IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Konstantinos
Stamatakis
02.04.2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive analysis (Classification)
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

- Project background and context

We predicted if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- Problems you want to find answers

- What influences if the rocket will land successfully?
- The effect each relationship with certain rocket variables will impact in determining the success rate of a successful landing.
- What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate.

Section 1

Methodology

Methodology

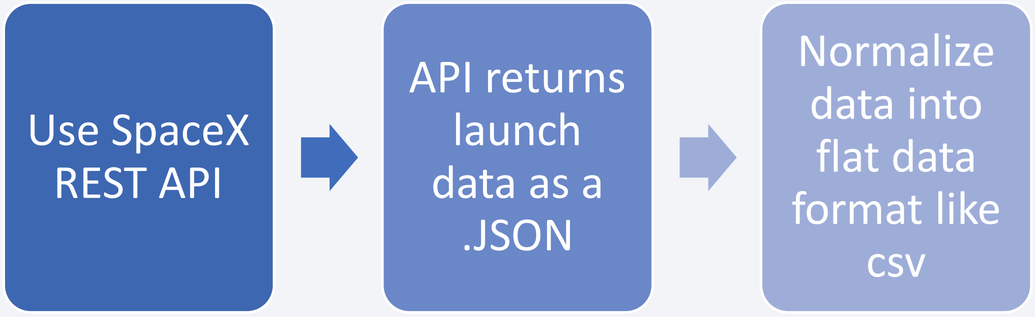
Executive Summary

- Data collection methodology:
  - SpaceX Rest API and Web Scrapping from Wikipedia
- Perform data wrangling
  - One Hot Encoding data fields for Machine Learning and removing irrelevant columns
- 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

Data Collection

- The following datasets was collected by
    - With the SpaceX REST API we obtained SpaceX launch data.
    - This API will provide us with data about the launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.
    - The goal is to use all the information to build a predictive model on whether SpaceX will attempt to land a rocket or not.
    - The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
    - Another data source for gathering Falcon 9 Launch data is web scraping Wikipedia with BeautifulSoup.

SpaceX API

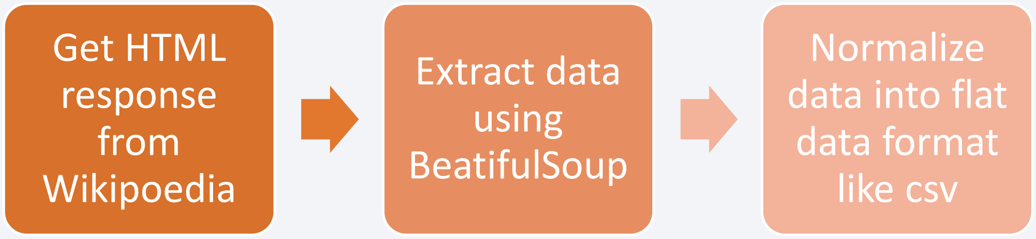


Use SpaceX REST API

API returns launch data as a .JSON

Normalize data into flat data format like csv

Web Scrapping



Get HTML response from Wikipoedia

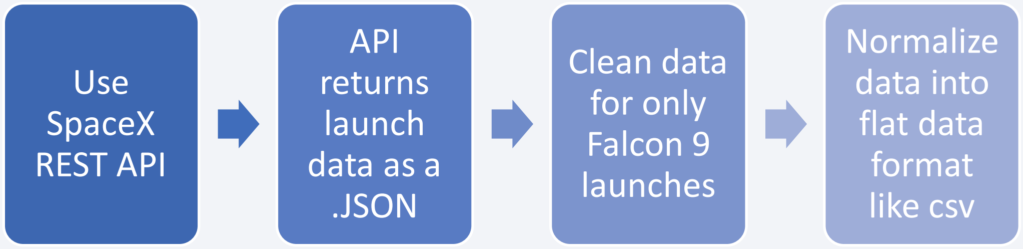
Extract data using BeatifulSoup

Normalize data into flat data format like csv

Data Collection – SpaceX API

1. Getting a response from SpaceX API
2. Convert response into an easy editable JSON file
3. Clean data with custom functions
4. Convert the data list to a dictionary and then into a dataframe
5. Filter dataframe and export to a flat file like .csv

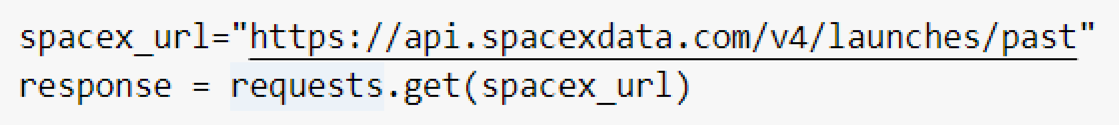
SpaceX API



Use SpaceX REST API → API returns launch data as a .JSON → Clean data for only Falcon 9 launches → Normalize data into flat data format like csv

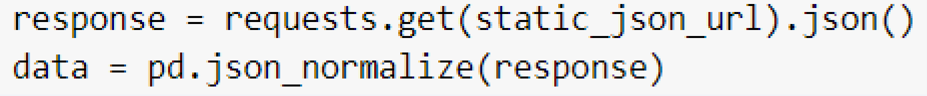
[Github link]

1.



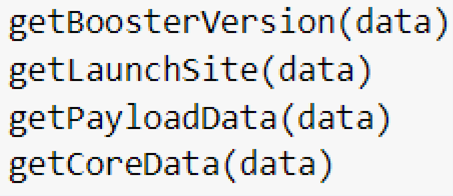
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2.



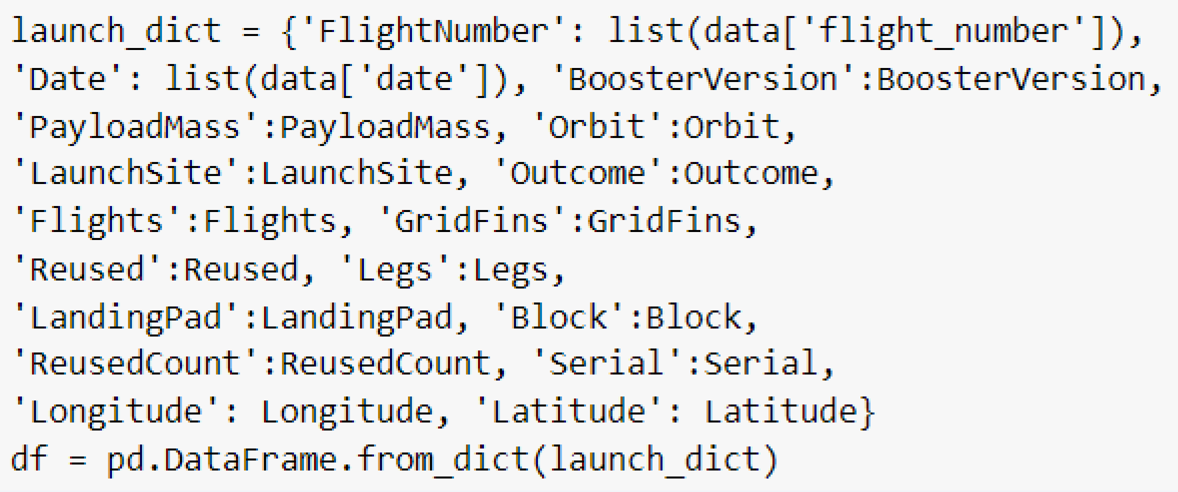
```
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

3.



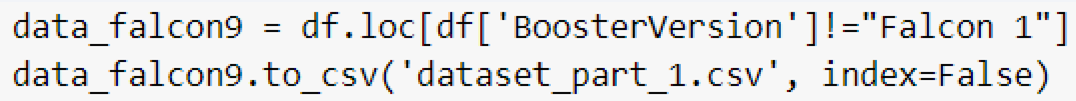
```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

4.



```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']), 'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass, 'Orbit':Orbit,
'LaunchSite':LaunchSite, 'Outcome':Outcome,
'Flights':Flights, 'GridFins':GridFins,
'Reused':Reused, 'Legs':Legs,
'LandingPad':LandingPad, 'Block':Block,
'ReusedCount':ReusedCount, 'Serial':Serial,
'Longitude': Longitude, 'Latitude': Latitude}
df = pd.DataFrame.from_dict(launch_dict)
```

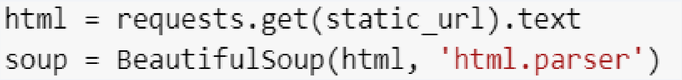
5.



```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

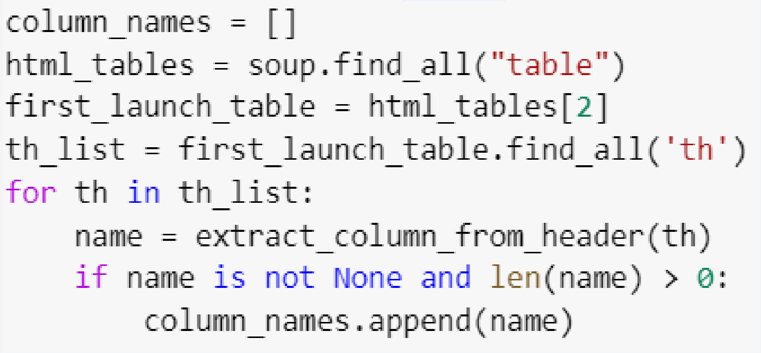
1.



```
html = requests.get(static_url).text
soup = BeautifulSoup(html, 'html.parser')
```

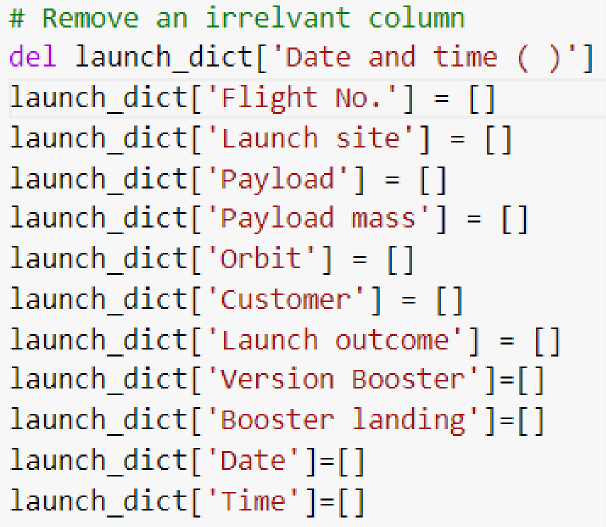
static url

2.



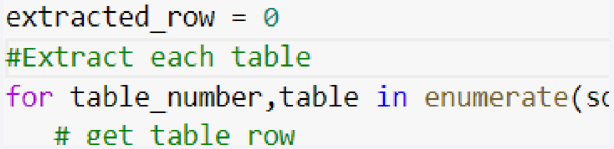
```
column_names = []
html_tables = soup.find_all("table")
first_launch_table = html_tables[2]
th_list = first_launch_table.find_all('th')
for th in th_list:
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

3.



