various Methods:
  - ▶ Data collection was done using get request to the SpaceX API.
  - ▶ Next, we decoded the response content as a Json using `.json()` function call and turn it into a pandas dataframe using `.json_normalize()`.
  - ▶ We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - ▶ In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - ▶ The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

# Data Collection – SpaceX API



We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.



GitHub URL is attached here:



<https://github.com/The-Acehole/Data-Science-Capstone-Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb>

## 1 – Get request for Rocket Launch

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)
```

## 2 – Use Json normalize to convert Json to DataFrame

```
In [10]: response=requests.get(static_json_url)

In [11]: response.status_code

Out[11]: 200

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

In [12]: # Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

## 3 – Performed Data Cleaning and filled Missing Data

```
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(data_falcon9['PayloadMass'].mean())
data_falcon9.isnull().sum()
```

calls

- ▶ To collect the Data, we applied WebScraping to the Falcon 9 launch records, using BeautifulSoup
- ▶ We parsed the Table and converted it into a Pandas DataFrame
- ▶ The link to the Notebook is:
  - ▶ [https://github.com/The-Acehole/Data-Science-Capstone-Project/blob/main/jupyter-labs-webscraping%20\(1\).ipynb](https://github.com/The-Acehole/Data-Science-Capstone-Project/blob/main/jupyter-labs-webscraping%20(1).ipynb)

# DATA COLLECTION - SCRAPING

1. Apply HTTP get method top request Falcon 9 rocket launch page:

```
In [4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=104472124"
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
}
```

2. Create a BeautifulSoup object from the HTML response:

```
In [5]: # use requests.get() method with the provided static_url and headers
# assign the response to a object
html_data = requests.get(static_url, headers=headers)
html_data.status_code

Out[5]: 200
```

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text)

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

```
In [7]: # Use soup.title attribute
soup.title

Out[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3. Extract all Column names from the HTML table header:

```
In [12]: column_names = []
# Apply find_all() function with 'th' element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name ('if name is not None and len(name) > 0') into a list called column_names

for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)

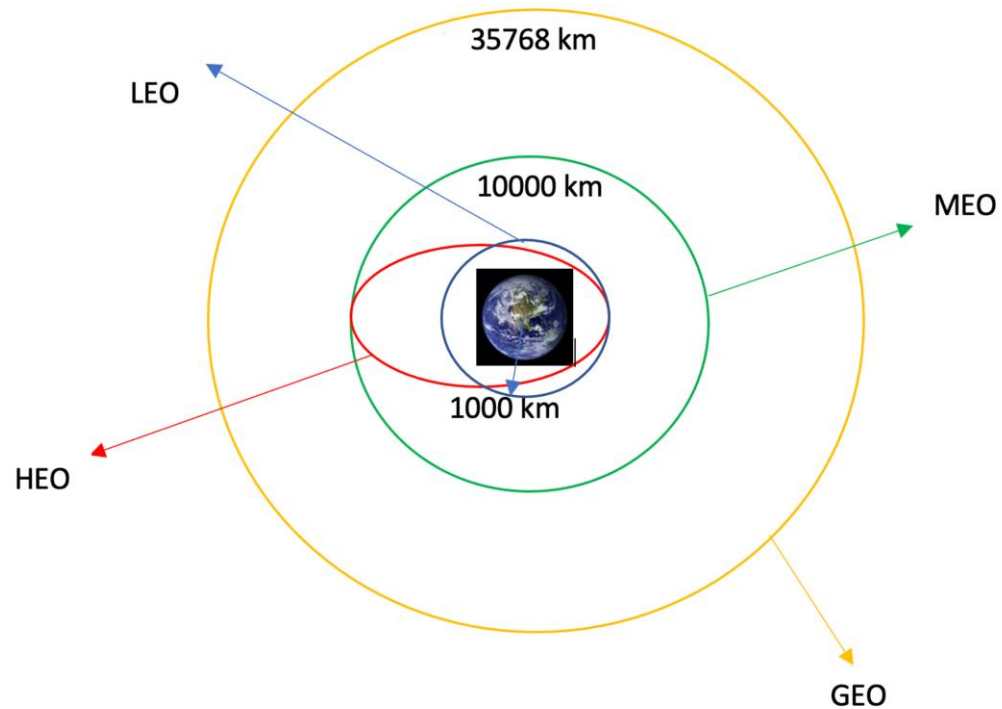
Check the extracted column names
```

```
In [14]: print(column_names)

