IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Juan Cristóbal Rivera

December 23rd, 2025

Outline

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

Executive Summary

- SpaceX launch data was collected using the SpaceX API, as well as through Web Scraping the Wikipedia page of SpaceX launches.
- Data wrangling techniques were used to obtain a single feature which indicates whether a launch was successful or a failure.
- SQL and data visualization were used for Exploratory Data Analysis (EDA), to try to see patterns and trends in the data.
- Folium was used to visualize geospatial information about the launches, while a Dashboard was created to display the data interactively.
- Various ML models were trained and tested, and the Decision Tree Classifier was found to give the best results, with a score of 94.4%

Introduction

- In this hypothetical project, we're a competitor of the aerospace company SpaceX, and we want to develop our own line of self-landing rocket boosters.

- Instead of rocket science, we want to use Machine Learning to develop a model that will predict whether a booster landing will be successful or not based on things like Booster Version, Launch Site, Orbit, Payload Mass, and more.

- For this purpose, we will analyze the data of previous SpaceX launches. This data will be used to develop the model that predicts whether a booster will successfully land, or if it will fail to do so.

Section 1

Methodology

Methodology

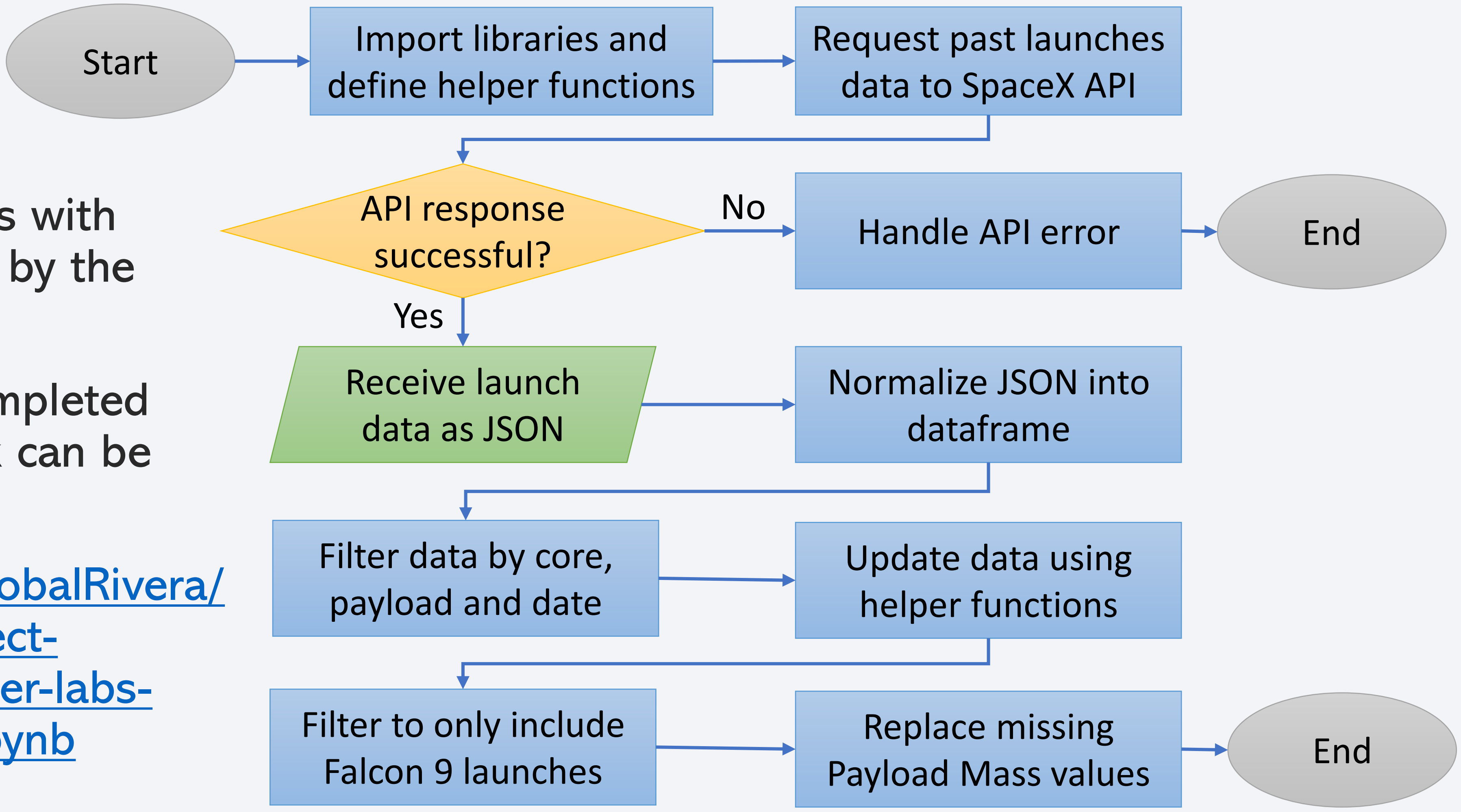
Executive Summary

- Data collection methodology:
  - SpaceX API
  - Wikipedia web scraping using BeautifulSoup
- Data wrangling methodology:
  - Landing outcome preprocessing
- Exploratory Data Analysis (EDA) methodology:
  - Visualization EDA
  - SQL EDA
- Interactive visual analytics methodology:
  - Folium
  - Plotly Dash
- Predictive analysis methodology:
  - Building, training and evaluating classification models using scikit-learn

Data Collection

- Two data collection approaches are used:
  - In the first approach, the data is collected through the use of an API using the **requests** library, which connects to the SpaceX data repositories using a .get() request, and returns the requested information as a json. The json is normalized into a dataframe and further processed until the desired data format is reached.
  - In the second approach, the **BeautifulSoup** library is used to perform web scraping on the Wikipedia page about SpaceX launches. The information on the html is parsed into a dataframe with the desired structure.

Data Collection – SpaceX API



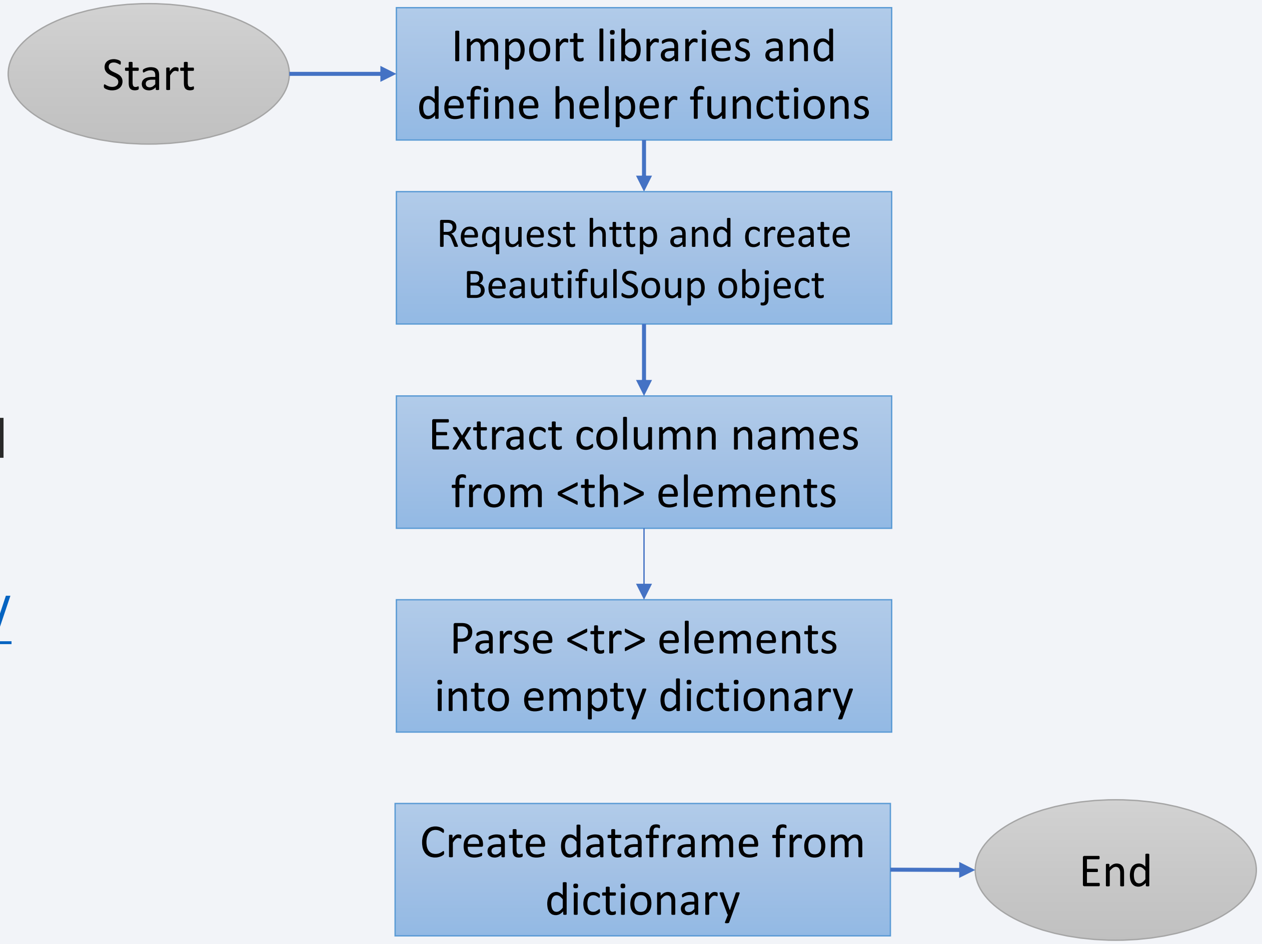
- The data collection process with SpaceX API is represented by the flowchart shown.
- The GitHub URL of the completed SpaceX API calls notebook can be found here:

https://github.com/JuanCristobalRivera/Data-Science-capstone-project-SpaceX/blob/main/1-1-jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