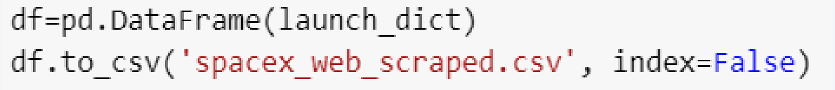
```
# Remove an irrelvant column
del launch_dict['Date and time ( )']
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

4. Whole code too long



```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(sc
   # get table row
```

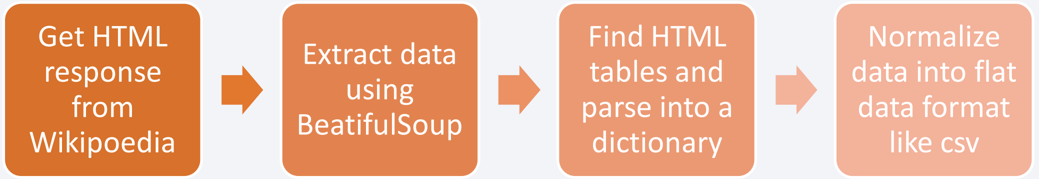
5.



```
df=pd.DataFrame(launch_dict)
df.to_csv('spacex_web_scraped.csv', index=False)
```

1. Getting response from HTML and creating a Beautiful soup object
2. Finding tables and get column names
3. Create a dictionary
4. Append data to keys
5. Convert filled dictionary into a dataframe and xport into a flat file format like .csv

Web Scrapping



Get HTML response from Wikipoedia → Extract data using BeatifulSoup → Find HTML tables and parse into a dictionary → Normalize data into flat data format like csv

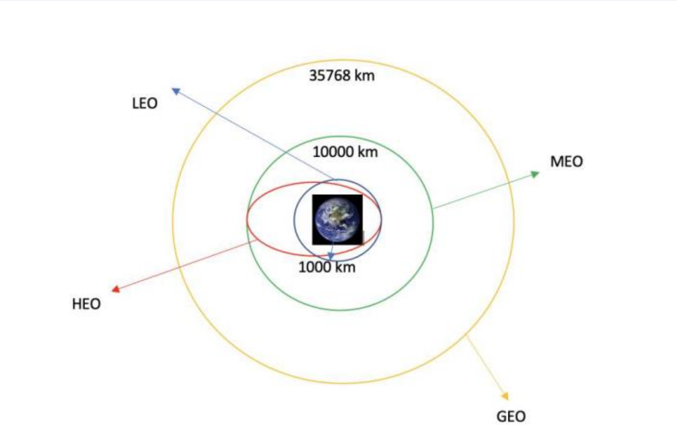
Github link

Data Wrangling

Github link

Introduction:

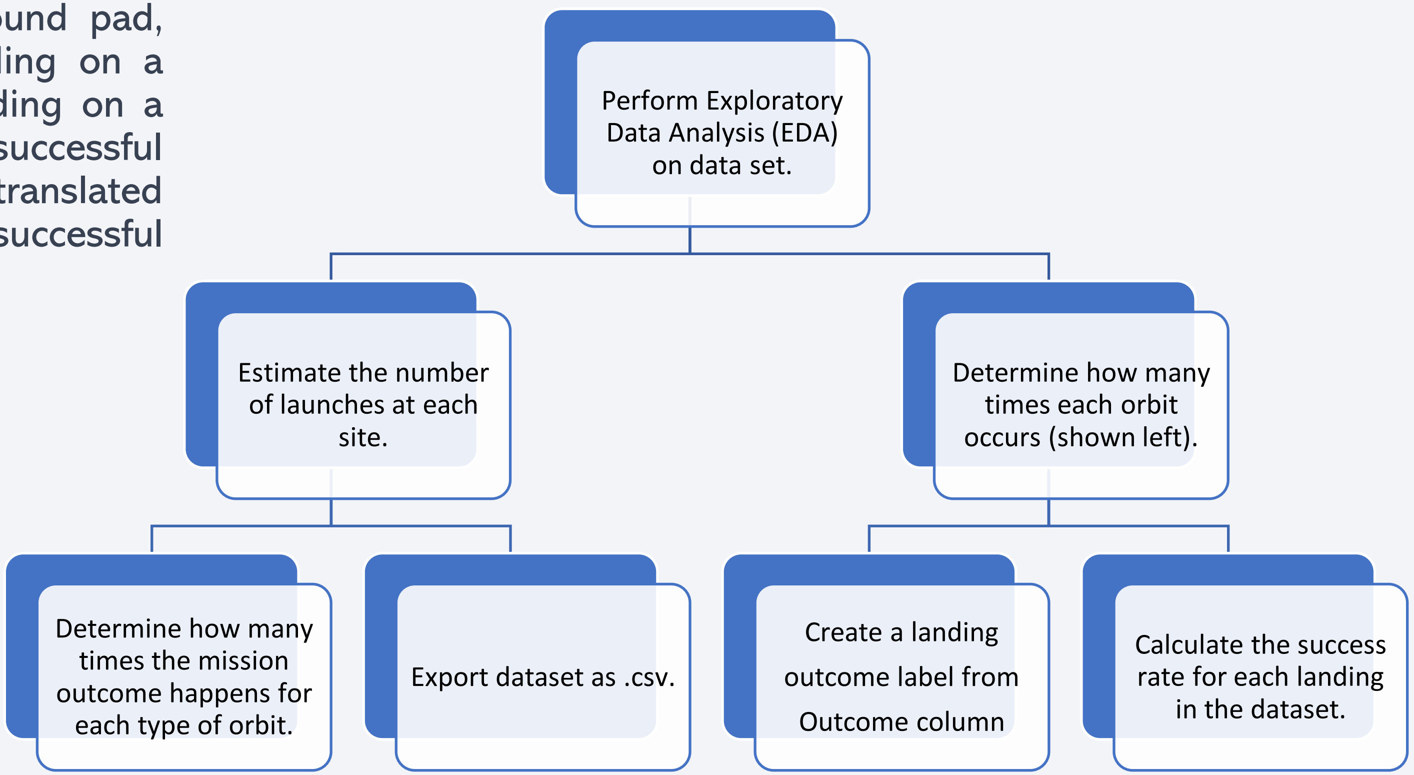
The dataset contains various instances where the booster failed to land successfully. Some attempts at landing were unsuccessful due to accidents. For instance, a mission outcome marked as "True Ocean" indicates a successful landing in a specific region of the ocean, while "False Ocean" indicates an unsuccessful landing in a specific region of the ocean. Similarly, "True RTLS" signifies a successful landing on a ground pad, whereas "False RTLS" denotes an unsuccessful landing on a ground pad. "True ASDS" indicates a successful landing on a drone ship, while "False ASDS" indicates an unsuccessful landing on a drone ship. These outcomes are mainly translated into training labels, with a value of 1 indicating a successful landing and 0 indicating an unsuccessful one.



A visual representation of the typical types of orbits that SpaceX employs

Add the GitHub URL of your completed data wrangling related notebooks, as an external reference and peer-review purpose

Data Wrangling Process:



Perform Exploratory Data Analysis (EDA) on data set.

Estimate the number of launches at each site.

Determine how many times each orbit occurs (shown left).

Determine how many times the mission outcome happens for each type of orbit.

Export dataset as .csv.

Create a landing outcome label from Outcome column

Calculate the success rate for each landing in the dataset.

EDA with Data Visualization

Github link

Scatter Graphs:

Flight Number VS. Payload Mass

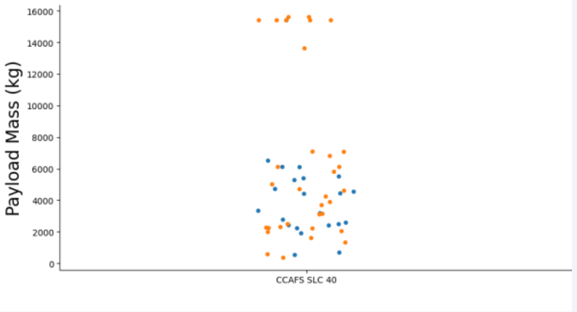
Flight Number VS. Launch Site

Payload VS. Launch Site

Orbit VS. Flight Number

Payload VS. Orbit Type

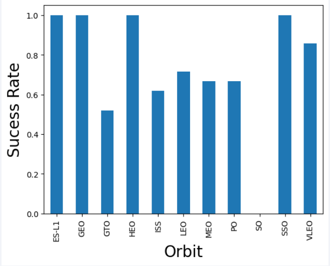
Orbit VS. Payload Mass



A scatter plot is a graphical representation that displays how one variable is influenced by another variable. The degree to which the two variables are related is referred to as correlation. Typically, scatter plots present a substantial amount of data.

Bar Graph:

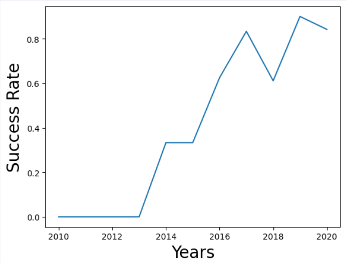
Mean VS Orbit



A bar chart is a simple visual tool that enables quick and easy comparison of data sets among various groups. It presents categories on one axis and a numerical value on the other, with the aim of illustrating the relationship between the two axes. Additionally, bar charts can effectively demonstrate significant changes in data over time.

Line Graph:

Succes Rate VS. Year



Line graphs are a valuable tool as they effectively illustrate data variables and trends, allowing for easy interpretation and analysis. They can also aid in predicting the outcomes of data that has not yet been recorded.

- Add the GitHub URL of your completed EDA with data visualizatio

EDA with SQL

Performed SQL queries to gather information about the dataset:

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC'
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster_versions which have carried the maximum payload mass.
- Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Ranking the count of successful landing_outcomes between the date 2010 06 04 and 2017 03 20 in descending order.