['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch status']
```

4. Create a DataFrame by parsing the launch HTML tables

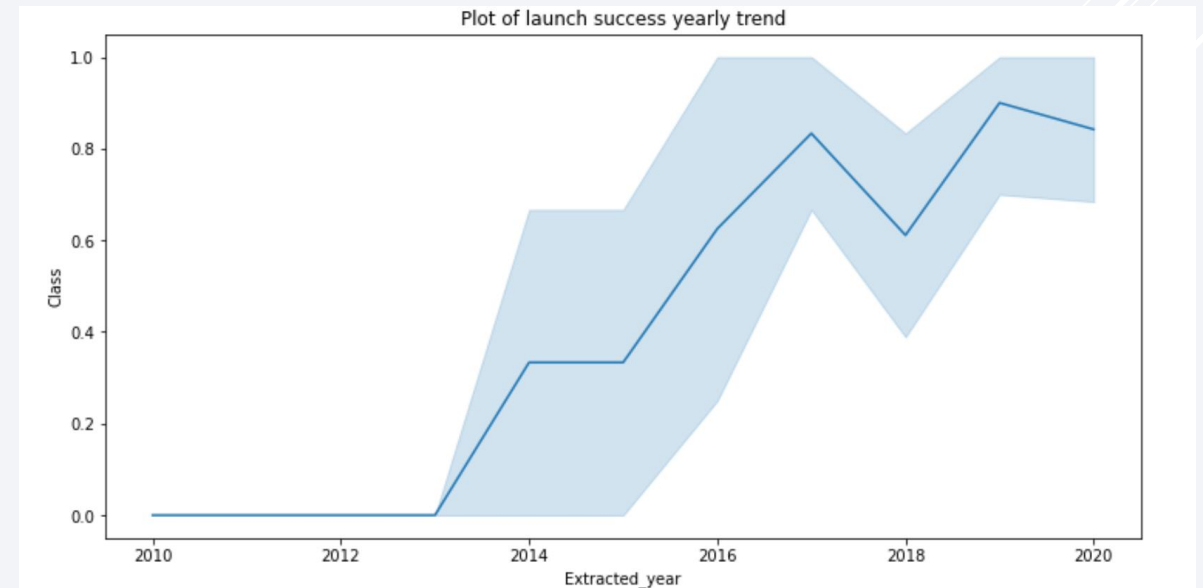
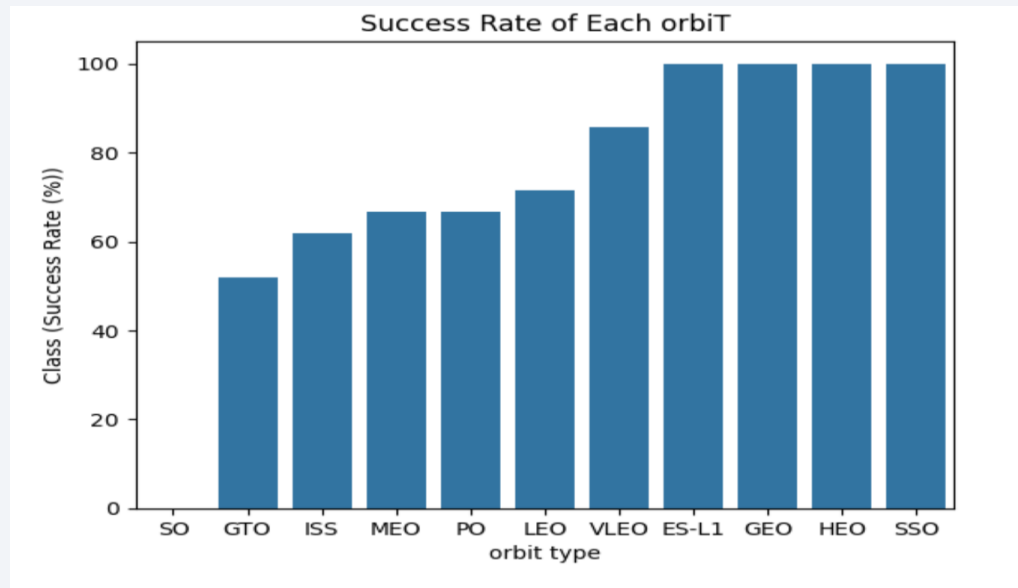
5. Export data to CSV



- ▶ We performed exploratory data analysis and determined the training labels.
- ▶ We calculated the number of launches at each site, and the number and occurrence of each orbits
- ▶ We created landing outcome label from outcome column and exported the results to csv.
- ▶ The link top the GitHub notebooks:  
▶ <https://github.com/The-Acehole/Data-Science-Capstone-Project/blob/main/nbs-jupyter-spacex-Data%20wrangling.ipynb>

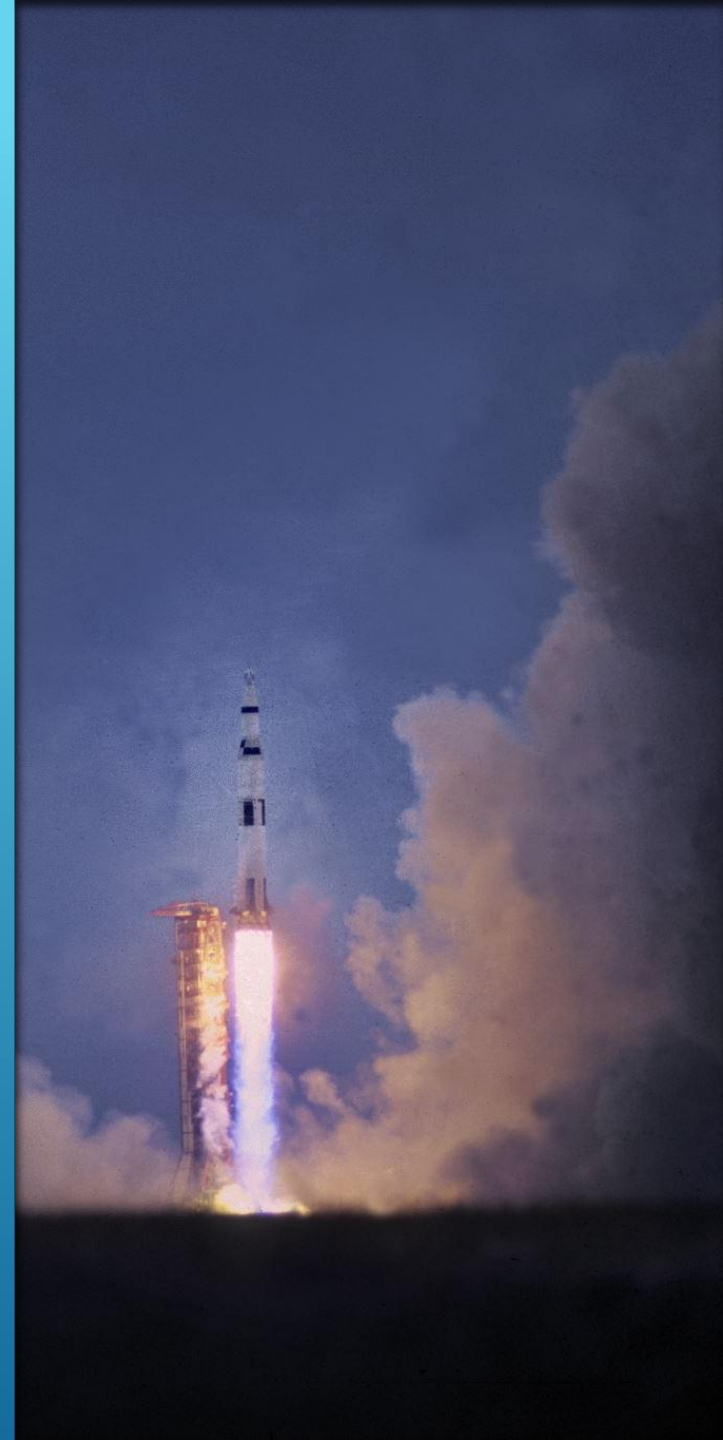
# EDA with Data Visualization

- ▶ We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.
- ▶ The link for the Notebook is: <https://github.com/The-Acehole/Data-Science-Capstone-Project/blob/main/edadataviz.ipynb>



- ▶ We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- ▶ We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - ▶ The names of unique launch sites in the space mission.
  - ▶ The total payload mass carried by boosters launched by NASA (CRS)
  - ▶ The average payload mass carried by booster version F9 v1.1
  - ▶ The total number of successful and failure mission outcomes
  - ▶ The failed landing outcomes in drone ship, their booster version and launch site names.
- ▶ The link to the Notebook is: [https://github.com/The-Acehole/Data-Science-Capstone-Project/blob/main/jupyter-labs-eda-sql-coursera\\_sqlite%20\(6\).ipynb](https://github.com/The-Acehole/Data-Science-Capstone-Project/blob/main/jupyter-labs-eda-sql-coursera_sqlite%20(6).ipynb)


## EDA WITH SQL



# Build an Interactive Map with Folium

---