- The data collection process using web scraping is represented by the flowchart shown.
- The GitHub URL of the completed web scraping notebook can be found here:

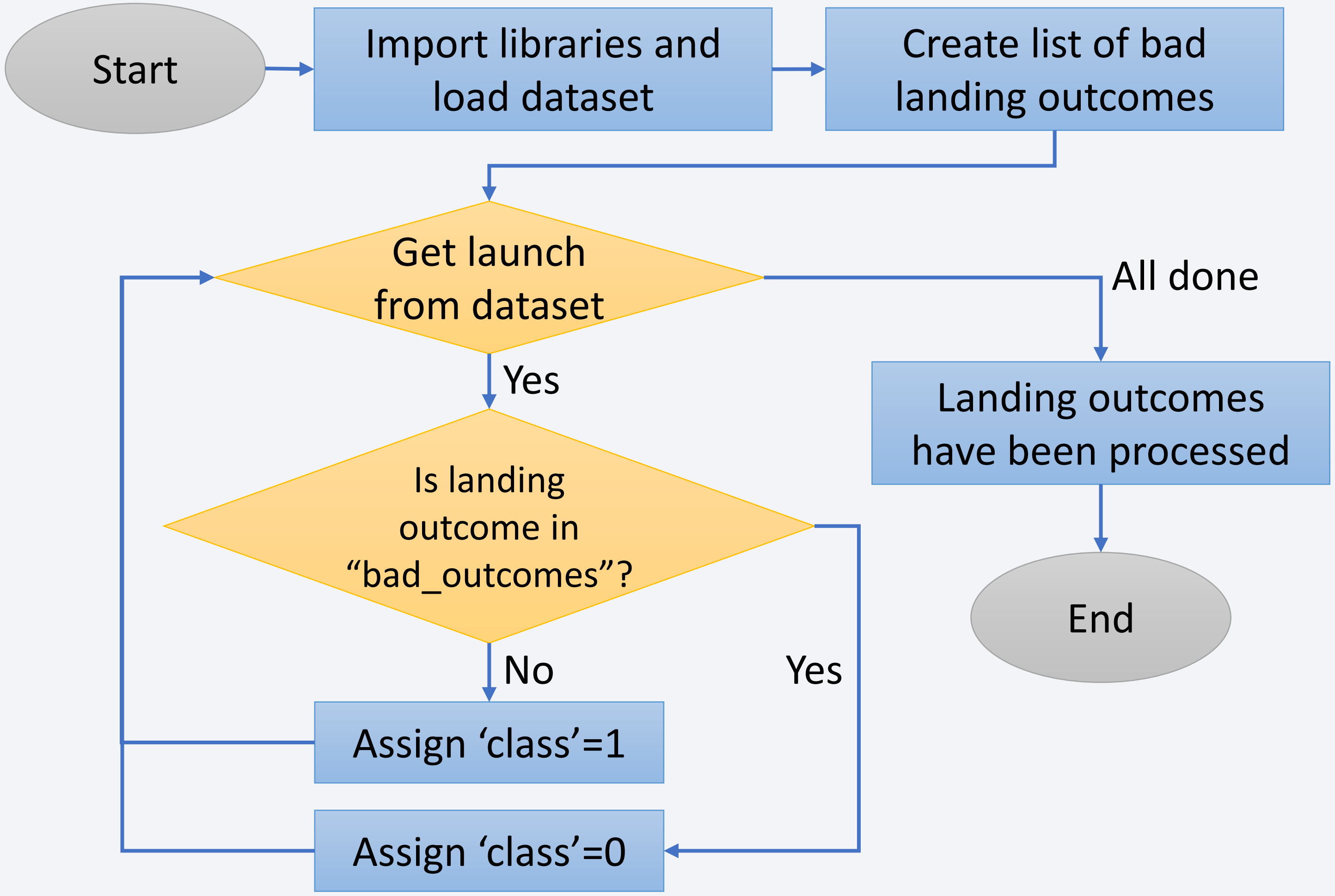
https://github.com/JuanCristobalRivera/Data-Science-capstone-project-SpaceX/blob/main/1-2-jupyter-labs-webscraping.ipynb



Data Wrangling

- Before data wrangling a brief data exploration was done:
  - The number of launches on each site were listed.
  - The orbit types of the missions were listed.
  - The totals of each of the landing outcomes of the dataset were listed.
- The data wrangling process is represented by the flowchart shown.
- The GitHub URL of the completed data wrangling notebook can be found here:

https://github.com/JuanCristobalRivera/Data-Science-capstone-project-SpaceX/blob/main/1-3-labs-jupyter-spacex-data-wrangling.ipynb



EDA with Data Visualization

- For the first stage of EDA, the library **seaborn** was used to plot and study the relationships between different variables of the dataset, which included:
  - Scatter plot of Flight Number vs Launch Site (slide 18)
  - Scatter plot of Payload Mass vs Launch Site (slide 19)
  - Bar chart of Success Rate by Orbit Type (slide 20)
  - Scatter plot of Flight Number vs Orbit Type (slide 21)
  - Scatter plot of Payload Mass vs Orbit Type (slide 22)
  - Line plot of Success Rate by Year (slide 23)
- The GitHub URL of the completed Data Visualization EDA notebook can be found here:

https://github.com/JuanCristobalRivera/Data-Science-capstone-project-SpaceX/blob/main/2-2-jupyter-labs-eda-data-viz.ipynb

EDA with SQL

- In the second stage of EDA the library **sqlite3** was used to perform SQL queries on the database, in order to understand it better. These queries included:
  - Displaying the names of the unique launch sites (slide 24)
  - Displaying 5 record where the launch site begins with ‘CCA’ (slide 25)
  - Displaying the total mass carried by boosters launched by ‘NASA (CRS)’ (slide 26)
  - Displaying the average Payload Mass carried by Booster Version F9 1.1 (slide 27)
  - Listing the date of the first successful landing in a ground pad (slide 28)
  - Listing the names of the boosters with a successful landing on a drone ship that carried a payload mass between 4000 and 6000 kgs (slide 29)
  - Listing the total number of successful and failure outcomes (slide 30)
  - Listing all the Booster Versions that have carried the maximum payload using a subquery (slide 31)
  - Listing all the records with a failed landing outcome on a drone ship that occurred in 2015, including Month, Booster Version, and Launch Site (slide 32)
  - Ranking the landing outcomes between 2010-06-04 and 2017-03-20 in descending order (slide 33)
- The GitHub URL of the completed SQL EDA notebook can be found here:

https://github.com/JuanCristobalRivera/Data-Science-capstone-project-SpaceX/blob/main/2-1-jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- In order to get a geospatial representation of the Launch Sites and their landing success rates, we used the library folium.
- The map objects added to the folium map included:
  - Circles, and markers showing each of the Launch Sites
  - Green and red markers at each Launch Site to represent the successful and failed landings respectively
  - Line and markers with the distance to the various proximities of a Launch Site
- The GitHub URL of the completed Folium notebook can be found here:

https://github.com/JuanCristobalRivera/Data-Science-capstone-project-SpaceX/blob/main/3-1-jupyter-labs-launch-site-location.ipynb