Github link

Build an Interactive Map with Folium

Github link

- We created **an interactive map** by using **the Launch Data**. This was done by taking the Latitude and Longitude Coordinates of each launch site and marking them with a Circle Marker on the map. The name of the launch site was also added as a label within the Circle Marker.
- The dataframe launch_outcomes, which includes both **failures and successes**, was categorized into two classes: **0 and 1**. Each class was then assigned a specific marker color on the map. Failures were given a Red marker and successes were given a Green marker. The markers were also grouped into clusters using the **MarkerCluster()** function.
- We utilized **Haversine's formula** to determine the **distance between the Launch Site and different landmarks**. By doing this, we were able to identify trends and patterns regarding what is present around the Launch Site. To visualize this, lines were drawn on the map to represent the distances to the landmarks.

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

BUILDING MODEL

- Load the dataset into NumPy and Pandas.
- Transform Data.
- Split our data into training and test data sets.
- Check how big the test samples are.
- Decide which type of machine learning algorithms to use.
- Set the parameters and algorithms to GridSearchCV.
- Fit our datasets into the GridSearchCVobjects and train our dataset.

EVALUATING MODEL

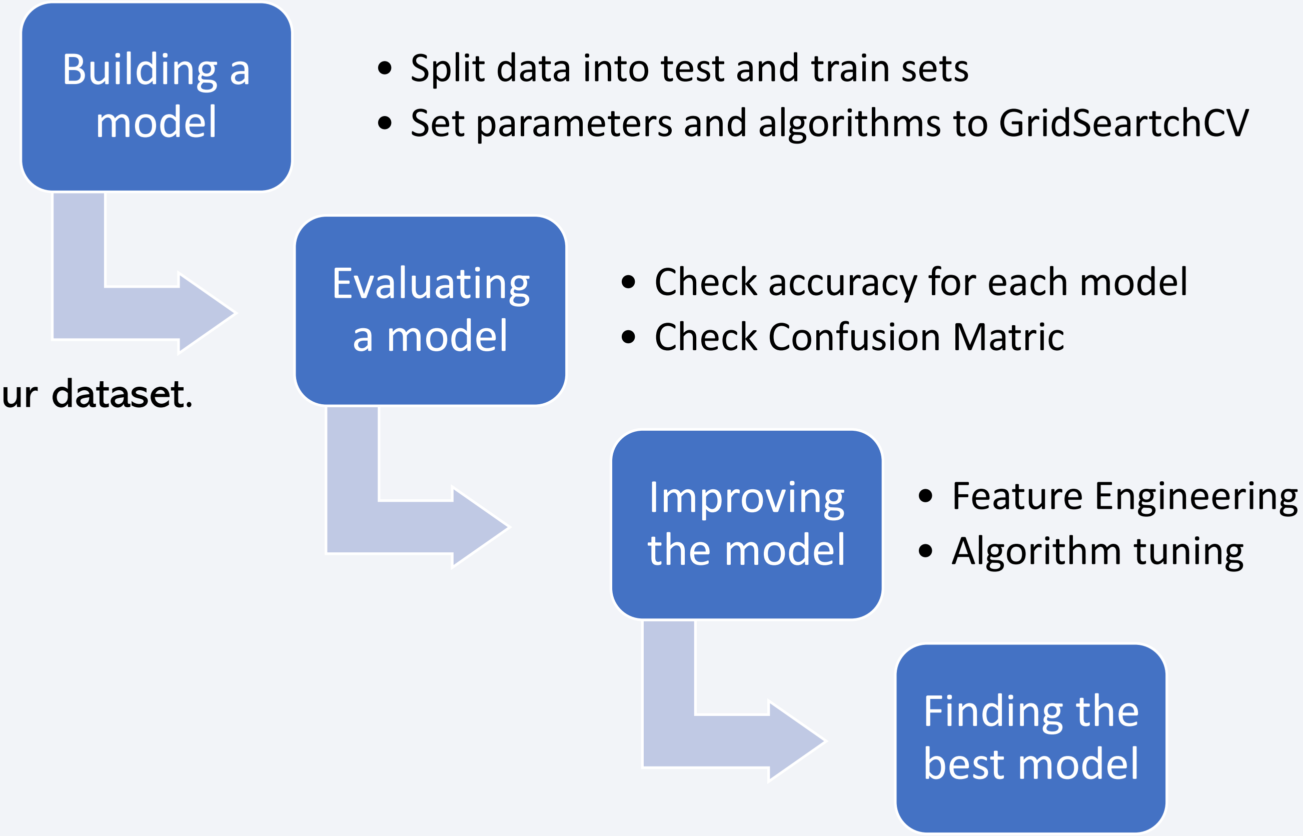
- Check accuracy for each model.
- Get tuned hyperparameters for each type of algorithms.
- Plot the Confusion Matrix.

IMPROVING MODEL

- Feature Engineering.
- Algorithm Tuning.

FINDING THE BEST PERFORMING CLASSIFICATION MODEL

- The model with the best accuracy score wins as the best performing model.
- At the end of the notebook there is a dictionary of algorithms with scores.



Building a model

- Split data into test and train sets
- Set parameters and algorithms to GridSeartchCV

Evaluating a model

- Check accuracy for each model
- Check Confusion Matric

Improving the model

- Feature Engineering
- Algorithm tuning

Finding the best model

[Github link]()

Results

- Exploratory data analysis results

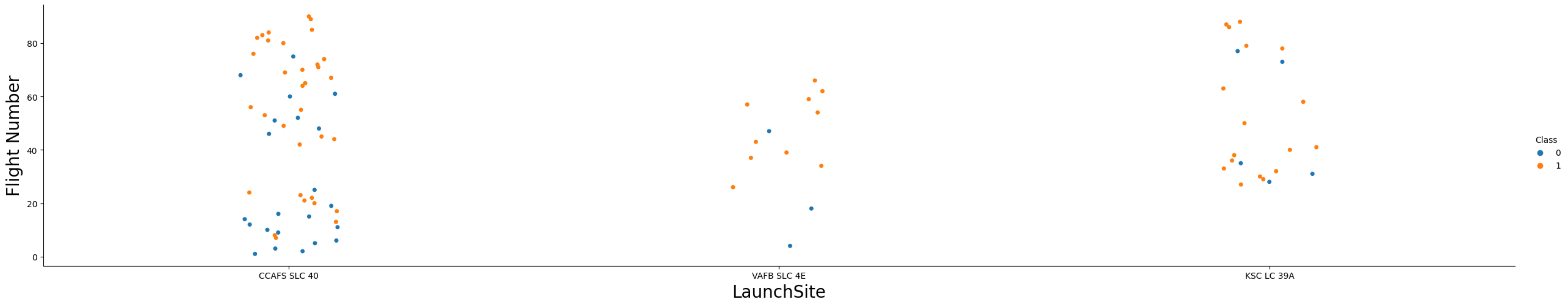
- Interactive analytics demo in screenshots

- Predictive analysis results

Section 2

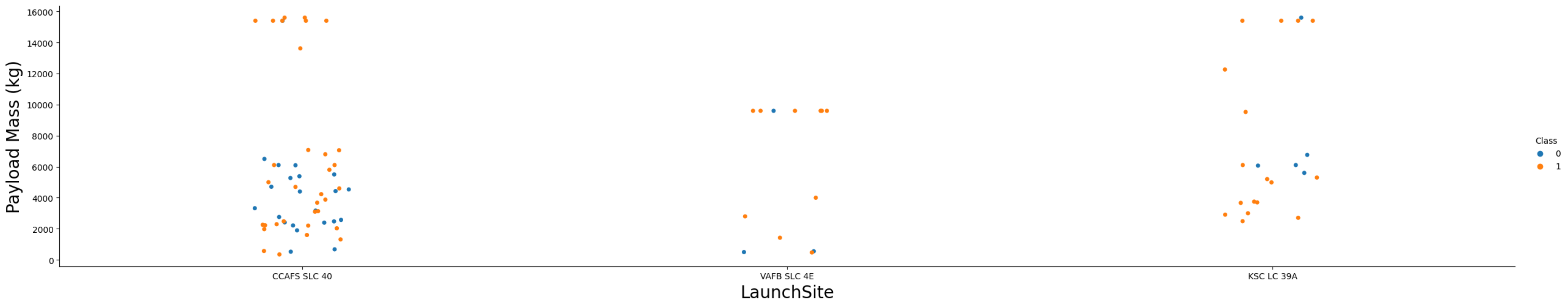
Insights drawn from EDA

Flight Number vs. Launch Site



The greater the number of flights from a launch site, the higher the probability of success at that site.

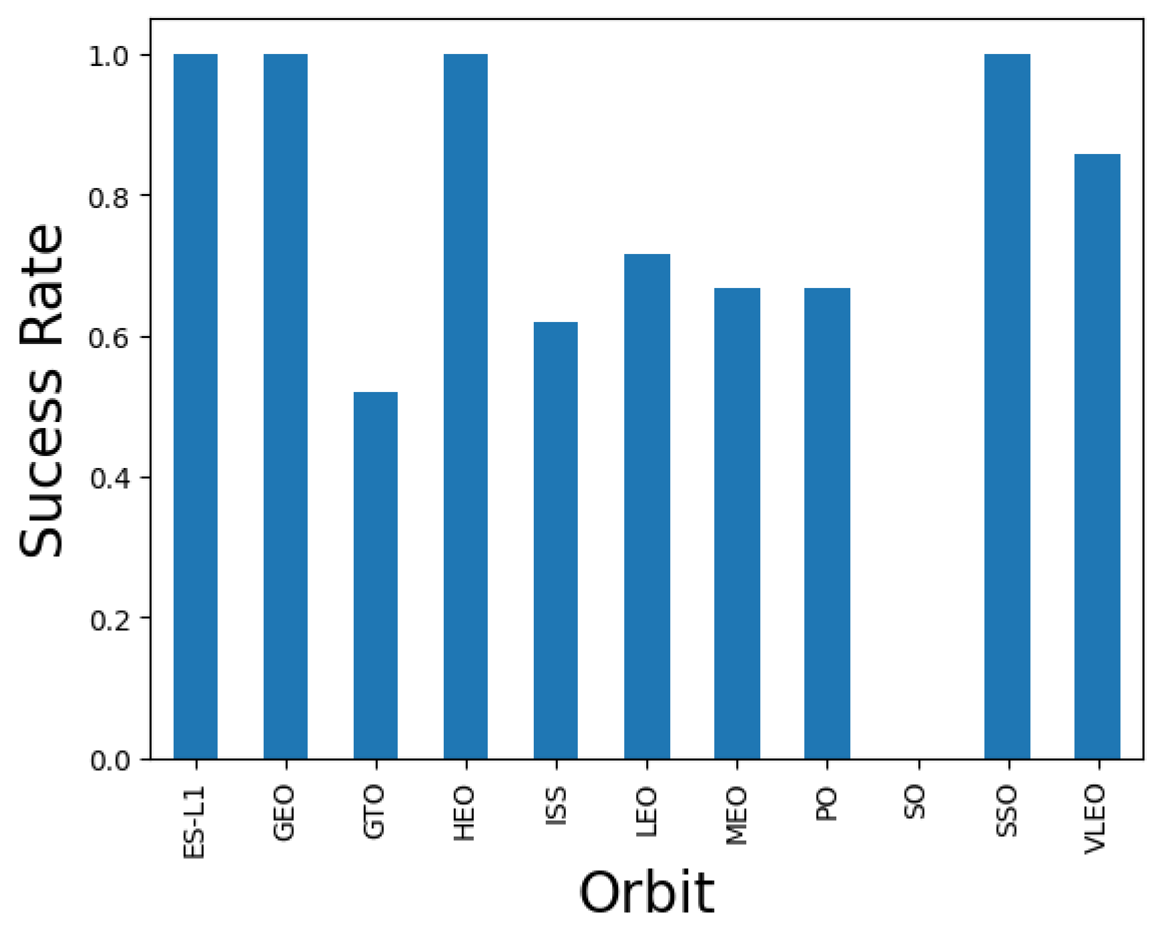
Payload vs. Launch Site



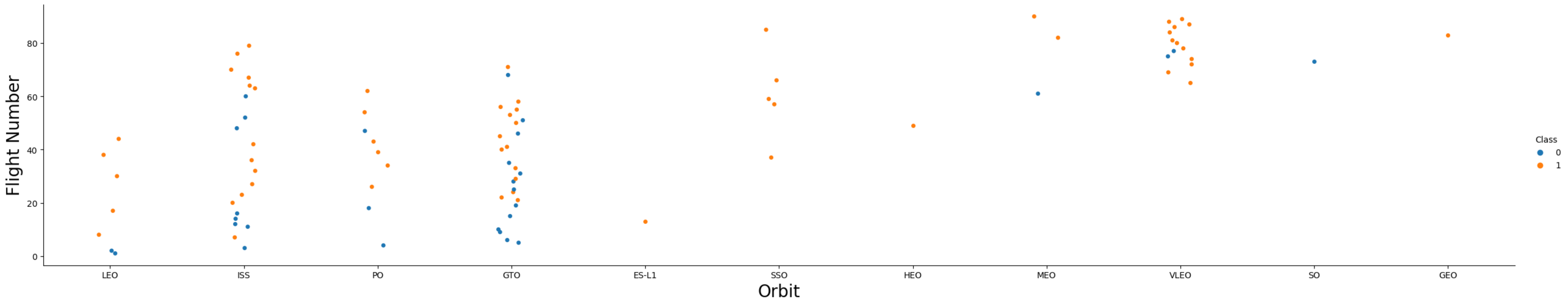
f the payload mass is increased for Launch Site CCAFS SLC 40, there is a corresponding increase in the success rate of the rocket. However, based on the visualization, it is difficult to determine if there is a clear dependency between the launch site and the payload mass for achieving a successful launch.

Success Rate vs. Orbit Type

Orbit ES-L1, GEO, HEO, SSO has the best Success Rate

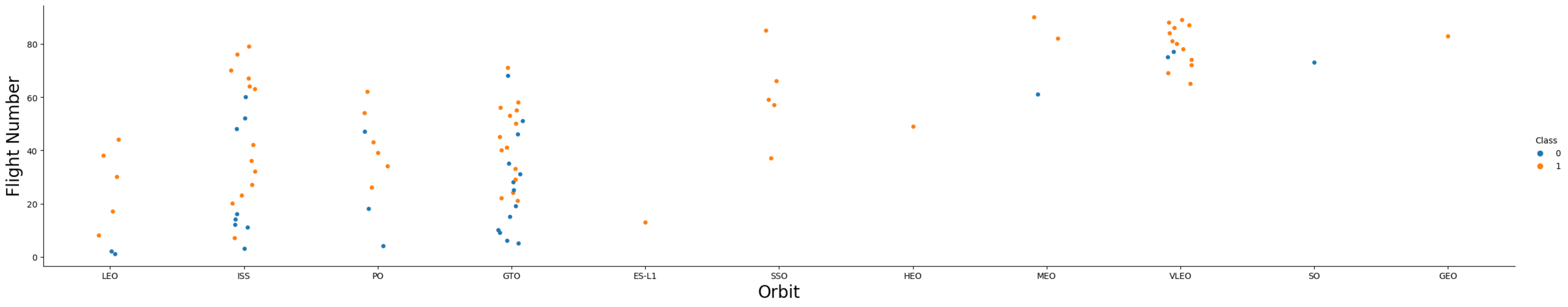


Flight Number vs. Orbit Type



When observing the Low Earth Orbit (LEO), there appears to be a correlation between the number of flights and the rate of success. However, in the case of the Geostationary Transfer Orbit (GTO), there seems to be no discernible relationship between the number of flights and the success rate.

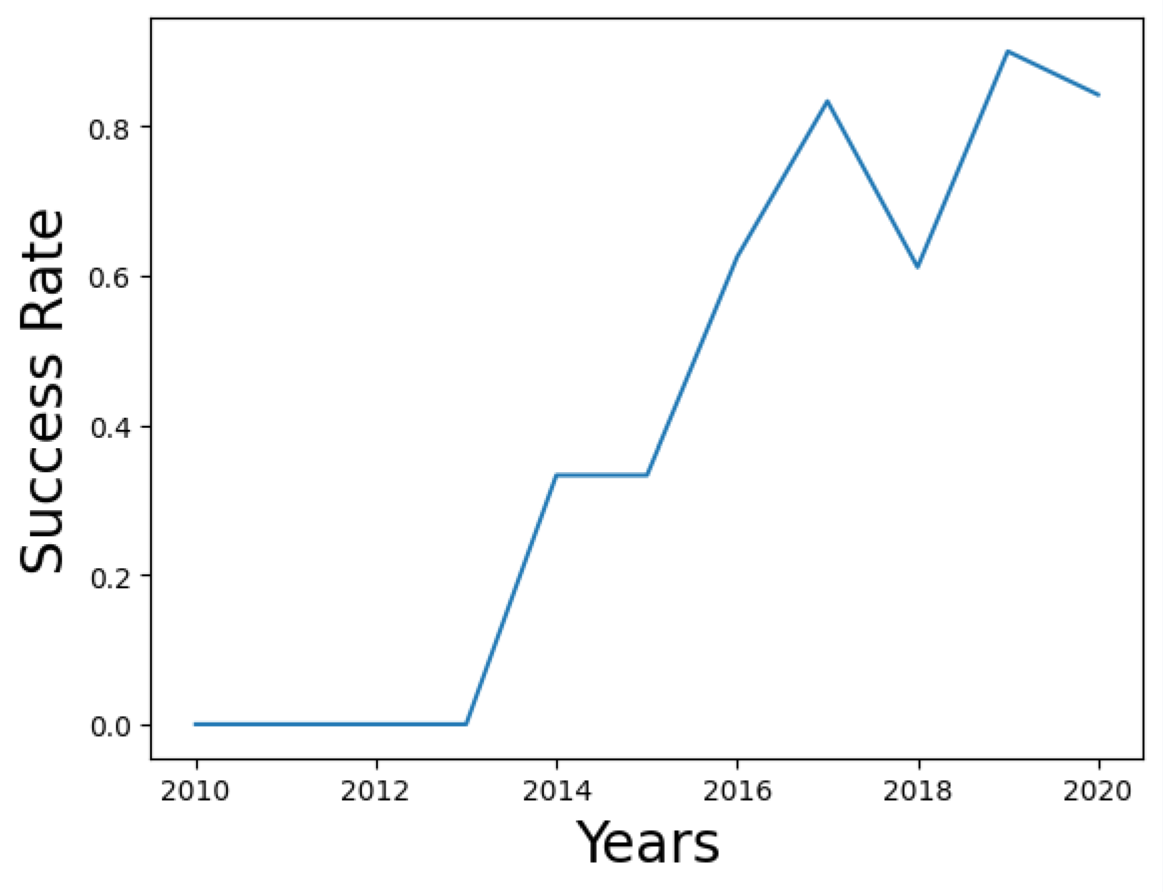
Payload vs. Orbit Type



It can be observed that heavy payloads have a negative impact on the success rate of rockets in GTO orbits. On the other hand, heavy payloads have a positive effect on the success rate of rockets in GTO and Polar LEO (International Space Station) orbits.

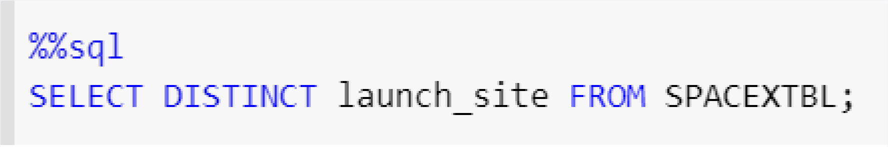
Launch Success Yearly Trend

It is noticeable that the success rate has been on an upward trend from 2013 to 2020.



All Launch Site Names

SQL Query code:

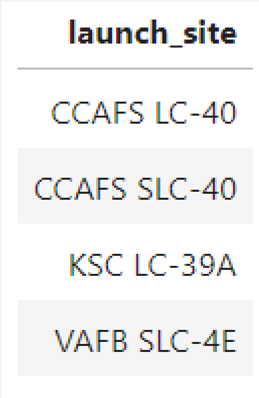