We marked all of the launch sites, and added Map objects – such as Markers, Circles, Lines to mark the success or failure of launches for each site on the folium map



We assigned the feature launch outcomes (success or failure) to Class 0 and 1 – (i.e. 0 for failure, 1 for success)



Using the colour-labelled marker clusters, we identified which launch sites have relatively high success rate



We calculated the distance between a launch site to it's proximities. We answered some questions for instances

- Are Launch sites near railways, highways, or coastlines
- Do Launch sites keep certain distance away from cities



We built an interactive Dashboard using Plotly Dash



We plotted Pie Charts showing the total launches by certain sites



We plotted scatter graphs showing the relationship with Outcomes and Payload Mass (Kg) for the different booster version



Link to the Notebook: [https://github.com/The-Acehole/Data-Science-Capstone-Project/blob/main/spacex-dash-app%20\(1\).py](https://github.com/The-Acehole/Data-Science-Capstone-Project/blob/main/spacex-dash-app%20(1).py)

# BUILD A DASHBOARD WITH PLOTLY DASH

# Predictive Analysis (Classification)

---

We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.



We built different machine learning models and tune different hyperparameters using GridSearchCV.



We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.



We found the best performing classification model.



The link to the Notebook is: [https://github.com/The-Acehole/Data-Science-Capstone-Project/blob/main/SpaceX\\_Machine%20Learning%20Prediction\\_Part\\_5.ipynb](https://github.com/The-Acehole/Data-Science-Capstone-Project/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb)



EXPLORATORY  
DATA ANALYSIS  
RESULTS



INTERACTIVE  
ANALYTICS DEMO  
IN SCREENSHOTS



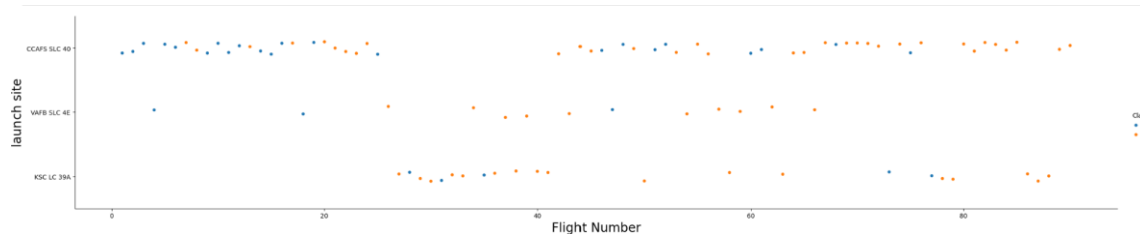