Build a Dashboard with Plotly Dash

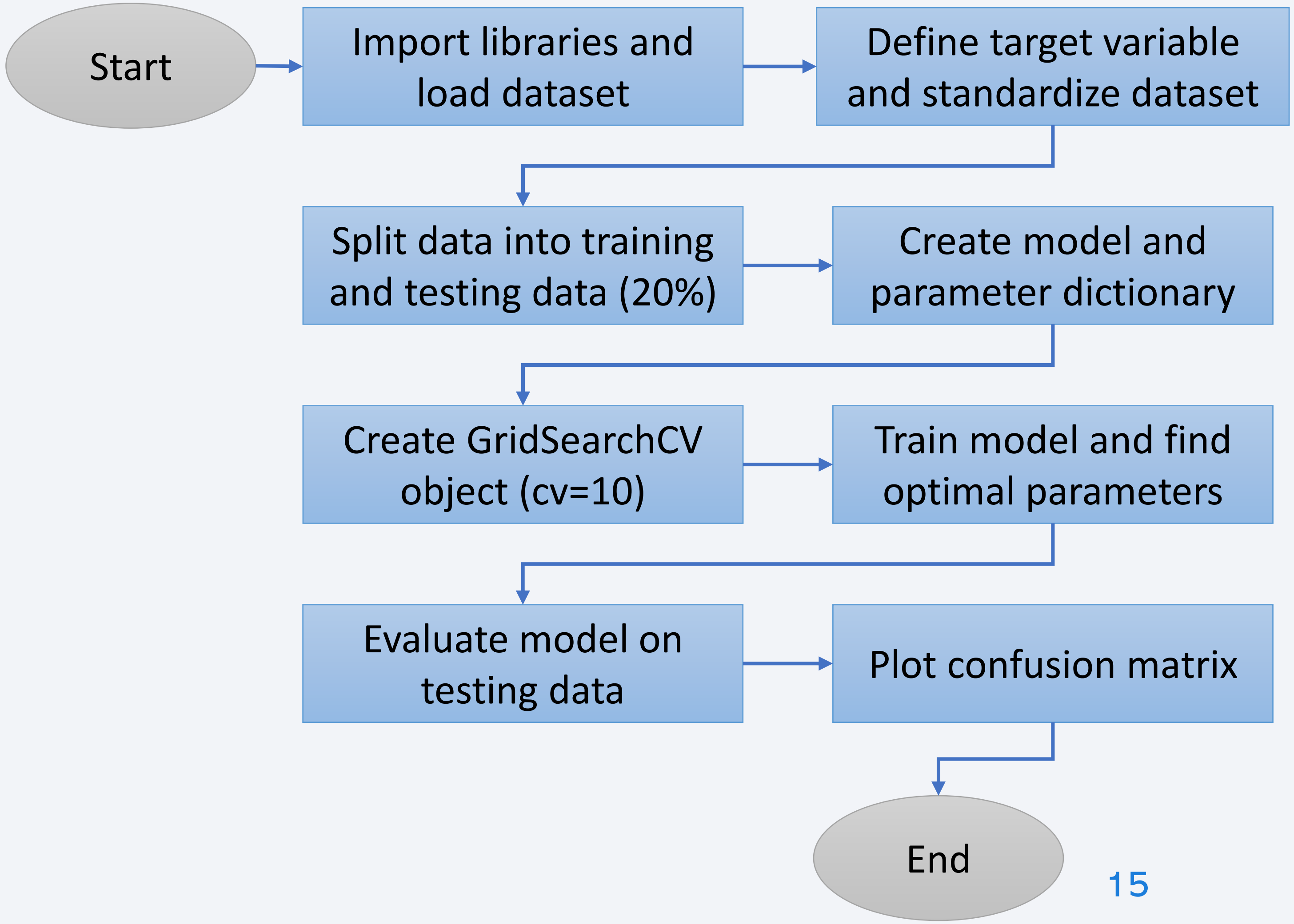
- In order to study the variables of the dataset in an interactive way, a Dashboard was build using **Plotly Dash**.
- The dashboard includes features such as:
    - A dropdown list to select between all launch sites, and each launch site individually
    - A pie chart showing total successful landings per landing site, or number of successes and failures in the case of an individual site
    - A slider to select range for payload mass
    - A scatter chart showing correlation between Payload Mass and landing success.
- The GitHub URL of the completed SQL EDA notebook can be found here:

https://github.com/JuanCristobalRivera/Data-Science-capstone-project-SpaceX/blob/main/3-2-jupyter-labs-spacex-dash-app.py

Predictive Analysis (Classification)

- After the data preparation and EDA steps, the library **scikit-learn** was used to build, train and evaluate various classification models, including:
    - Logistic regression
    - Support vector machine (SVM)
    - Decision tree classifier
    - K-nearest neighbors
- The process for each model is shown in the flowchart.
- The GitHub URL of the completed predictive analysis notebook can be found here:

https://github.com/JuanCristobalRivera/Data-Science-capstone-project-SpaceX/blob/main/4-1-jupyter-labs-SpaceX-Machine-Learning-Prediction.ipynb



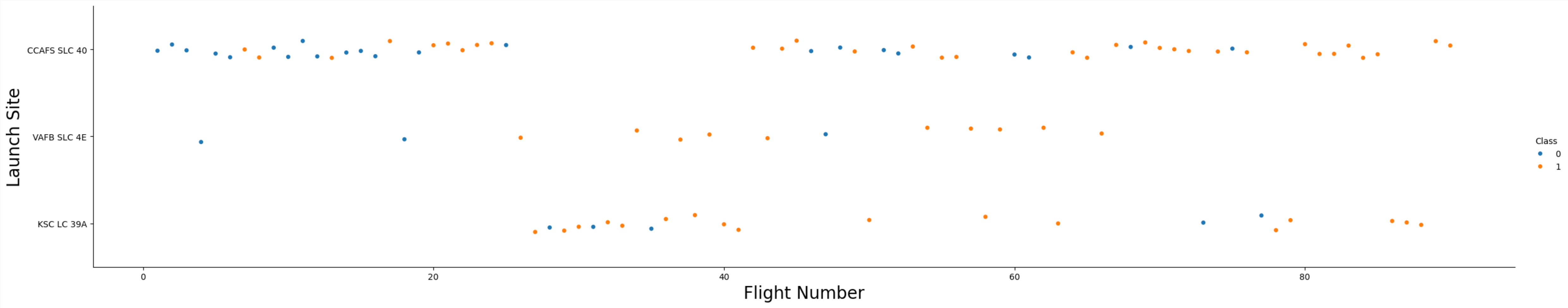
Results

- The following slides will shows the results and insight obtained from:
  - Exploratory Data Analysis, which includes the Matplotlib/Seaborn charts and plots, as well as the SQL queries and their results.
  - Interactive Visual Analytics, which includes the Geospatial information using Folium, as well as the interactive data display using Dashboard.
  - Predictive Analysis, which includes the scores from each model trained, as well as the best classifier model to predict the result of a booster landing.

Section 2

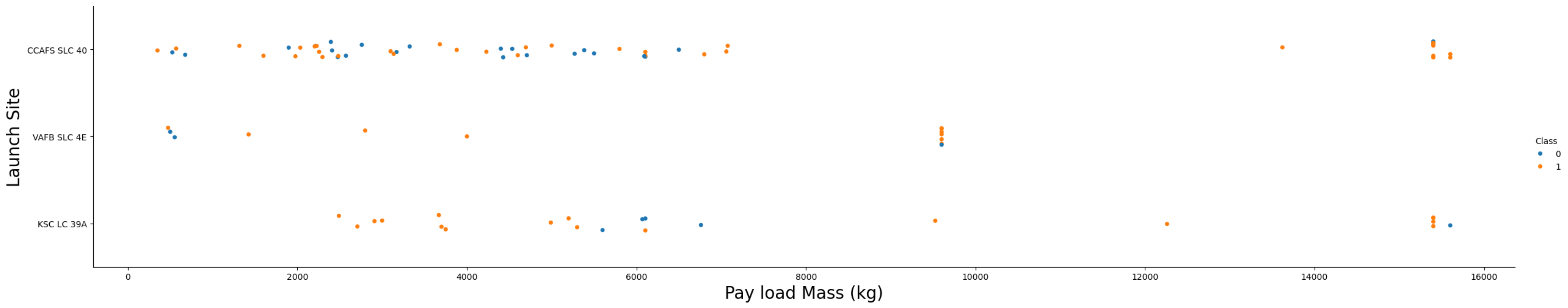
Insights drawn from EDA

Flight Number vs. Launch Site



- Scatter plot of Flight Number vs. Launch Site, where orange color indicates a successful booster landing, and blue indicates a failure.
- The majority of the flights were launched from site CCAFS SLC 40, with the exception of flights between 27 to 42 (approximately), which were mostly launched from KSC LC 39A.
- While an upwards trend of landing success vs flight number can be observed, the success rate of each site seems to be similar to the others.