```
%%sql
SELECT DISTINCT launch_site FROM SPACEXTBL;
```

QUERY EXPLAINATION

Using the word ***DISTINCT*** in the query means that it will only show Unique values in the ***Launch_Site*** column from ***SpaceXTBL***

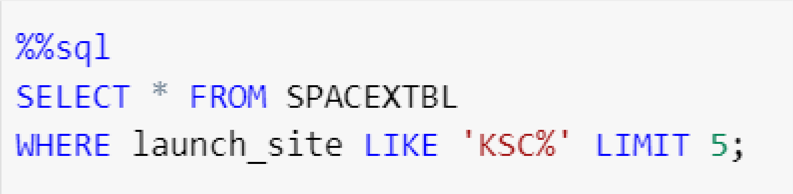
SQL Query result:



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

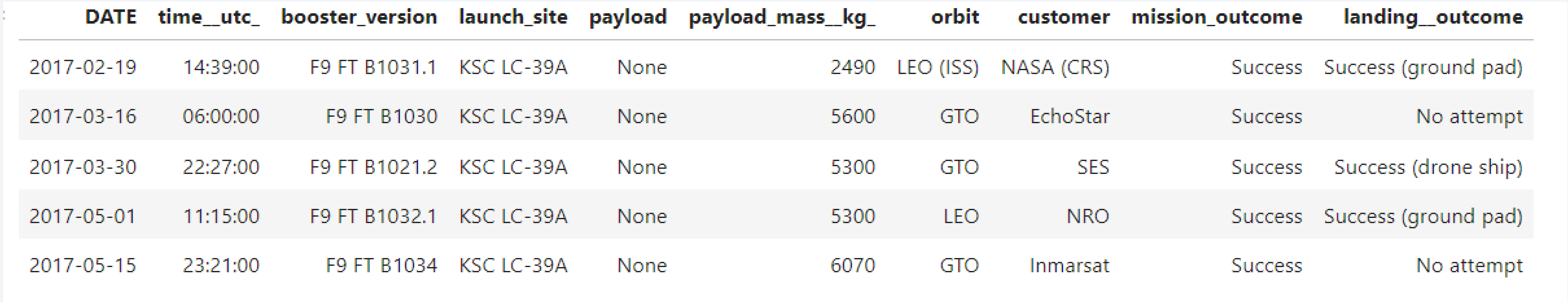
Launch Site Names Begin with 'KSC'

SQL Query code:



```
%%sql
SELECT * FROM SPACEXTBL
WHERE launch_site LIKE 'KSC%' LIMIT 5;
```

SQL Query result:



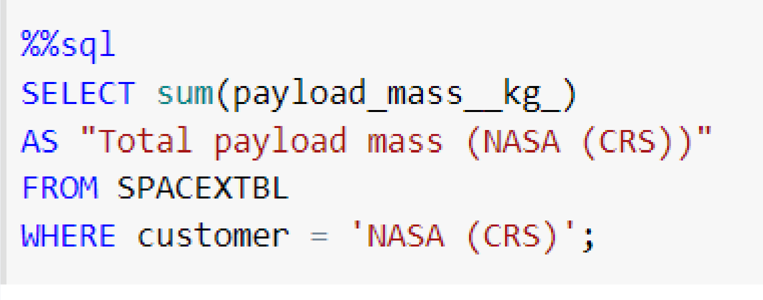
DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	None	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	06:00:00	F9 FT B1030	KSC LC-39A	None	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	None	5300	GTO	SES	Success	Success (drone ship)
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	None	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	None	6070	GTO	Inmarsat	Success	No attempt

QUERY EXPLAINATION

This query will retrieve a limited set of data from **SpaceXTBL** by using the phrase “**limit 5**" to indicate that only five records should be displayed. Additionally, the “**like**" keyword is used with a wildcard to search for Launch_Site names that start with "**KSC**". The percentage sign at the end of the phrase indicates that any characters can follow "**KSC**".

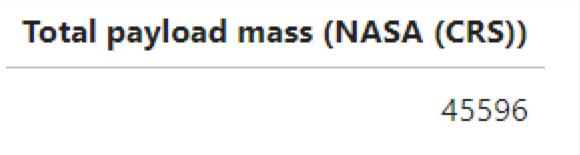
Total Payload Mass

SQL Query code:



```
%%sql
SELECT sum(payload_mass__kg_)
AS "Total payload mass (NASA (CRS))"
FROM SPACEXTBL
WHERE customer = 'NASA (CRS)';
```

SQL Query result:



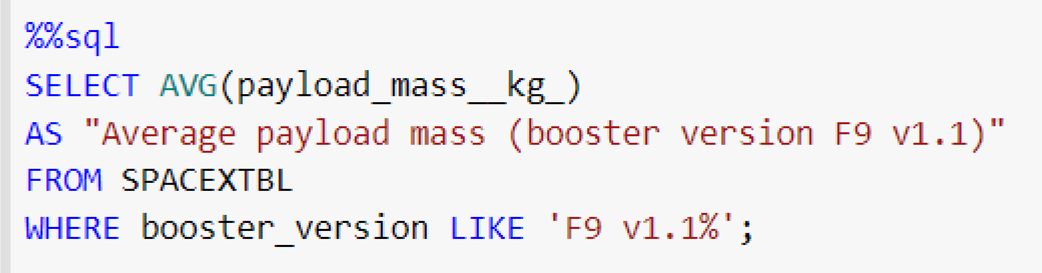
Total payload mass (NASA (CRS))
45596

QUERY EXPLAINATION

In this statement, the “**sum**" function is utilized to calculate the total of the values in the "Payload_Mass_kg_" column. The “**where**" clause is then employed to limit the dataset to only those records where the "**Customer**" field is "**NASA (CRS)**". This ensures that the calculations performed by the “**sum**" function only apply to the relevant data.

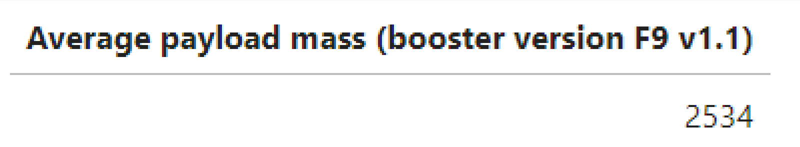
Average Payload Mass by F9 v1.1

SQL Query code:



```
%%sql
SELECT AVG(payload_mass__kg_)
AS "Average payload mass (booster version F9 v1.1)"
FROM SPACEXTBL
WHERE booster_version LIKE 'F9 v1.1%';
```

SQL Query result:



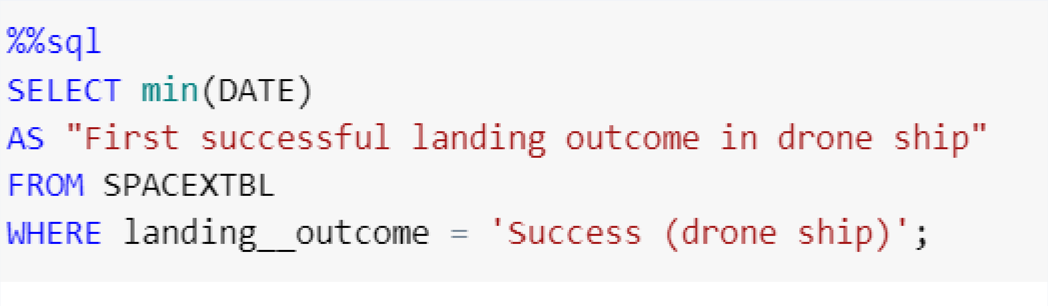
Average payload mass (booster version F9 v1.1)
2534

QUERY EXPLAINATION

This statement utilizes the “**avg**" function to compute the average value of the “**Payload_Mass_kg_**" column. The “**where**" clause is then used to filter the data so that the calculation is only performed on records where the "Booster_version" field is "**F9 v1.1**". This ensures that the average calculated by the “**avg**" function is only based on the relevant subset of data.

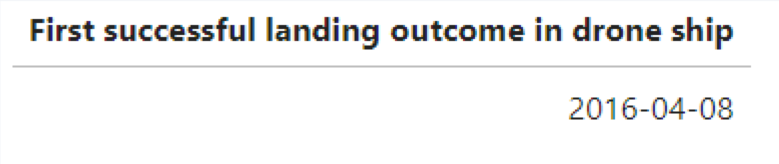
First Successful Ground Landing Date

SQL Query code:



```
%%sql
SELECT min(DATE)
AS "First successful landing outcome in drone ship"
FROM SPACEXTBL
WHERE landing__outcome = 'Success (drone ship)';
```

SQL Query result:



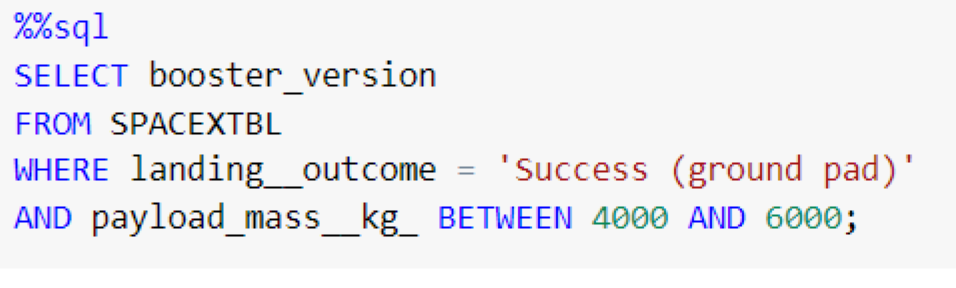
First successful landing outcome in drone ship
2016-04-08

QUERY EXPLAINATION

This statement employs the “**min**" function to determine the earliest date value in the "**Date**" column. The “**where**" clause is then used to limit the dataset to only those records where the "**Landing_Outcome**" field is "**Success (drone ship)**". This ensures that the minimum date calculated by the “**min**" function is based only on the relevant subset of data.

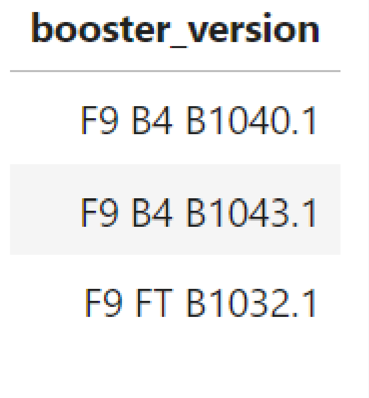
Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query code:



```
%%sql
SELECT booster_version
FROM SPACEXTBL
WHERE landing__outcome = 'Success (ground pad)'
AND payload_mass__kg_ BETWEEN 4000 AND 6000;
```

SQL Query result:



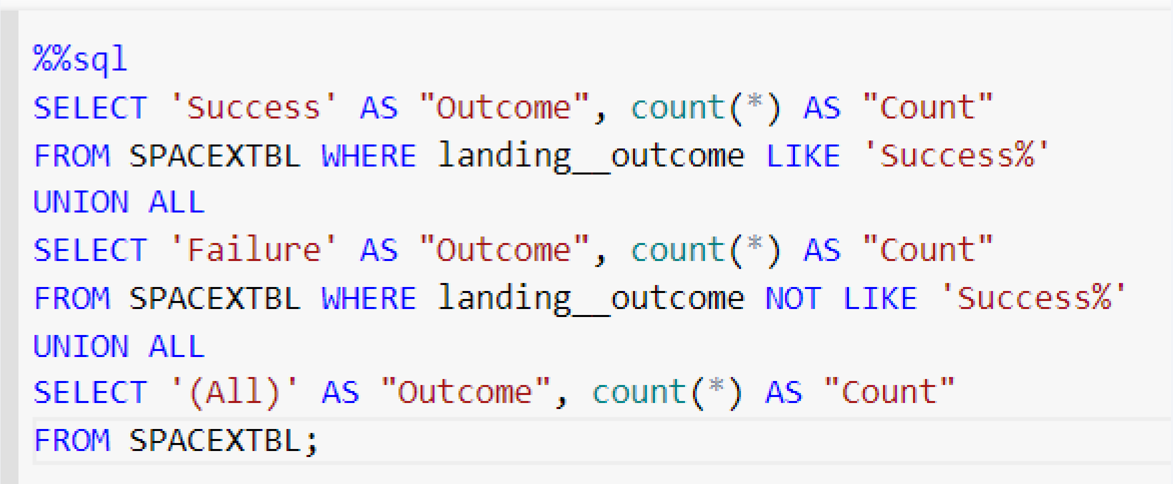
booster_version
F9 B4 B1040.1
F9 B4 B1043.1
F9 FT B1032.1

QUERY EXPLAINATION

In this statement, only the "**Booster_Version**" field is selected from the dataset. The “**where**" clause is then used to filter the data so that only records where the "**Landing_Outcome**" field is "**Success (drone ship)**" are included. Additionally, the “**and**" clause specifies further conditions that the "**Payload_Mass_kg**" field must satisfy - namely, that it must be greater than or equal to 4000 and less than or equal to 6000. This ensures that the returned data meets all of the specified filtering criteria.

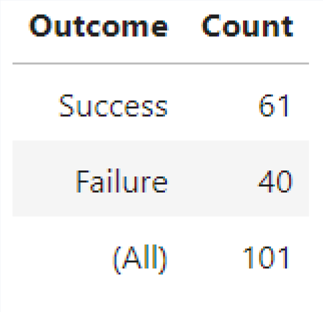
Total Number of Successful and Failure Mission Outcomes

SQL Query code:



```
%%sql
SELECT 'Success' AS "Outcome", count(*) AS "Count"
FROM SPACEXTBL WHERE landing__outcome LIKE 'Success%'
UNION ALL
SELECT 'Failure' AS "Outcome", count(*) AS "Count"
FROM SPACEXTBL WHERE landing__outcome NOT LIKE 'Success%'
UNION ALL
SELECT '(All)' AS "Outcome", count(*) AS "Count"
FROM SPACEXTBL;
```

SQL Query result:



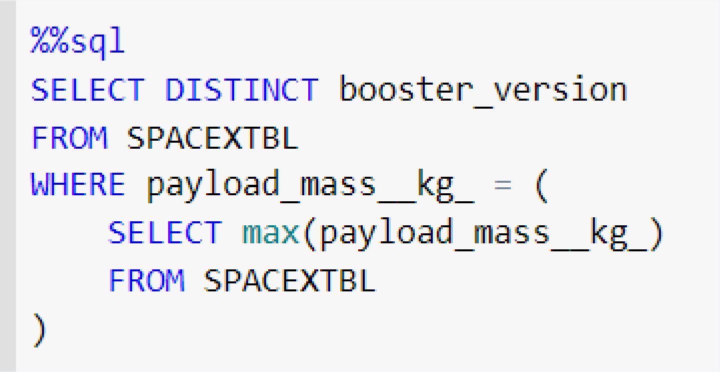
Outcome	Count
Success	61
Failure	40
(All)	101

QUERY EXPLAINATION

This is a complex query that utilizes subqueries to generate the results. The “**like**“ and “not like” operator with the “**Success%**" wildcard is used to search for the phrase “**Success**" within any part of the string in the records. This means that any record that contains the letters “**Success**" in any sequence and location will be included in the first part (**Success**) and any record which does not has “**Success**" will results in the **Failure**.

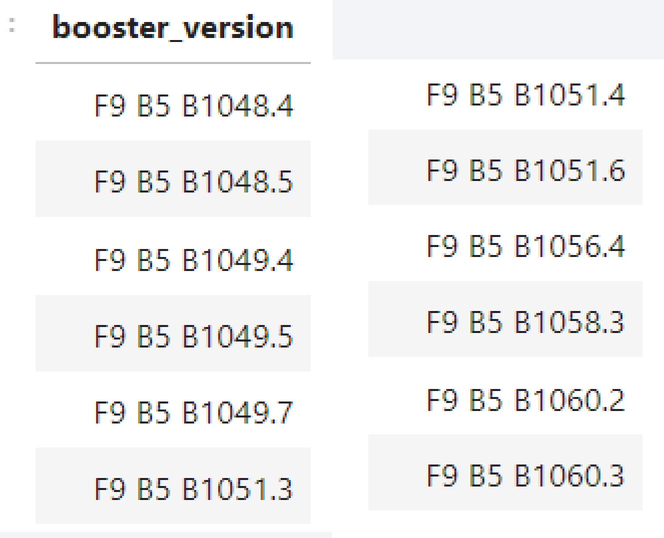
Boosters Carried Maximum Payload

SQL Query code:



```
%%sql
SELECT DISTINCT booster_version
FROM SPACEXTBL
WHERE payload_mass__kg_ = (
    SELECT max(payload_mass__kg_)
    FROM SPACEXTBL
)
```

SQL Query result:



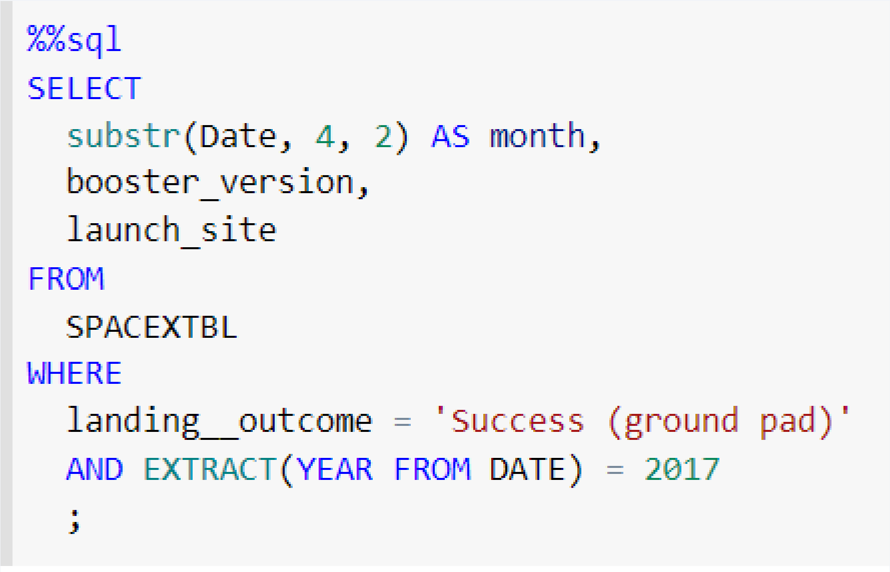
booster_version
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

QUERY EXPLAINATION

This query includes the “**distinct**" keyword to ensure that only unique values are displayed in the "**Booster_Version**" column from the “**SpaceXTBL**" table with the **WHERE** as condition which booster to choose. The Ffunction **MAX** is used to search for the highest payload mass in the column.

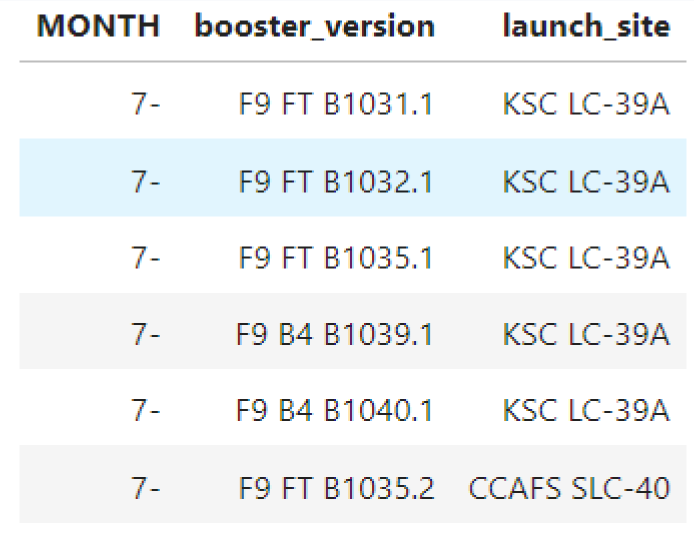
2017 Launch Records

SQL Query code:



```
%%sql
SELECT
  substr(Date, 4, 2) AS month,
  booster_version,
  launch_site
FROM
  SPACEXTBL
WHERE
  landing__outcome = 'Success (ground pad)'
  AND EXTRACT(YEAR FROM DATE) = 2017
  ;
```

SQL Query result:



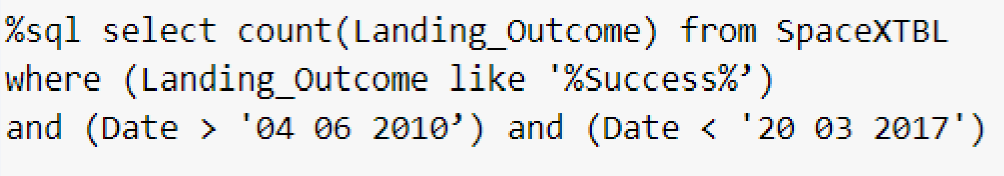
MONTH	booster_version	launch_site
7-	F9 FT B1031.1	KSC LC-39A
7-	F9 FT B1032.1	KSC LC-39A
7-	F9 FT B1035.1	KSC LC-39A
7-	F9 B4 B1039.1	KSC LC-39A
7-	F9 B4 B1040.1	KSC LC-39A
7-	F9 FT B1035.2	CCAFS SLC-40

QUERY EXPLAINATION

In this query we choose our output values to be **booster version**, **launch site** and from the **date** column the **months**. SQLLite does not support monthnames. That is why we used **substr(Date, 4, 2)** as **month** to get the months and **substr(Date,7,4)='2017'** for **year**. Then we choose our conditions to be the **successful** landing outcomes on the **ground pad** from **2017**.

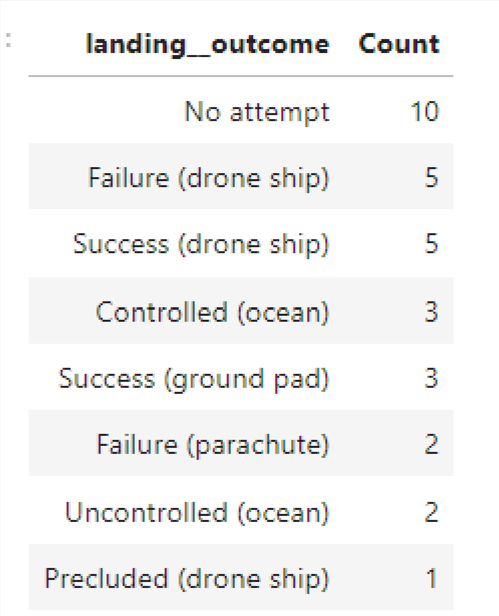
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

SQL Query code:



```
%sql select count(Landing_Outcome) from SpaceXTBL
where (Landing_Outcome like '%Success%’)
and (Date > '04 06 2010’) and (Date < '20 03 2017')
```

SQL Query result:



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

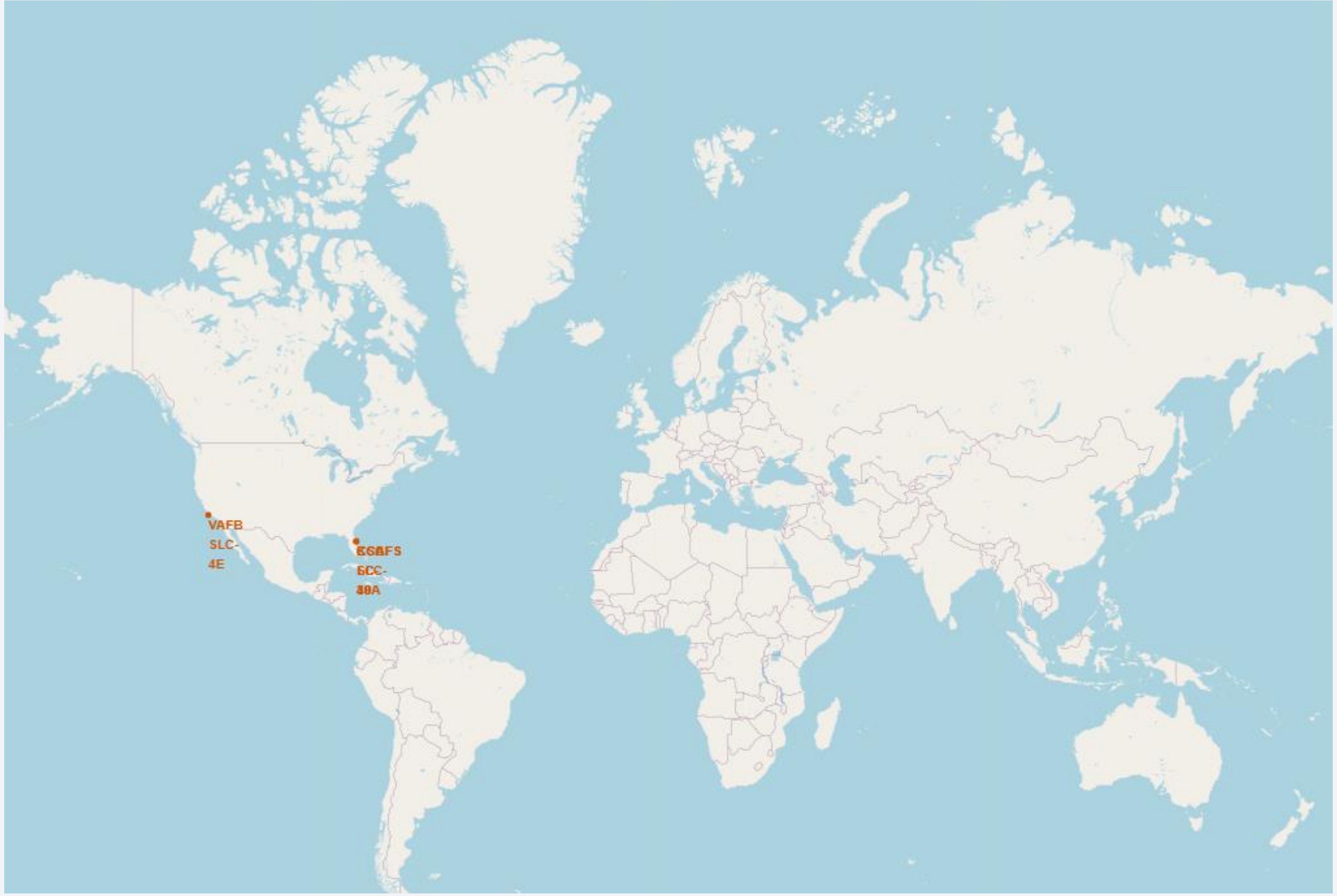
QUERY EXPLAINATION

This statement uses the “**count**" function to count the number of records in a column. The “**where**" clause is then used to filter the data so that only the relevant subset of records are included in the count. This allows for more specific calculations and analysis of the data.

Section 3

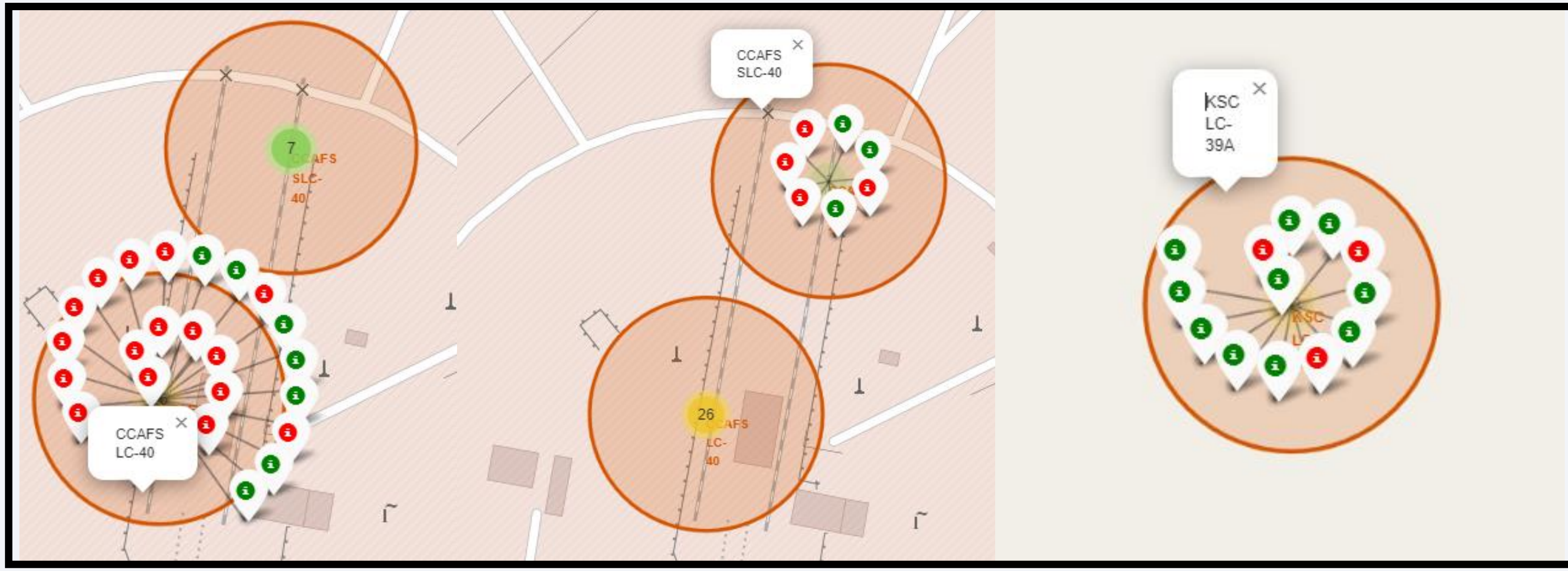
Launch Sites Proximities Analysis

Global markers for the launch sites

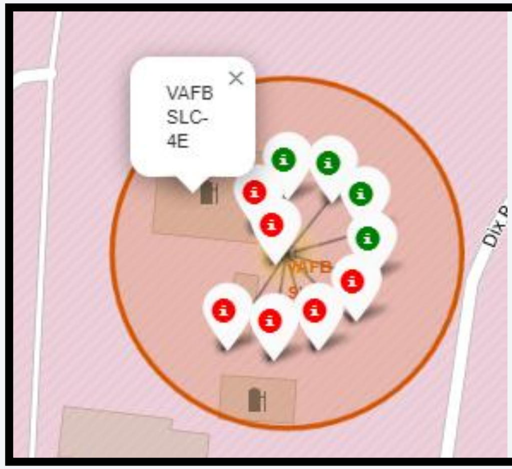


It is apparent that SpaceX's launch sites are situated along the coastlines of Florida and California in the United States of America.

Outcome colored marker for each launch site



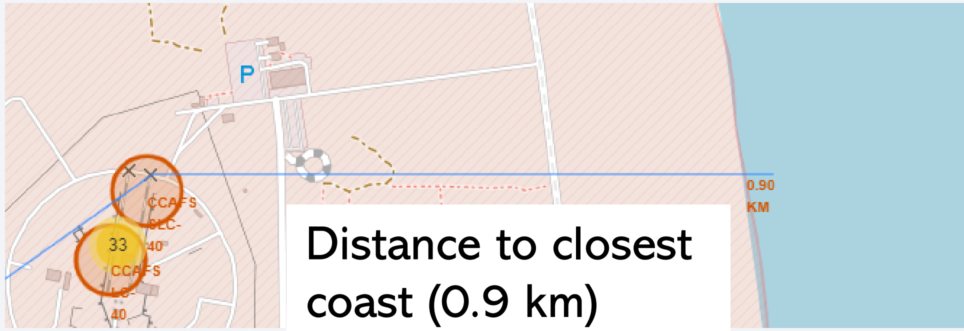
Florida launch site



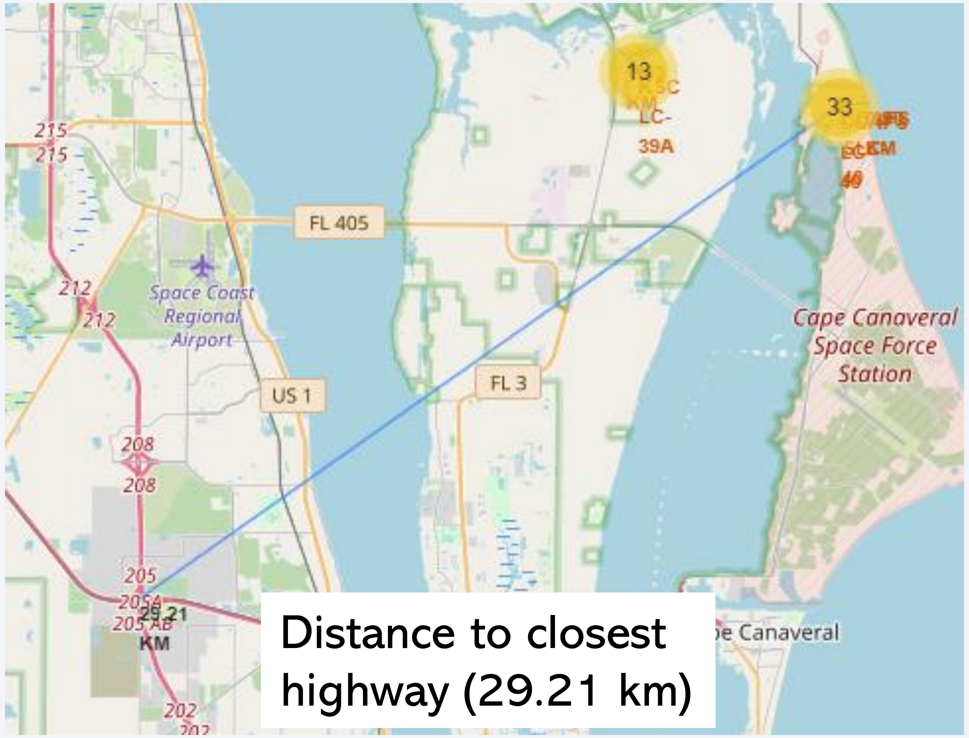
California launch site

The **Green Marker** indicates **successful** launches, while the **Red Marker** indicates **failures**.

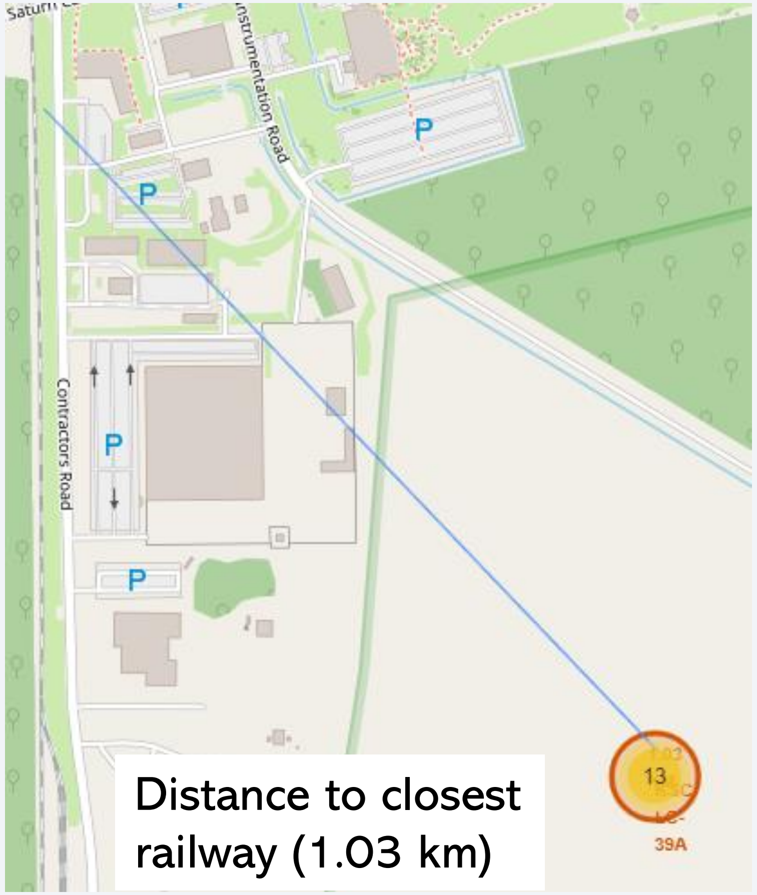
Discovering distances to the closest highway, coast and railway for CCAFS-SLC-40



Distance to closest coast (0.9 km)

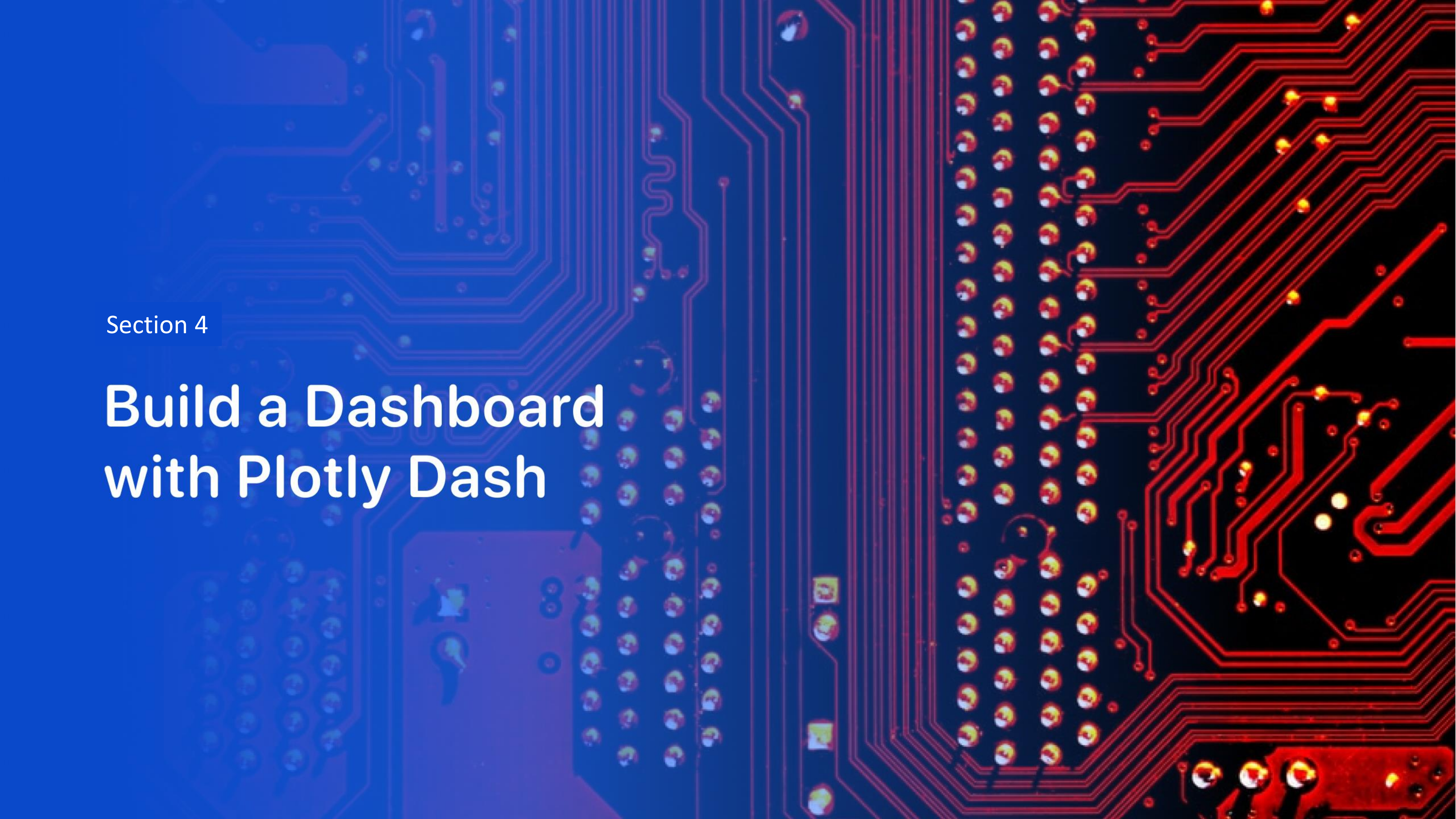


Distance to closest highway (29.21 km)



Distance to closest railway (1.03 km)

Do launch facilities tend to be located near railways? **No.**
Do launch facilities tend to be located near highways? **No**.
Are launch sites typically located near the coastline? **Yes**.
Do launch sites generally maintain a certain distance from cities? **Yes**.



Section 4

Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