PREDICTIVE  
ANALYSIS RESULTS

# RESULTS

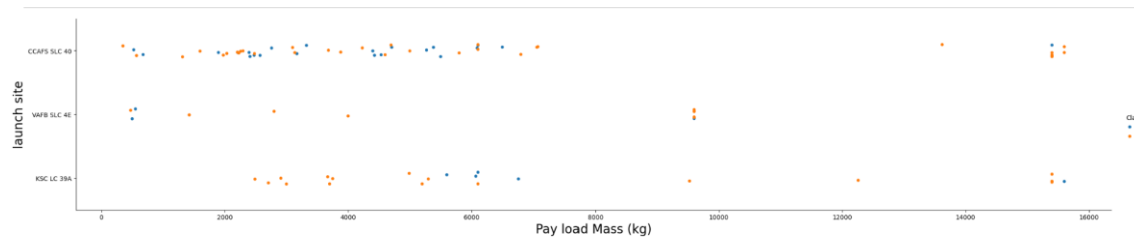
The background of the slide is an abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks in shades of blue and red, creating a sense of motion and depth. A faint, white grid pattern is visible across the entire image, adding a technical or digital feel. The text is positioned on the left side, where the blue background is more prominent.

Section 2

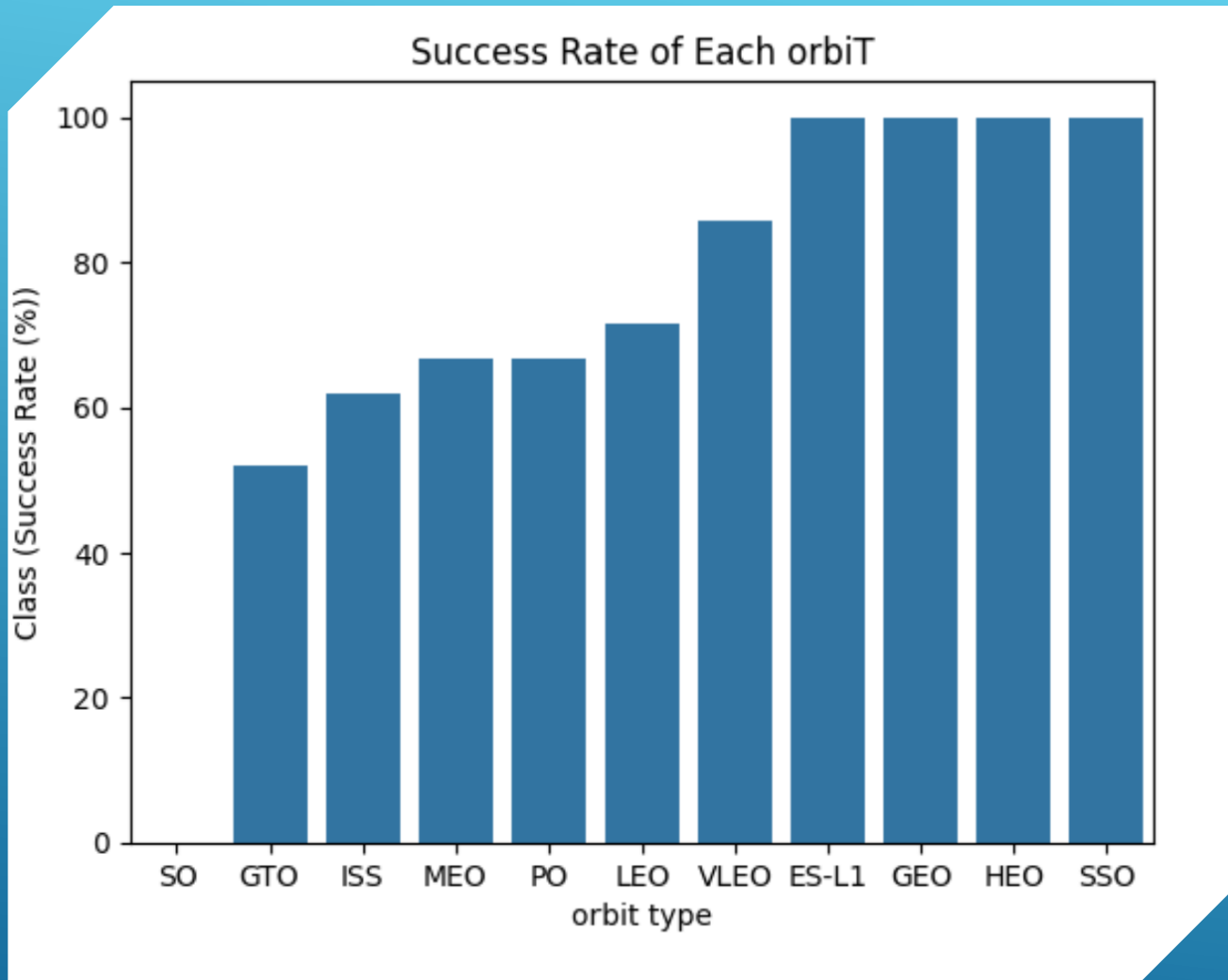
# Insights drawn from EDA



- ▶ From the scatter plot above, we see that CCAF5 SLC40 is the best launch site
- ▶ Second place was VAFB SLC 4E
- ▶ We see that over time, success rate generally improved



- ▶ This scatter plot shows that Payloads below 9,000Kg have excellent success rate
- ▶ Payloads over 12,000Kg seem to only happen at CCAFS SLC 40 and KSC LC 39A
- ▶ A too heavy payload can make a landing fail based off what the chart shows

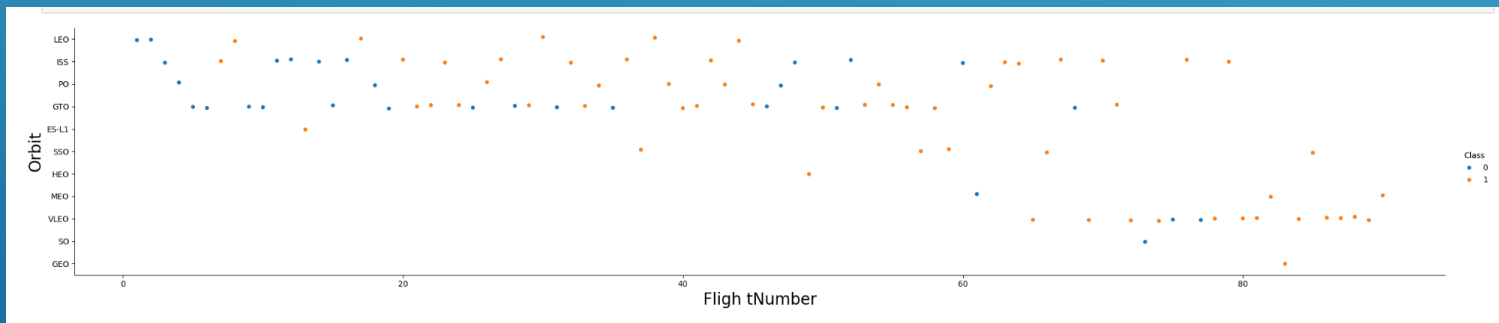


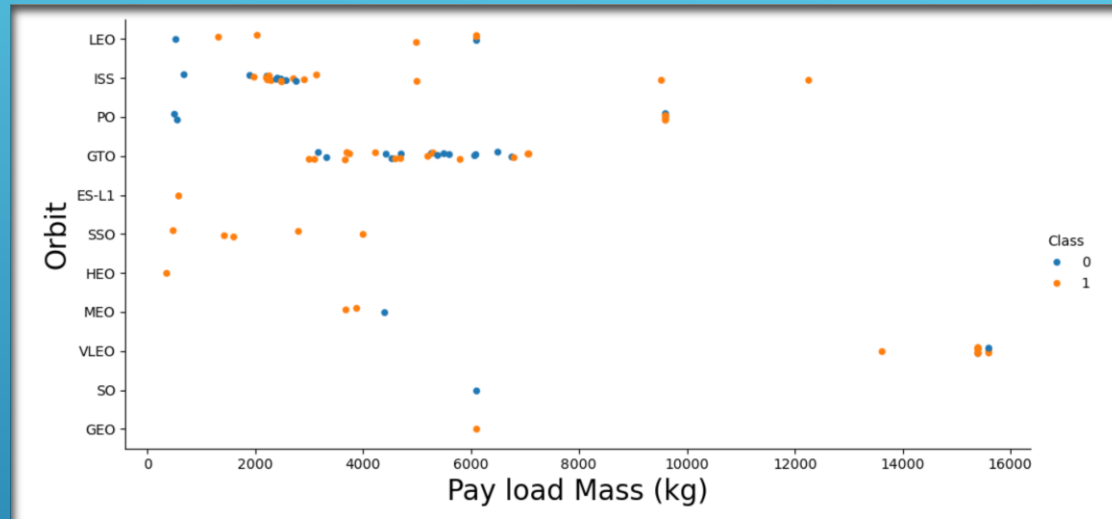
## SUCCESS RATE VS. ORBIT TYPE

- ▶ With the plot we see rate for different Orbit types
- ▶ ES-L1, GEO, HEO and SSO have the best/perfect success rate

- ▶ Success rate increases with flight number on LEO orbit
- ▶ GTO has no correlation between success rate and flight numbers
- ▶ High success rate of HEO and SSO could be due to knowledge learned during early launches of other orbits

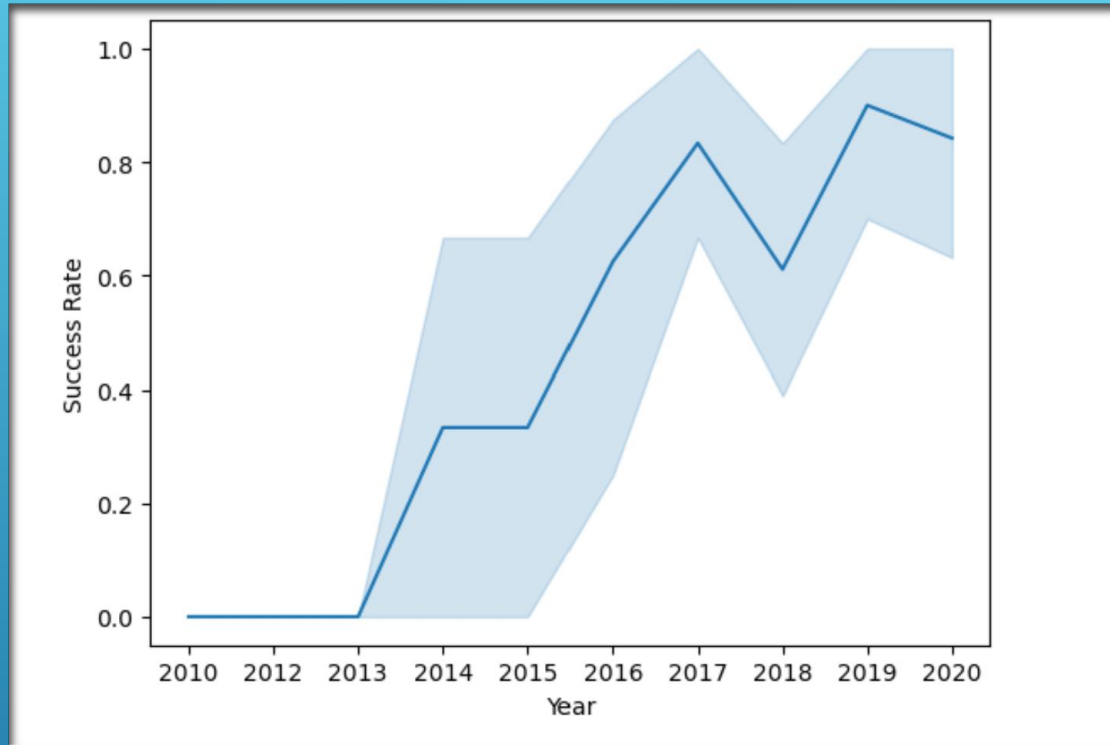
## FLIGHT NUMBER VS. ORBIT TYPE





- ▶ 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.

## PAYLOAD VS. ORBIT TYPE



- ▶ This plot shows that from 2013 to 2020 the success rate made solid improvements
- ▶ The first 3 years were a period of adjustments

## LAUNCH SUCCESS YEARLY TREND

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