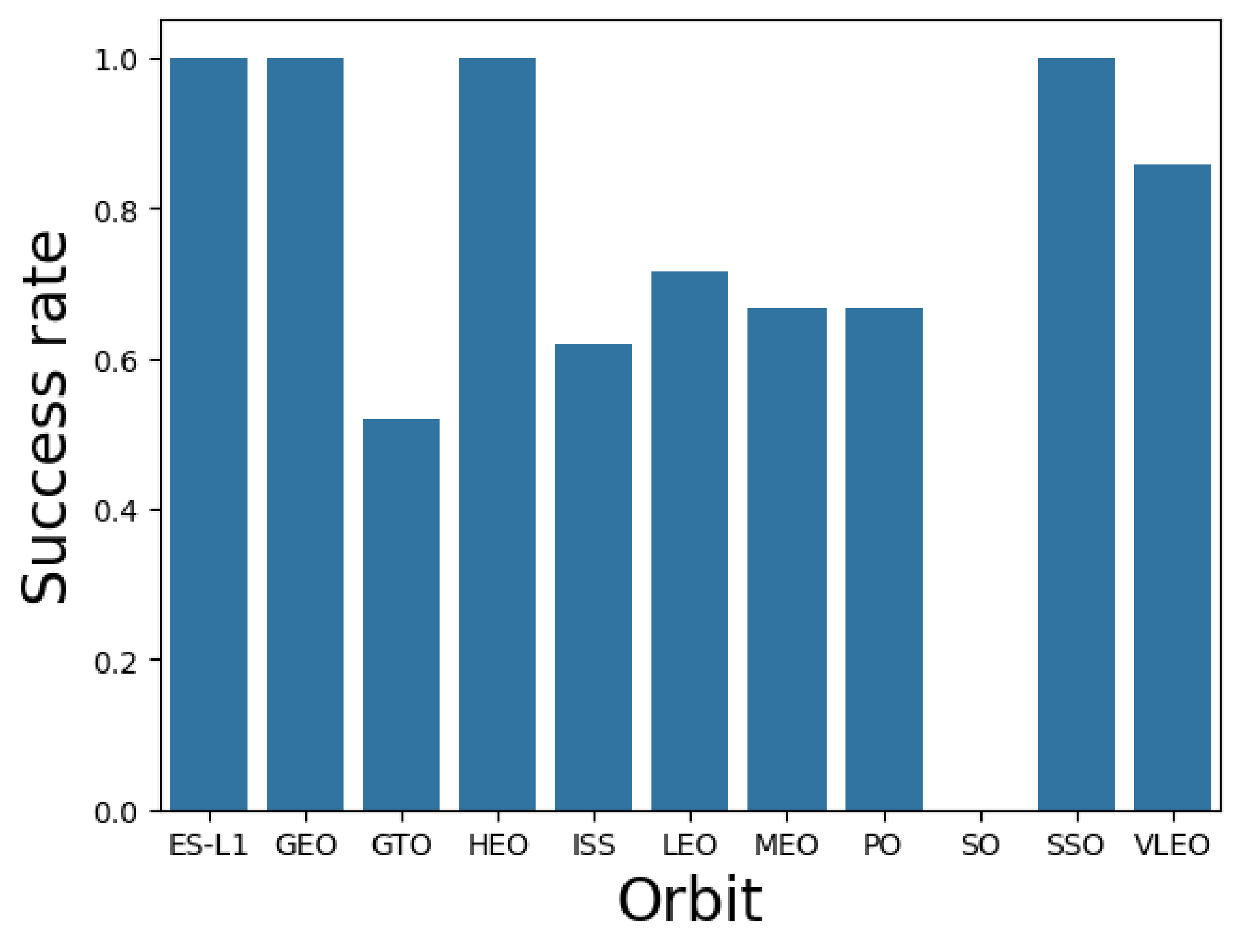
Payload vs. Launch Site



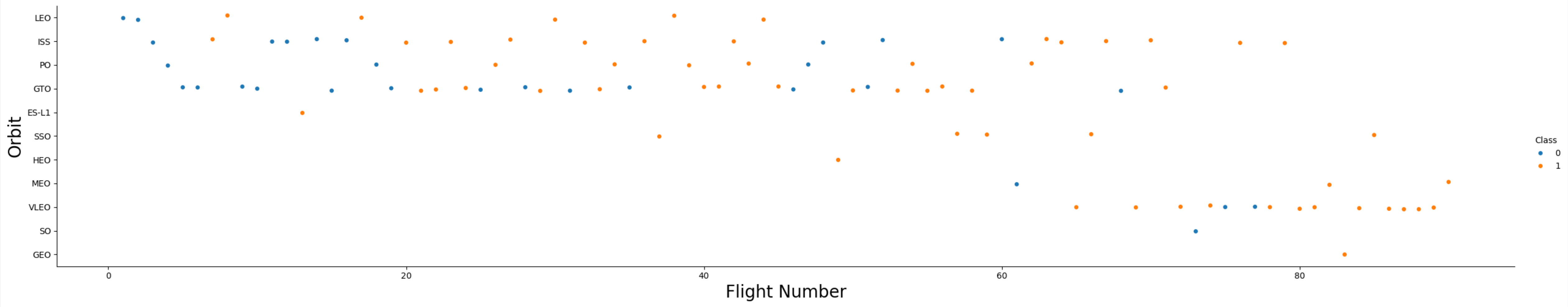
- Scatter plot of Payload Mass vs Launch Site, where orange color indicates a successful booster landing, and blue indicates a failure.
- There seems to be a weight limit for site VAFB SLC 4E, since its heaviest payload is under 10000 kg.
- For sites CCAFS SLC 40 and KSC LC 39A, there is a wide gap in the Payload Mass distribution between about 8000 kgs to 15000 kgs, with only 3 missions in that range.

Success Rate vs. Orbit Type

- Bar chart for the success rate of each orbit type.
- ES-L1, GEO, HEO and SSO orbits have a 100% success rate, while the SO orbit has a 0% success rate: this can be explained by their small sample size (most have just 1 launch, only SSO has 5 launches).
- GTO, ISS and VLEO have a larger sample size, which means a smaller variance, with success rates ranging from 50% to about 85%.

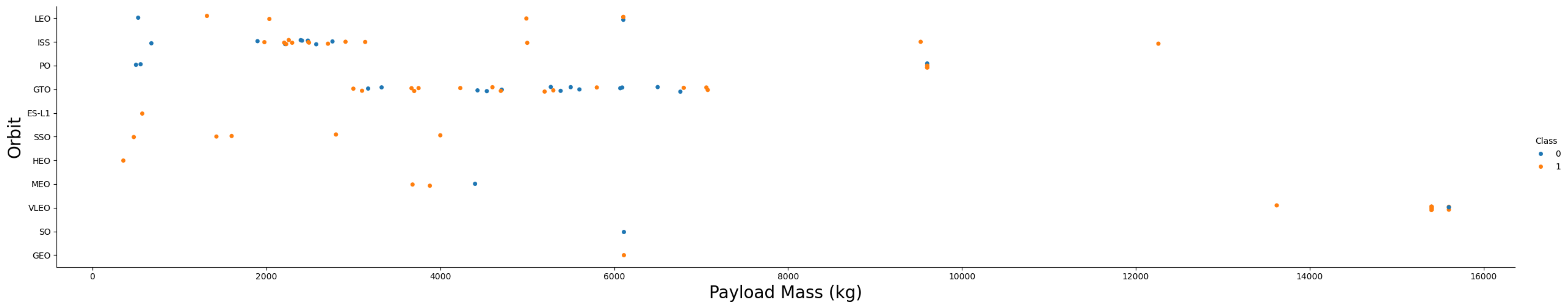


Flight Number vs. Orbit Type



- Scatter plot of Flight number vs. Orbit type, where orange color indicates a successful booster landing, and blue indicates a failure.
- From flight number 1-60 most launches had an LEO, ISS, PO and GTO orbit type, afterwards most launches had a VLEO orbit type.
- As mentioned in the previous slide, ES-L1, GEO, HEO and SO have only one launch each.