- Replace <Dashboard screenshot 1> title with an appropriate title
- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 2>

- Replace <Dashboard screenshot 2> title with an appropriate title

- Show the screenshot of the piechart for the launch site with highest launch success ratio

- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title

- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

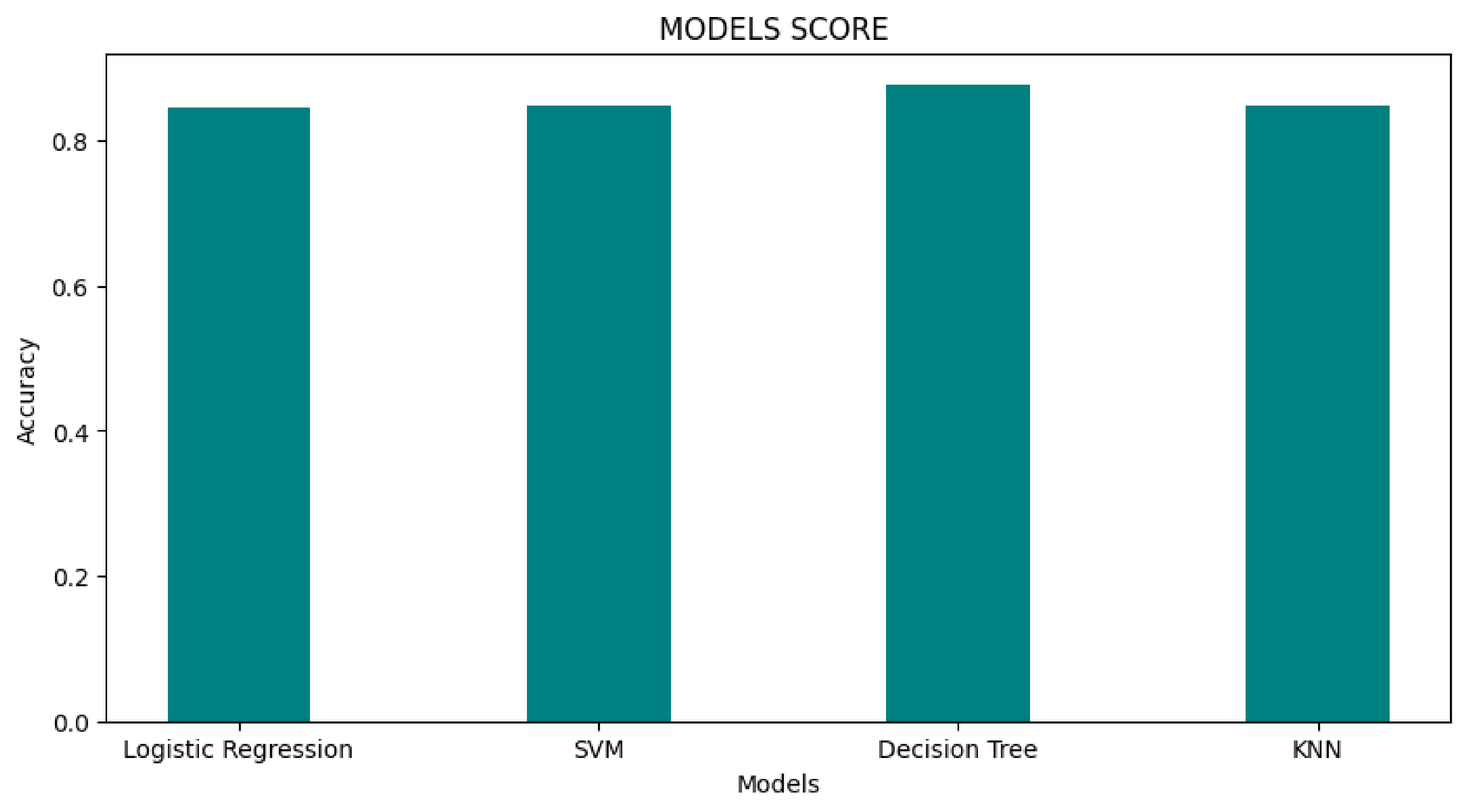
Predictive Analysis (Classification)

Classification Accuracy

It is evident that our accuracy is almost identical, but there is a slight difference when considering decimal places, which ultimately determines the winner when using this function.

Alogrithm	Accuracy
Logistic Regression	0.8464285714285713
SVM	0.8482142857142856
Tree	**0.8767857142857143**
KNN	0.8482142857142858

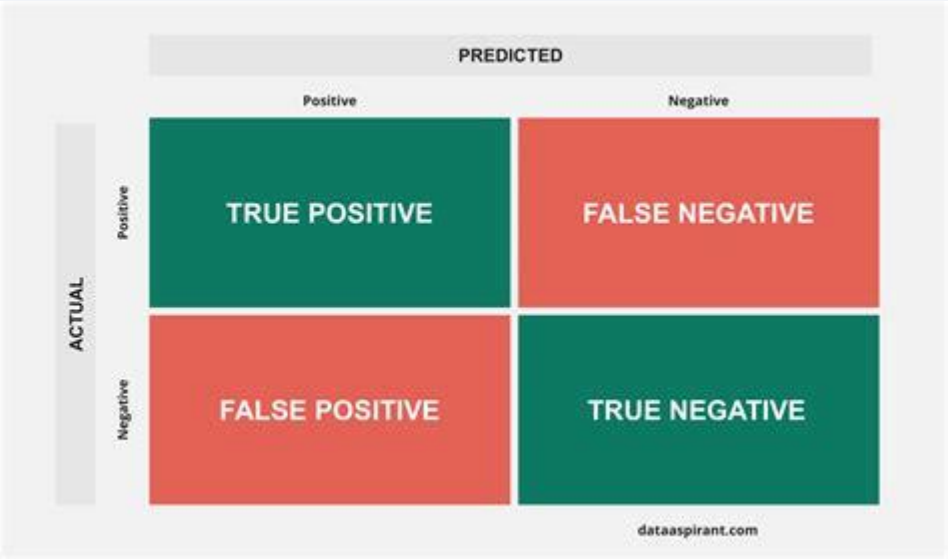
The best model is the tree model.



Confusion Matrix

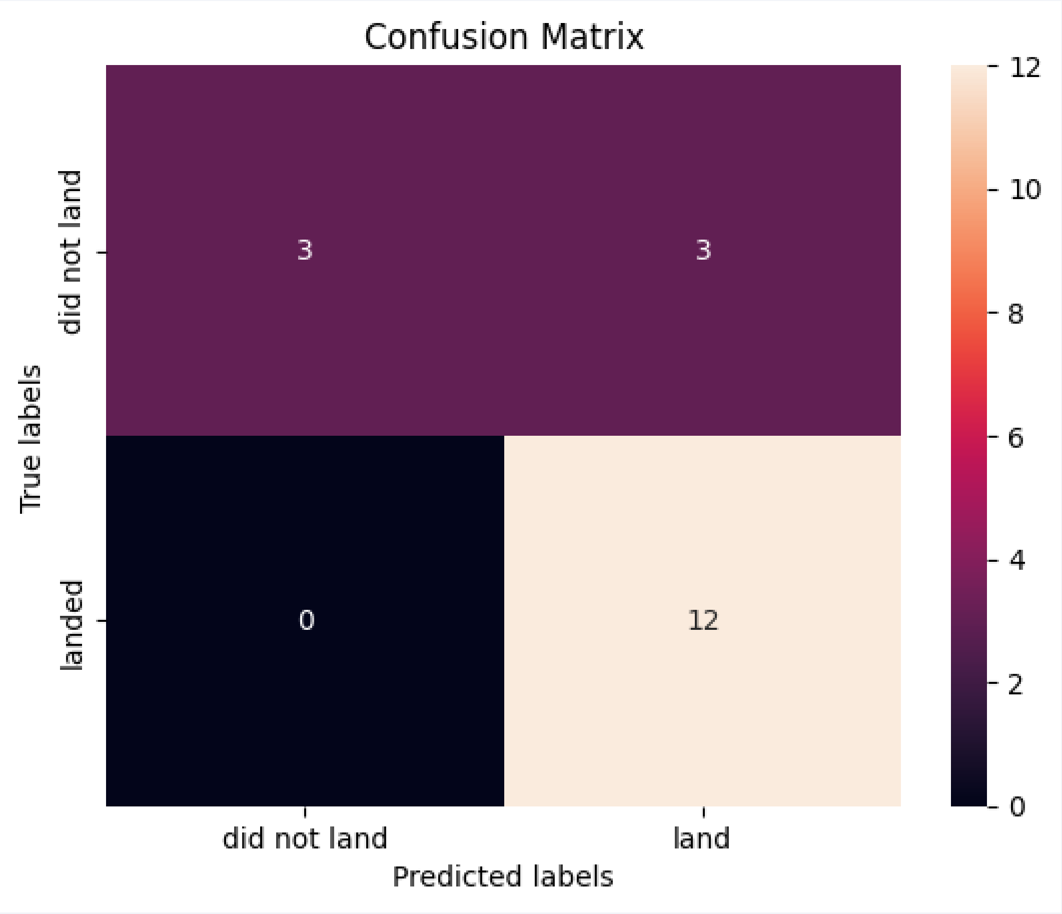
Upon analyzing the confusion matrix, it is apparent that the Tree is able to differentiate between the various classes. However, the primary issue lies with false positives.

Short recap on confusion matrix:



		PREDICTED	
		Positive	Negative
ACTUAL	Positive	TRUE POSITIVE	FALSE NEGATIVE
	Negative	FALSE POSITIVE	TRUE NEGATIVE

dataaspirant.com



Conclusions

1. The Tree Classifier Algorithm is the most effective Machine Learning method for this dataset.

2. Lighter payloads perform better than heavier ones in this context.

3. The more time SpaceX spends on perfecting their launches, the higher their success rates will be.

4. Out of all the launch sites, KSC LC 39A has had the highest number of successful launches.

5. The orbit types GEO, HEO, SSO, and ES L1 have the highest success rates.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!