```
[10]: %sql SELECT DISTINCT "Launch_Site" FROM SPACEXTBL;
```

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

```
[10]: Launch_Site
```

```
CCAFS LC-40
```

```
VAFB SLC-4E
```

```
KSC LC-39A
```

```
CCAFS SLC-40
```

- ▶ To find the names of all Site Names, we used the key word DISTINCT to show unique launch sites in our results for SpaceX data

# ALL LAUNCH SITE NAMES

```
[13]: %sql SELECT * FROM SPACEXTBL WHERE "Launch_Site" LIKE '%CCA%' LIMIT 5;
```

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

Done.

```
[13]:
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- We used the Query below to narrow our search down to sites beginning with CCA – attached is the list

# LAUNCH SITE NAMES BEGIN WITH 'CCA'

# TOTAL PAYLOAD MASS

- ▶ We used the below Query to find the total Payload Mass
- ▶ Payload Mass is displayed below the Query

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

```
[14]: %sql SELECT SUM("PAYLOAD_MASS_KG_") AS TOTAL_PAYLOAD FROM SPACEXTBL WHERE Customer LIKE "NASA (CRS)";
```

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

Done.

```
[14]: TOTAL_PAYLOAD
```

```
45596
```

```
Task 4
Display average payload mass carried by booster version F9 v1.1

[15]: %sql SELECT AVG("PAYLOAD_MASS_KG") AS AVERAGE_PAYLOAD_MASS FROM SPACEXTBL WHERE "Booster_Version" LIKE "F9 v1.1";
* sqlite:///my_data1.db
Done.
[15]: AVERAGE_PAYLOAD_MASS
      2928.4
```

- ▶ Above is the query used to calculate the Average Payload Mass for F9 v1.1
- ▶ We can see that the average payload comes out at – 2928.4

# AVERAGE PAYLOAD MASS BY F9 V1.1

# FIRST SUCCESSFUL GROUND LANDING DATE

- ▶ We observed that the 1<sup>st</sup> grounding date was 22-12-2015
- ▶ We got to this using the attached Query

```
In [9]: %sql select min(Date) from SPACEXTABLE where Landing_Outcome = 'Success (ground pad)'
* sqlite:///my_data1.db
Done.
In [9]: min(Date)
2015-12-22
```

# SUCCESSFUL DRONE SHIP LANDING WITH PAYLOAD BETWEEN 4000 AND 6000

- Here we used the WHERE clause to filter for Boosters, which have successfully landed on drone ship and applied the AND condition to determine successful landing with payload mass greater than 4000, but less than 6000

```
] : %sql select Booster_Version from SPACEXTABLE\  
where Landing_Outcome = "Success (drone ship)" and PAYLOAD_MASS_KG between 4000 and 6000;  
  
* sqlite:///my_data1.db  
Done.  
]  
Booster_Version  
-----  
F9 FT B1022  
F9 FT B1026  
F9 FT B1021.2  
F9 FT B1031.2
```

```
: %sql SELECT (SELECT COUNT("Mission_Outcome") FROM SPACEXTBL WHERE "Mission_Outcome" LIKE '%Success%') AS SUCCESSFUL, \
(SELECT COUNT("Mission_Outcome") FROM SPACEXTBL WHERE "Mission_Outcome" LIKE '%Failure%') AS FAILURE;
```

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

Done.

```
: SUCCESSFUL  FAILURE
```

---

100	1
-----	---

Here we used a % to filter for WHERE MissionOutcome was a success or a failure

```
: %sql SELECT "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
```

- We determined the boosters that have carried the maximum payload – using a subquery in the WHERE clause and the MAX() function

# 2015 LAUNCH RECORDS

- Here we used a combination of the WHERE clause, and filter for failed landing outcomes in the drone ship, their booster versions, and launch site names for 2015

```
] : %sql select substr(Date, 6, 2) as month, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE\  
where substr(Date, 1, 4) = '2015' and Landing_Outcome = 'Failure (drone ship)'
```

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

```
Done.
```

```
] : month Landing_Outcome Booster_Version Launch_Site
```

month	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

```
from SPACEXTABLE\  
where Date between '2010-06-04' and '2017-03-20'\  
group by Landing_Outcome\  
order by count_outcomes Desc;
```

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

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

## RANK LANDING OUTCOMES BETWEEN 2010- 06-04 AND 2017-03-20

- ▶ We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2017-03-20.
- ▶ We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