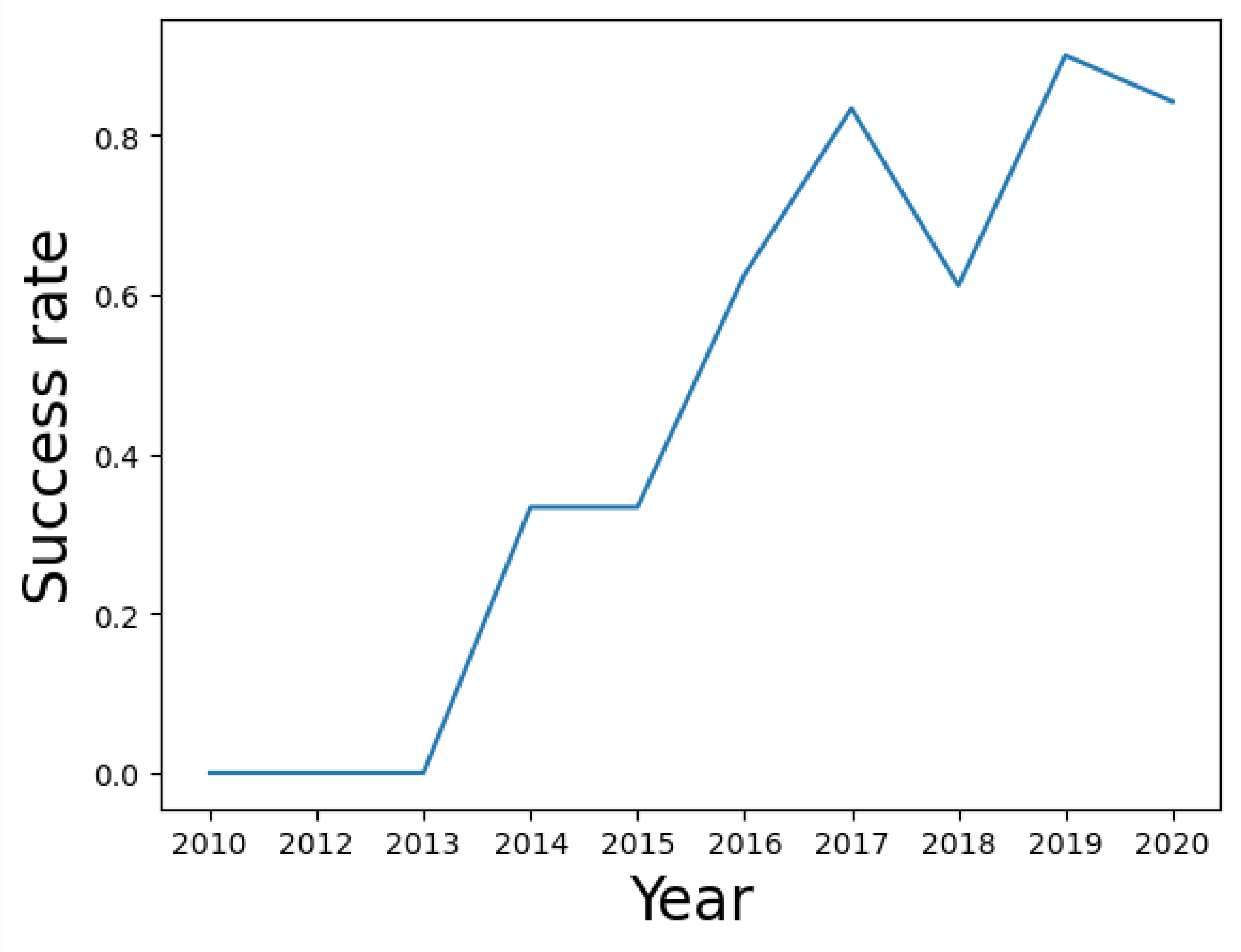
Payload vs. Orbit Type



- Scatter plot of payload vs. orbit type, where orange color indicates a successful booster landing, and blue indicates a failure.
- We can see that most launches with ISS orbit had a Payload Mass between 2000 and 3000 kgs, while most launches with a GTO orbit had a Payload Mass between 3000 and 7000 kgs.

Launch Success Yearly Trend

- Line chart of yearly average success rate
- We see a fairly steady rise from 2013 with 0% success, to over 80% success by 2019.



All Launch Site Names

- The query used to get all the names of the launch sites in the database is:

%sql select distinct Launch_Site from SPACEXTABLE

- The column 'Launch_Site' contains the launch site of each mission, so by adding the keyword **distinct**, we are able to get a list with all the unique launch sites in the database.

```
%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'

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

Python

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

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

- The query to get 5 records where launch sites begin with `CCA` is:

%sql select * from SPACEXTABLE where Launch_Site like ‘CCA%’ limit 5

- Adding **where Launch_Site like ‘CCA%’** to the query selects the sites that begin with CCA, while **limit 5** gives us only 5 records.

Total Payload Mass

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer = 'NASA (CRS)'

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

sum(PAYLOAD_MASS__KG_)
45596

- The query to calculate the total payload carried by boosters from NASA is:

%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer = ‘NASA (CRS)’

- By adding **where Customer = ‘NASA (CRS)’** we are making sure we select only missions from NASA, while using the **sum()** function gives us the total payload like we wanted.

Average Payload Mass by 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

- The query to calculate the average payload mass carried by booster version F9 v1.1 is:

%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version like ‘F9 v1.1%’

- Where we use the expression **where Booster_Version like ‘F9 v1.1%’** to select only the launches that used the F9 v1.1 booster, and the function **avg()** calculates the average.

First Successful Ground Landing Date

```
%sql select min(Date) from SPACEXTABLE where Landing_Outcome = 'Success (ground pad)'

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

min(Date)
2015-12-22

- The query to find the date of the first successful landing outcome on ground pad is:

%sql select min(Date) from SPACEXTABLE where Landing_Outcome = ‘Success (ground pad)’

- Where the expression **where Landing_Outcome = ‘Success (ground pad)’** selects only the launches with a successful landing outcome on a ground pad, while the function **min()** is used to find the earliest date this happened.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select distinct 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

- The query to list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 is:

%sql select distinct Booster_Version from SPACEXTABLE where Landing_Outcome = ‘Success (drone ship)’ and PAYLOAD_MASS__KG_ between 4000 and 6000

- The expression **where Landing_Outcome = ‘Success (drone ship)’** selects only launches with a successful landing on a drone ship, the expression **PAYLOAD_MASS__KG_ between 4000 and 6000** selects launches with payloads within 4000 and 6000 kgs, and the keyword **distinct** avoids repeats.

Total Number of Successful and Failure Mission Outcomes

```
%sql select Mission_Outcome, count(*) as 'Total' from SPACEXTABLE group by Mission_Outcome

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

Mission_Outcome	Total
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- The query to calculate the total number of successful and failure mission outcomes is:

%sql select Mission_Outcome, count(*) as ‘Total’ from SPACEXTABLE group by Mission Outcome

- The expression **group by Mission_Outcome** allows us to group the rows by mission outcome and also allows us to use the aggregate function **count()**, which counts the rows in each grouping.

Boosters Carried Maximum Payload

- The query to list the names of the booster which have carried the maximum payload mass is:

%sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)

- Where we used a subquery to find the maximum payload, and then a condition to select only boosters with that same payload value.

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

```
 * 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, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE \
where substr(Date,0,5)='2015' and Landing_Outcome = 'Failure (drone ship)'
```

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

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

- The query to list the failed landing_outcomes in drone ship, their booster versions, and launch site names in the year 2015 is:

%sql select substr(Date,6,2) as Month, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE where substr(Date,0,5) = '2015' and Landing_Outcome = 'Failure (drone ship)'

- The expressions **substr(Date,6,2)** and **substr(Date,0,5)** are used to get the Month and the Year respectively, while the expression **where substr(Date,0,5) = '2015' and Landing_Outcome = 'Failure (drone ship)'** selects missions in 2015 that had a failed landing outcome in a drone ship.

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

- The query to rank the count of landing outcomes between the dates 2010-06-04 and 2017-03-20, in descending order is:

%sql select Landing_Outcome, count(*) as Count from SPACEXTABLE group by Landing_Outcome having Date between ‘20100604’ and ‘20170320’ order by Count desc

- When using the expression **group by**, one must use the expression **having** to select specific rows.
- We also declare the **count()** function as ‘Count’, so that we can then order the result by this column.