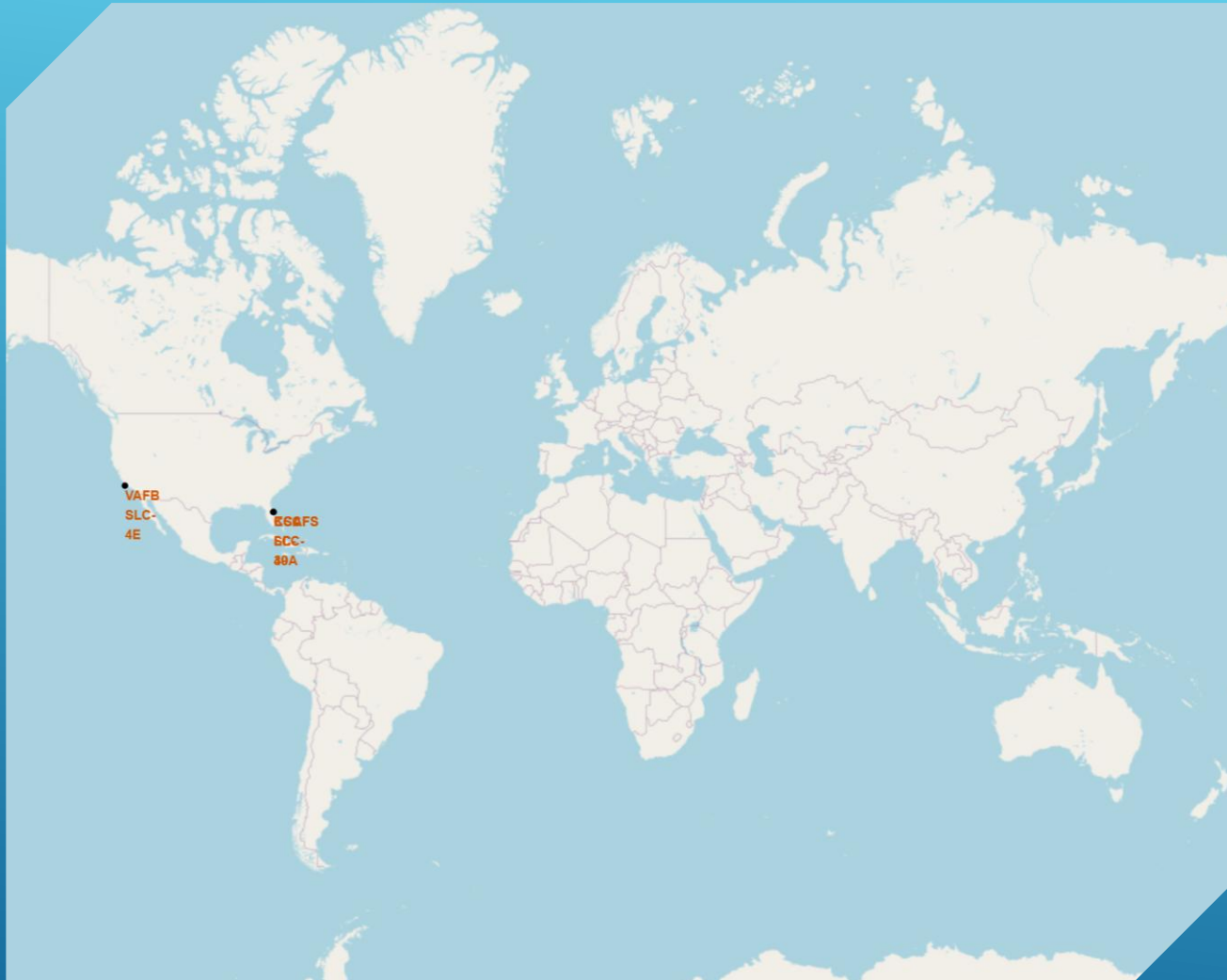
The background of the slide is a high-quality satellite image of Earth taken from space. The image shows the dark blue of the night sky above the horizon, with the bright blue of the Earth's atmosphere and oceans below. Numerous city lights are visible as glowing yellow and orange spots, particularly concentrated in the lower right quadrant. The horizon line curves across the middle of the frame. In the bottom right corner, there are several thin, white, parallel diagonal lines that add a modern, graphic touch to the design.

Section 3

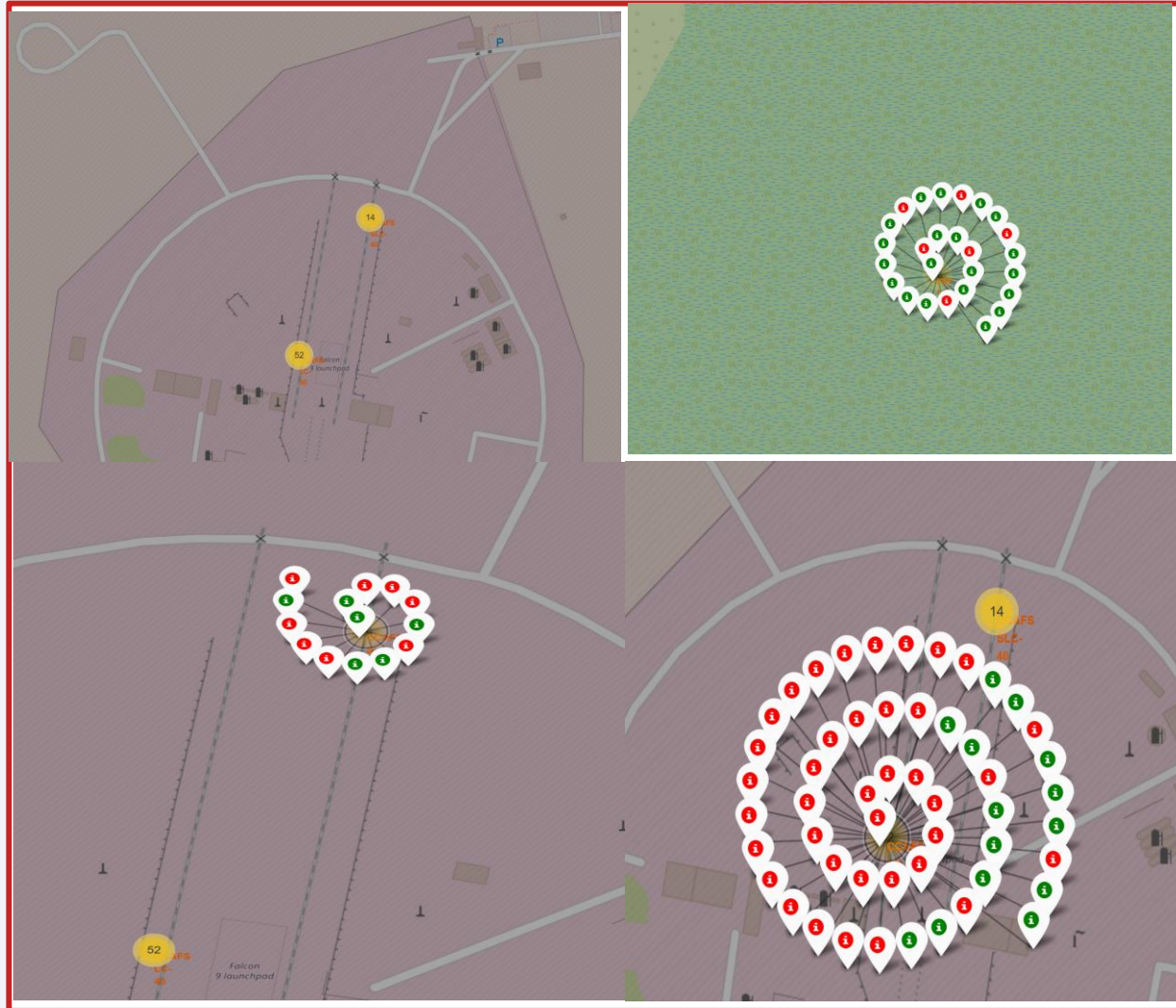
# Launch Sites Proximities Analysis

# GLOBAL LAUNCH SITES FOR SPACEX

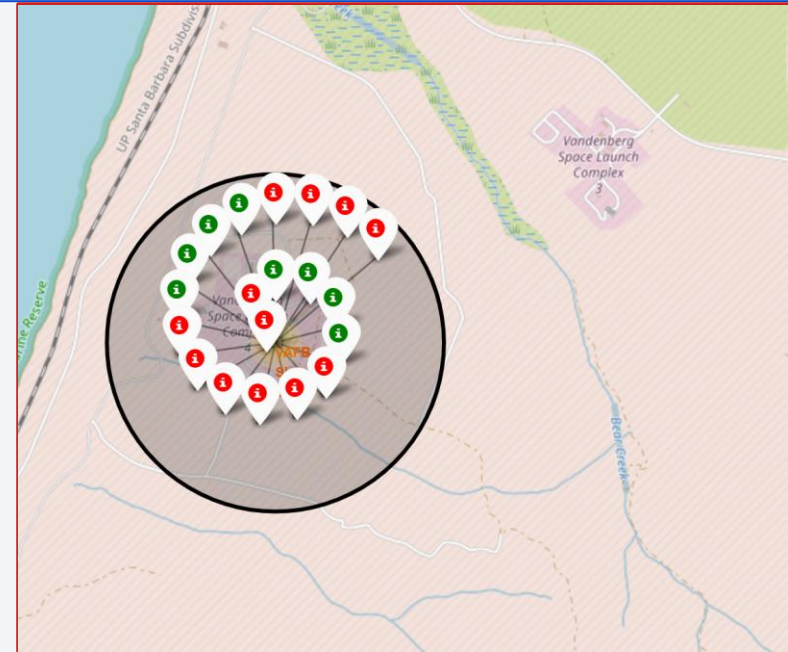
- From this world map, we can see that SpaceX launch sites are only in the United States of America – specifically Florida and California



# Markers showing Launch sites with Colour labels

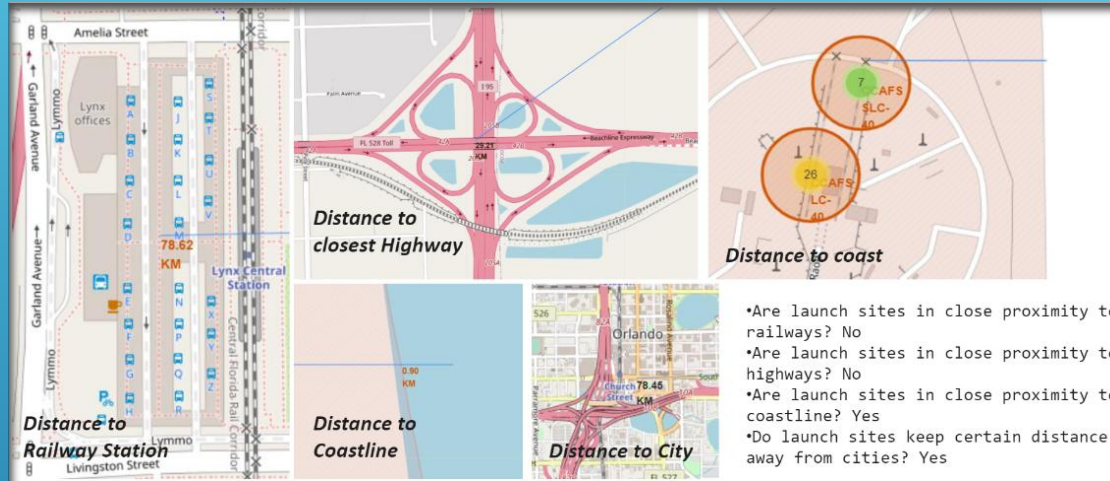


Florida Launch Sites



Californian Launch Site

- ▶ **Green Markers** indicate successful launches
- ▶ **Red Markers** indicate failures



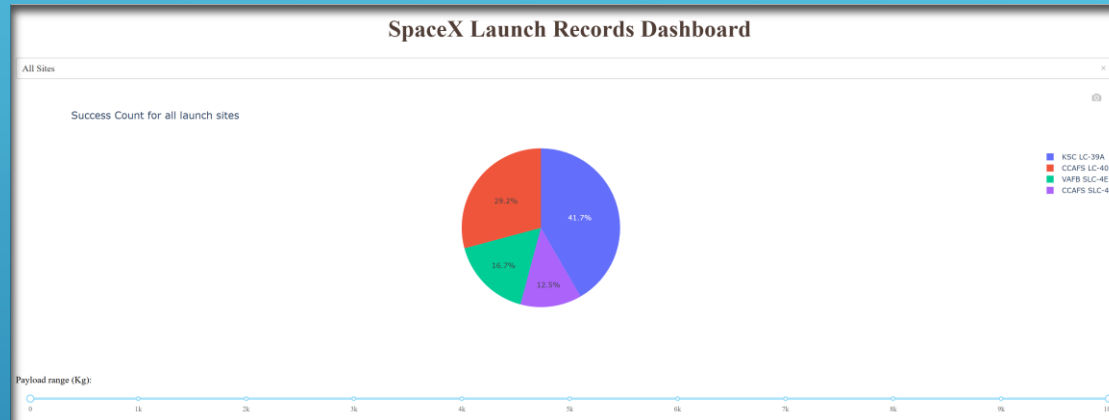
- The main takeaway we see is that Launch sites are kept away from Cities
- They are not specifically away from Railways, Highways, or Coastlines

# LAUNCH SITES DISTANCE TO LANDMARKS



Section 4

# Build a Dashboard with Plotly Dash



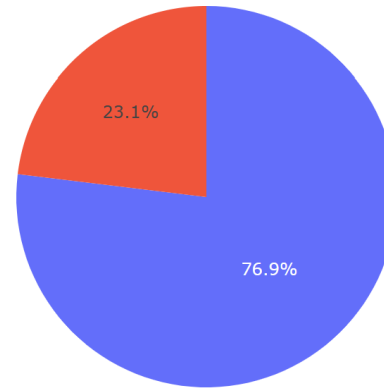
- ▶ Here we can see a breakdown of the SpaceX Launches by sites
- ▶ KSC LC-39A being the most successful overall site

PIECHART SHOWING SUCCESS RATIO  
FOR EACH LAUNCH SITE

# SpaceX Launch Records Dashboard

KSC LC-39A

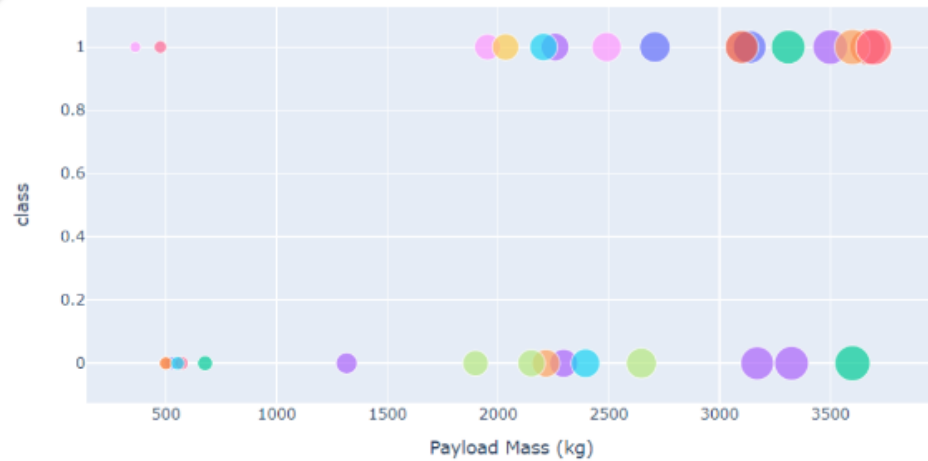
Total Success Launches for site KSC LC-39A