```
%sql select Landing_Outcome, count(*) as Count from SPACEXTABLE group by Landing_Outcome \
having Date between '20100604' and '20170320' order by Count desc
```

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

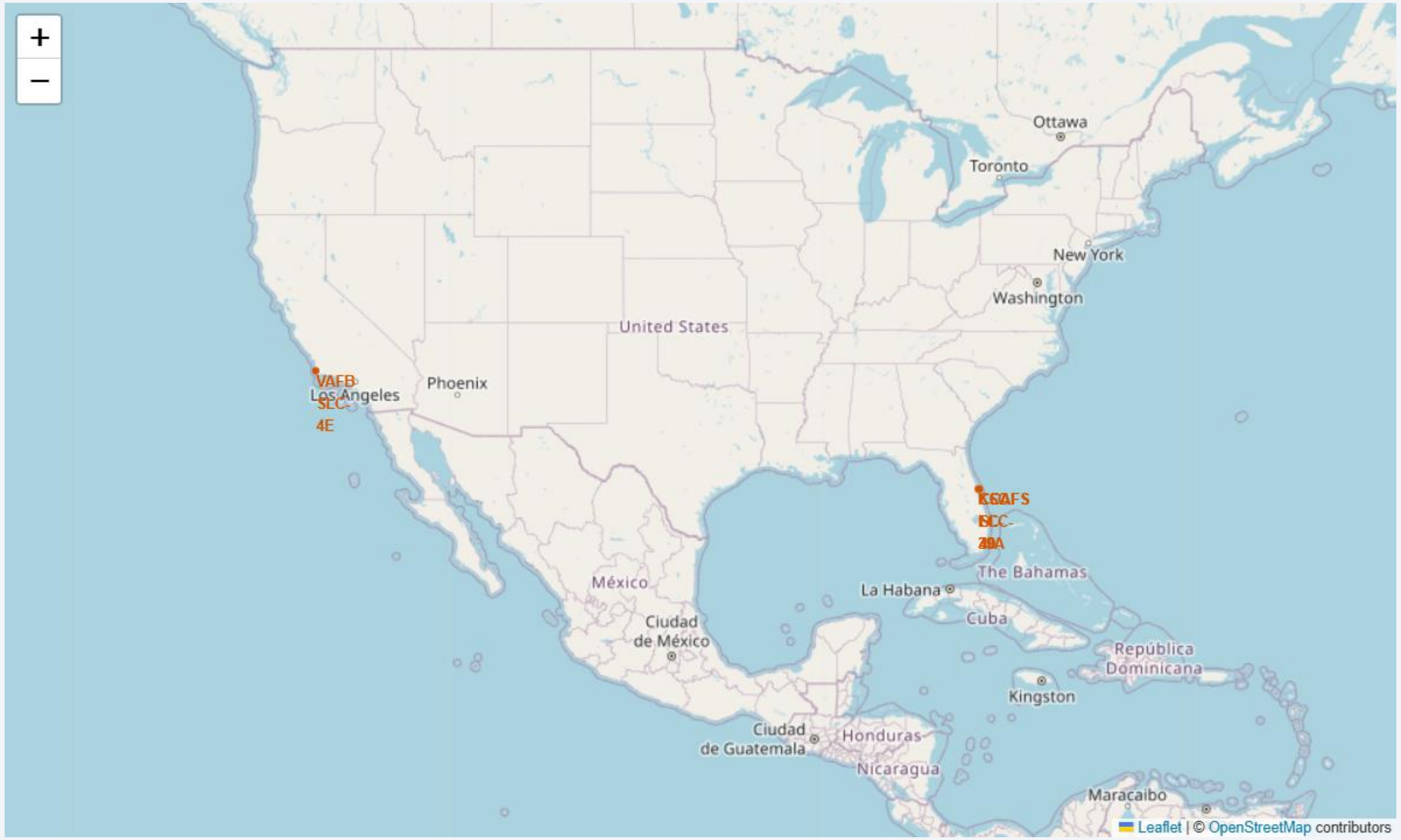
Landing_Outcome	Count
No attempt	21
Success (drone ship)	14
Success (ground pad)	9
Failure (drone ship)	5
Controlled (ocean)	5
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

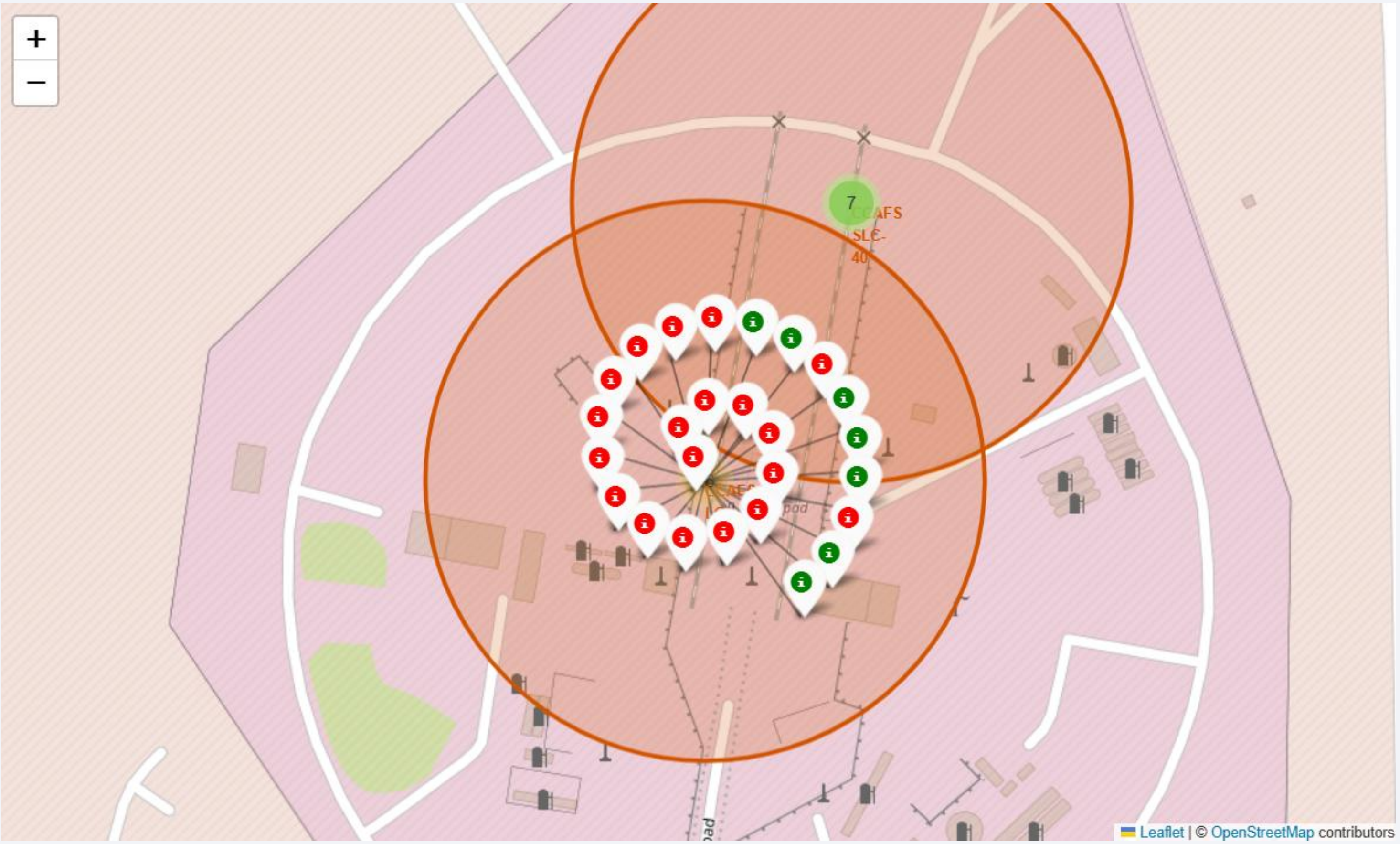
Launch Site locations on a world map using Folium

- The image shows the generated Folium map which includes all launch sites' location markers on a global map.
- We can see that there is one launch site on the west coast of the US, while there are 3 launch sites on the east coast of the US.



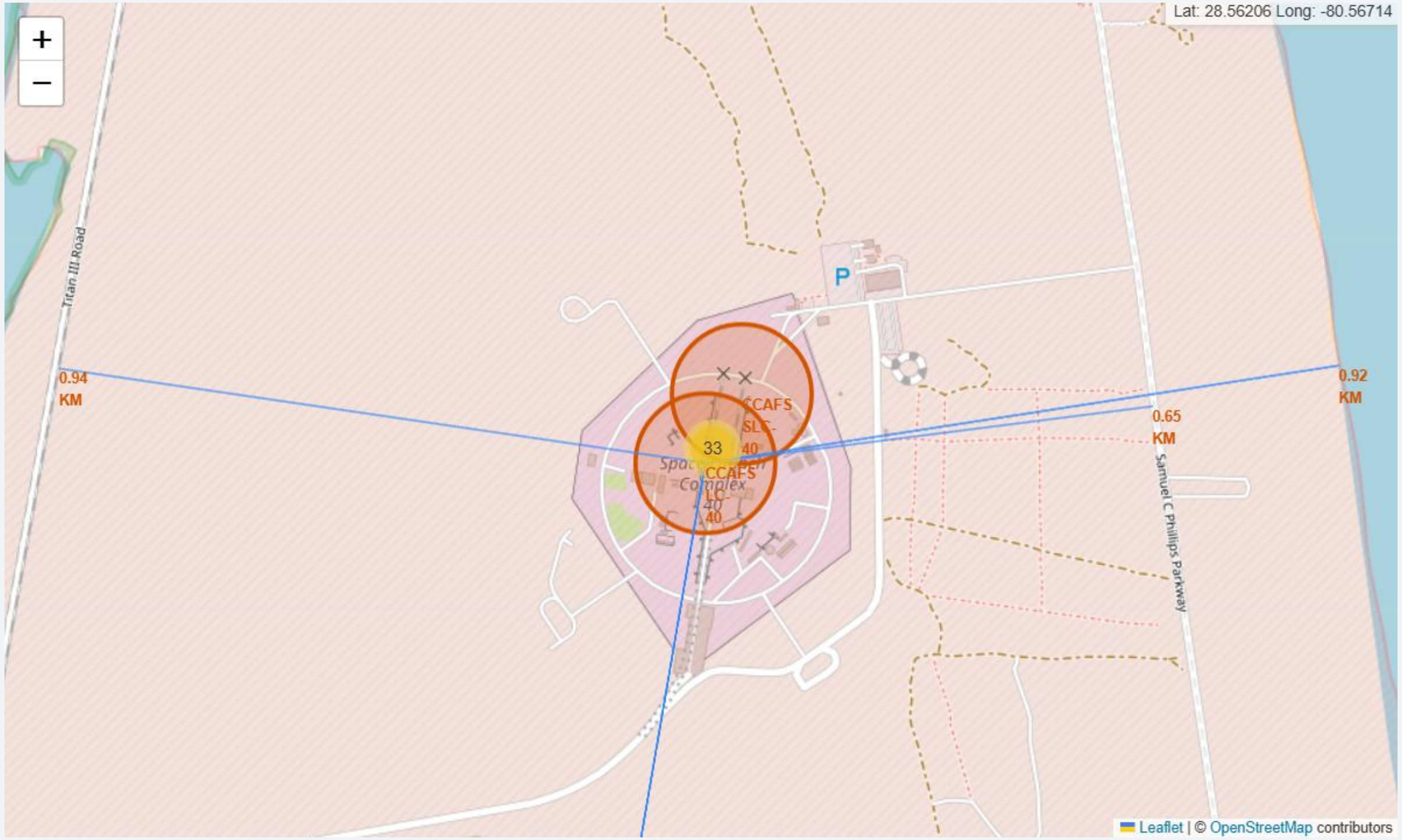
Successful landings for a Launch Site using Folium

- The image shows the folium map with the color-labeled launch outcomes for a specific launch site.
- We can see that this Launch Site in particular (CCAFS LC-40) has a low success rate.



Distances to proximities using Folium

- The image shows the generated folium map with the selected launch site and lines drawn to its proximities such as railway, highway, coastline, with distance calculated and displayed.