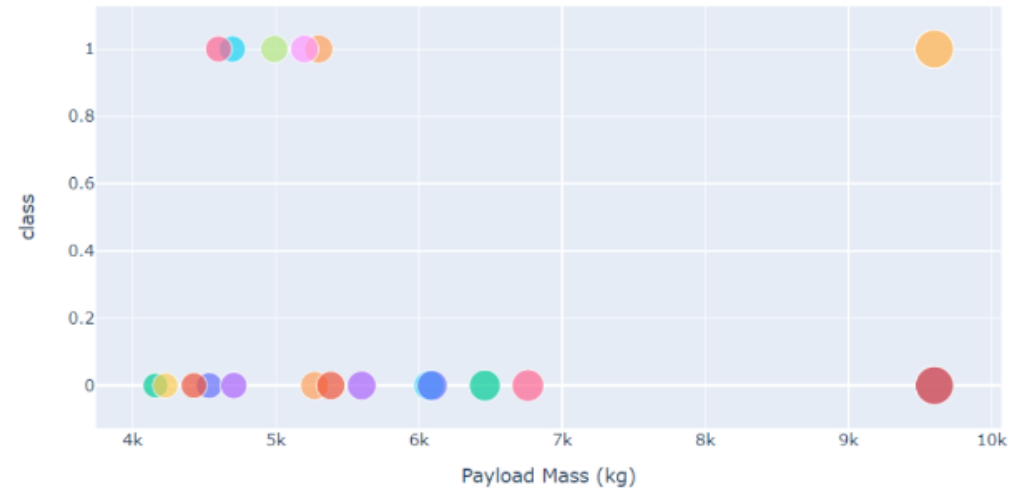
Pie chart  
showing  
highest  
success from  
a Launch Site

- ▶ KSC LC-39A had then highest success of a launch site – completing 76.9% of all launches

**Low Weighted Payload 0kg – 4000kg**



**Heavy Weighted Payload 4000kg – 10000kg**



*We can see the success rates for low weighted payloads is higher than the heavy weighted payloads*

# SCATTER PLOT OF PAYLOAD VS LAUNCH OUTCOMES

Section 5

# Predictive Analysis (Classification)

Find the method performs best:

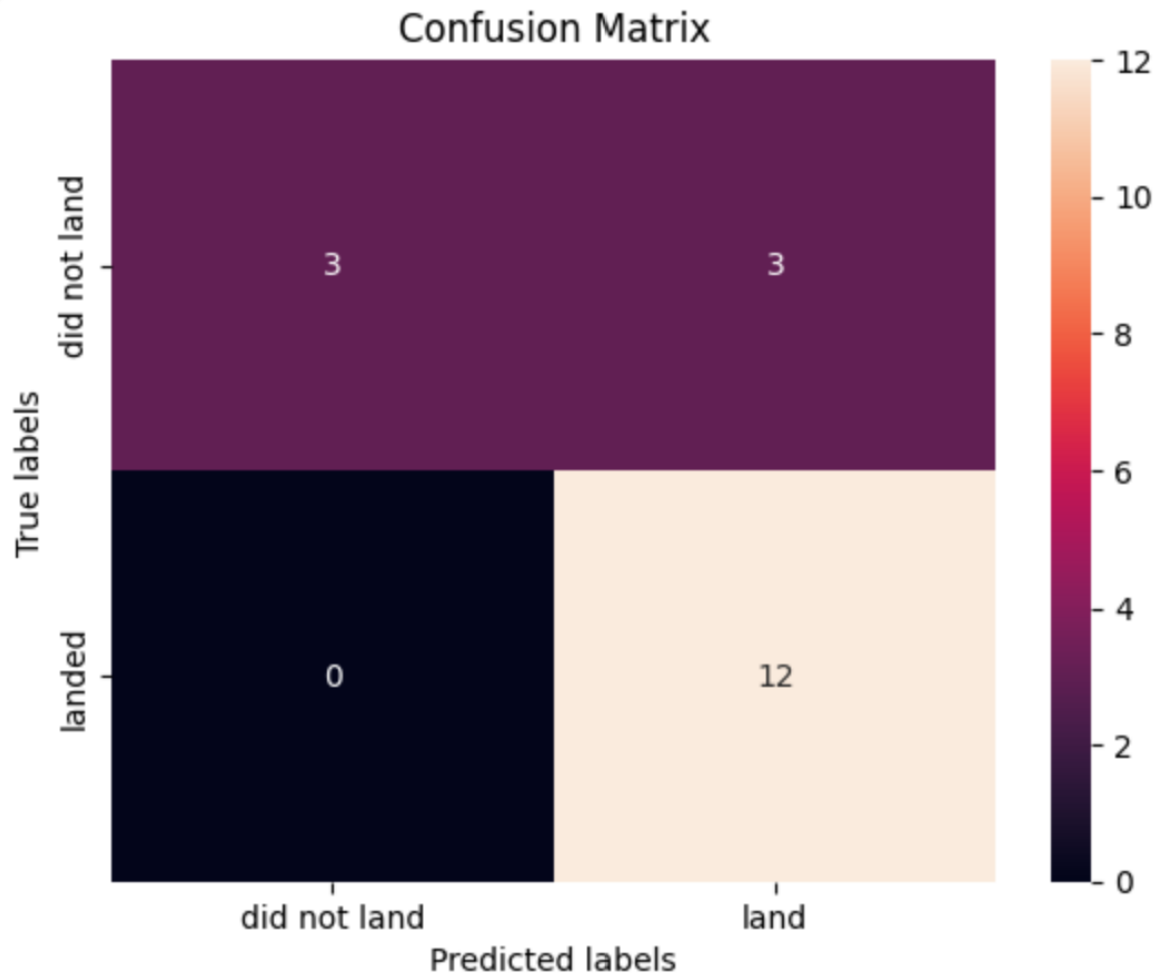
```
] models = {'KNeighbors': knn_cv.best_score_,
            'DecisionTree': tree_cv.best_score_,
            'LogisticRegression': logreg_cv.best_score_,
            'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm, 'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

Best model is DecisionTree with a score of 0.8732142857142856  
Best params is : {'criterion': 'gini', 'max\_depth': 6, 'max\_feat'

## CLASSIFICATION ACCURACY

- ▶ The Query above, picks out the best performing algorithm for classification
- ▶ We see that Decision Tree classifier is the most accurate model



## CONFUSION MATRIX

- ▶ Here we see the Confusion Matrix for the Decision Tree classifier – which was the most accurate classification
- ▶ The matrix shows that we can distinguish between different classes
- ▶ However one huge drawback, is that we get false positives – whereby unsuccessful landings could be classified as successful

# CONCLUSIONS

- ▶ We can conclude that:
- ▶ Launch site success rate increased between 2013 – 2020
- ▶ The more flights at a Launch site, the more success it ends up having at a site
- ▶ Orbits ES-L1, GEO, HEO, SSO and VLEO had the highest success rates
- ▶ KSC LC-39A had the most successful launches of all the sites
- ▶ The Decision Tree classifier was the best/most accurate Machine learning algorithm on this task

Thank you!