- We can see that the distance to the coastline, railway, and highway is short (< 1 km), while the distance to the closest city is larger (out of frame).



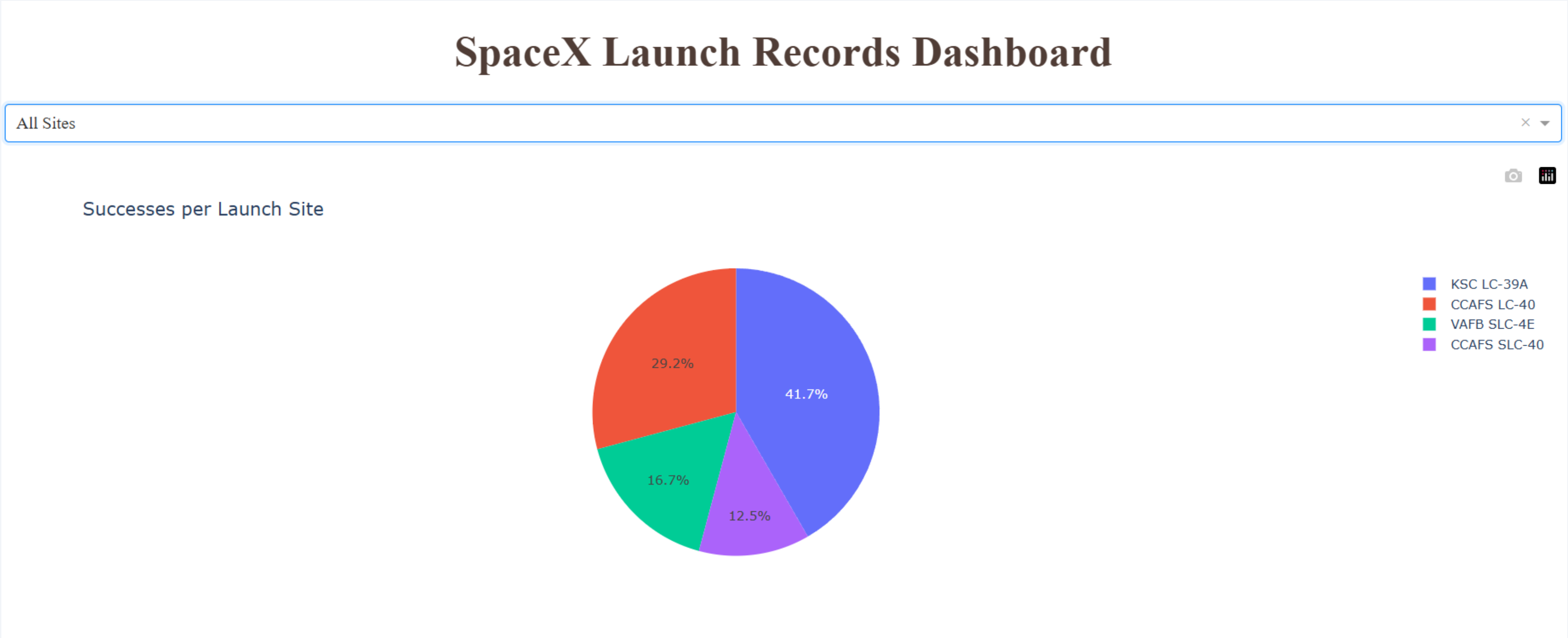
Section 4

Build a Dashboard with Plotly Dash

Successes per Launch Site using Dashboard

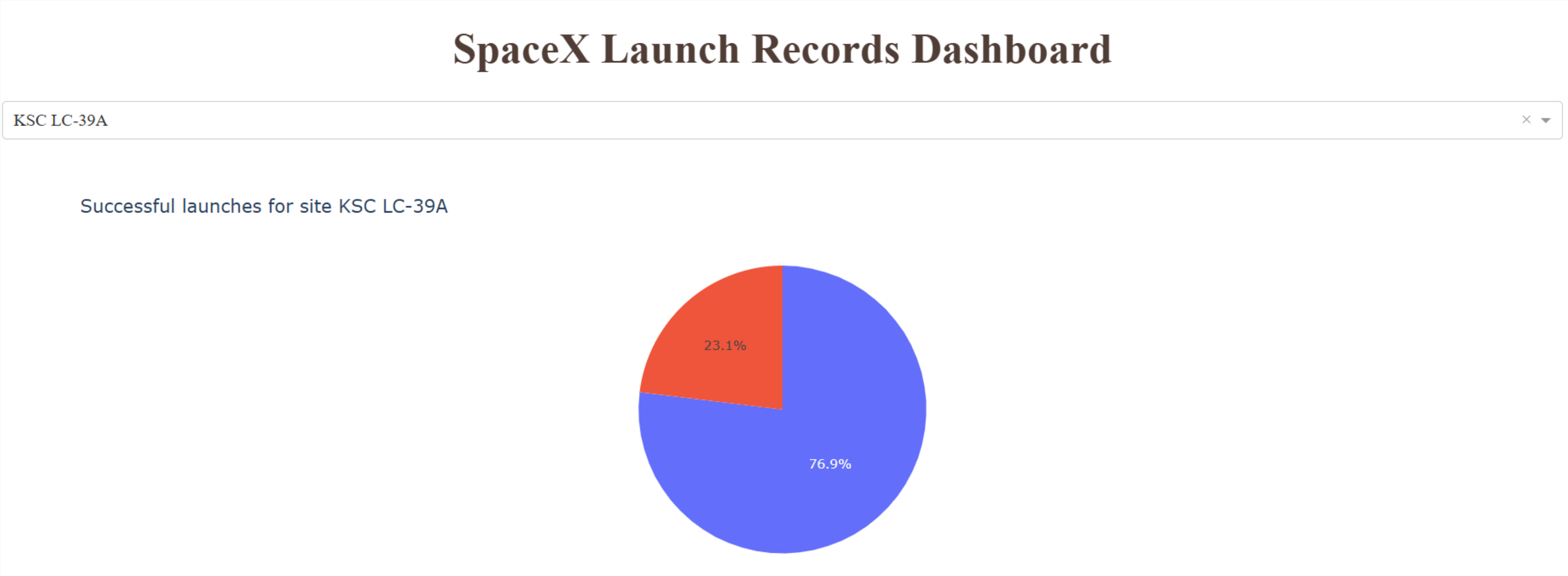
- This screenshot of the dashboard shows the launch success count for all sites, in the form of a pie chart.

- We can see that most of the successes happened in the KSC LC-39A site, with 41.7%, followed by CCAFS LC-40 with 29.2%.



Launch Site with the highest success ratio using Dashboard

- The screenshot of the Dashboard shows the pie chart for the launch site with highest launch success ratio.
- We have used a dropdown menu to select the different launch sites, and found that site KSC LC-39A has the highest success ratio, with 76.9%.



Payload Mass vs Launch Outcome using Dashboard

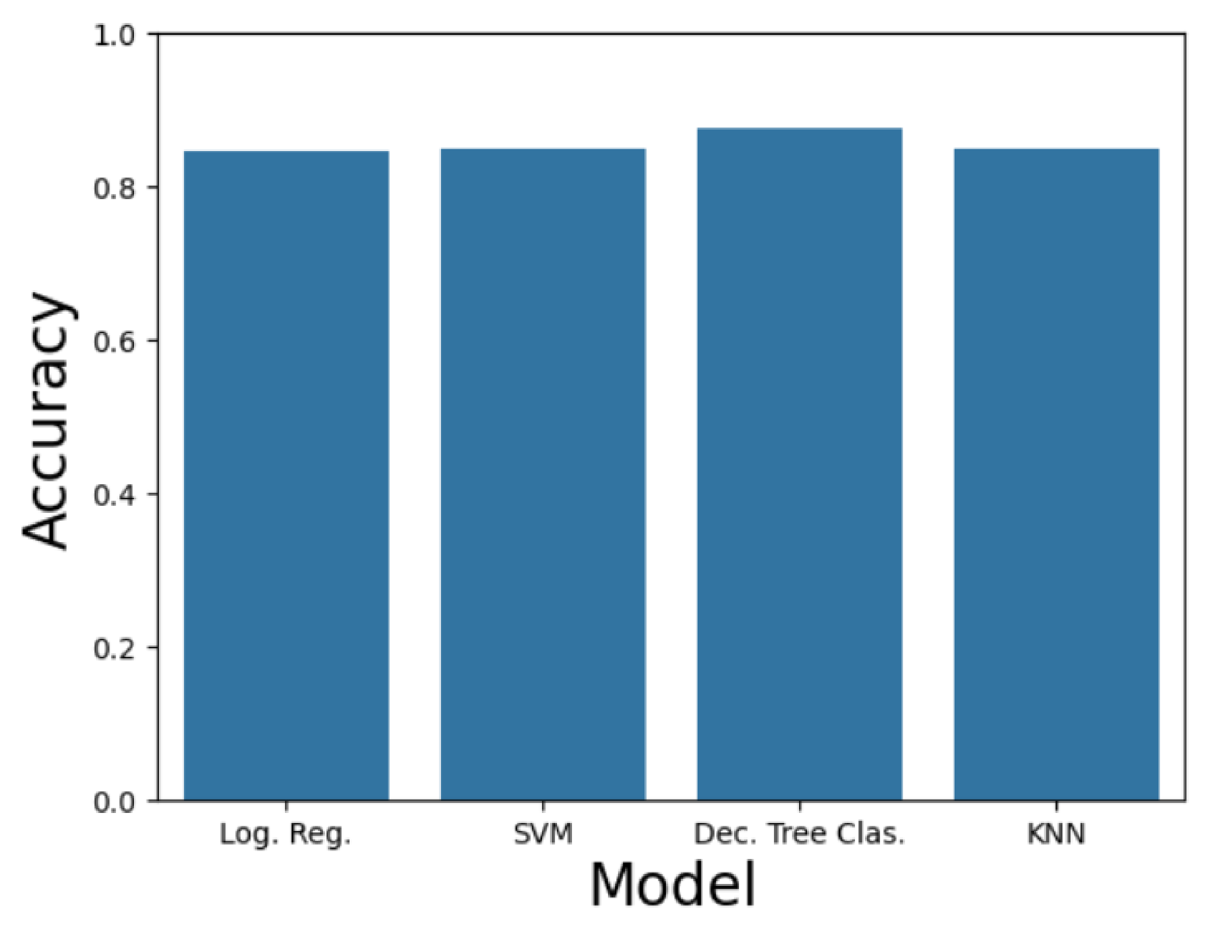
- The screenshot of the Dashboard shows a Payload Mass vs. Launch Outcome scatter plot for all sites, with a specific Payload Mass range (2000 - 6000 kgs) selected in the range slider.
- We can see that in this Payload Mass range, Booster Version FT has a high success rate, while Booster Version v1.1 has a low success rate.

Section 5

Predictive Analysis (Classification)

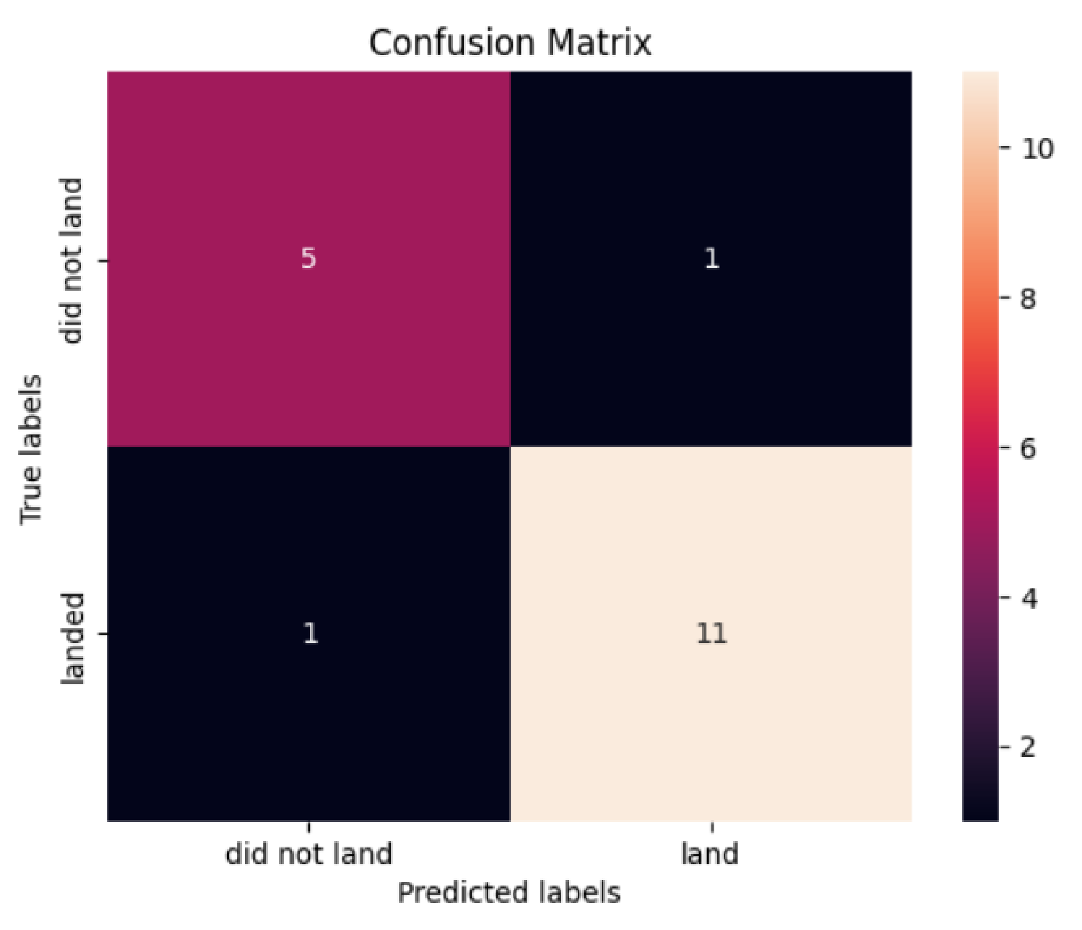
Classification Accuracy

- The bar chart shows the model accuracy for all built classification models.
- We can see that while most models have a similar accuracy, the Decision Tree Classifier has a slightly higher accuracy than the other models.



Confusion Matrix

- Here is the confusion matrix of the best performing model, which is the Decision Tree Classifier.
- The other models had problems with false positives, while correctly predicting all failed landings (0 false negatives). In contrast, the Decision Tree Classifier trades that perfect failed landing prediction for a much more accurate prediction for successful landings, significantly reducing the false positives, and achieving a better overall performance.



Conclusions

- Launch and landing records for SpaceX missions were gathered using the SpaceX API and Wikipedia web scraping.
- The data was preprocessed to classify each mission as having either a successful booster landing, or a failed booster landing.
- An EDA was performed using tools like SQL and Matplotlib/Seaborn.
- Using the scikit-learn library, various classification models were built, trained, and evaluated based on the collected data
- Of these models, the Decision Tree Classifier was found to have the best performance, with a score of 88.9%.

Thank you!