IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Todd Wilcox
Jan 23, 2024

## Outline

## Executive Summary

- Methodology Summary:
  - Data was gathered from SpaceX APIs and Wikipedia tables
  - Data was cleansed and normalized (null value cleanup, StandardScalar, One-Hot Encoding)
  - Exploratory Data Analysis was done visually through Plotly and through SQL Queries
  - A variety of model building techniques were used to find the most suitable approach
- The data set of landings is limited with under 100 records. While that is not a large sample, it was enough to allow our models to be accurate with scores of 0.833
- Despite all models trained on the training data performing equally well on the test data, the Support Vector Machine model was slightly more accurate when rebuilt using the full set of data, achieving a score of 0.877 with only 4 False Positives

## Introduction

- SpaceX is an American aerospace manufacturer and space transportation company founded by Elon Musk in 2002. Known for its ambitious goals of reducing space transportation costs, SpaceX has achieved numerous milestones, including the development of the Falcon and Starship rockets
- The Falcon 9, a key player in SpaceX's fleet, is a two-stage rocket designed for the reliable and safe transport of satellites and the Dragon spacecraft into orbit. It features reusability, with the first stage capable of returning to Earth for refurbishment and reuse in multiple launches.
- The overall cost of a launch is about $62 million if the first stage can be re-used, but nearly $165MM if it cannot.
- This analysis looks to predict the likelihood or first stage re-use by evaluating past launches and the whether the first stage was recovered for future use.  The following drivers are analyzed in the analysis:
    - Launch Date
    - Payload Mass
    - Launch Site
    - Orbit
    - Booster Version
    - Landing Pad

Section 1

## Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - Data was gathered from two sources, the SpaceX API and the SpaceX Wikipedia Page
- Perform data wrangling
  - Landing success was simplified, Missing Payload Mass data was assigned and categorical fields were re-mapped to enable successful modeling
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Four different modeling techniques (Logistic Regression, Support Vector Machine, Decision Trees and K Nearest Neighbor were evaluated to determine the most predictive

## Data Collection

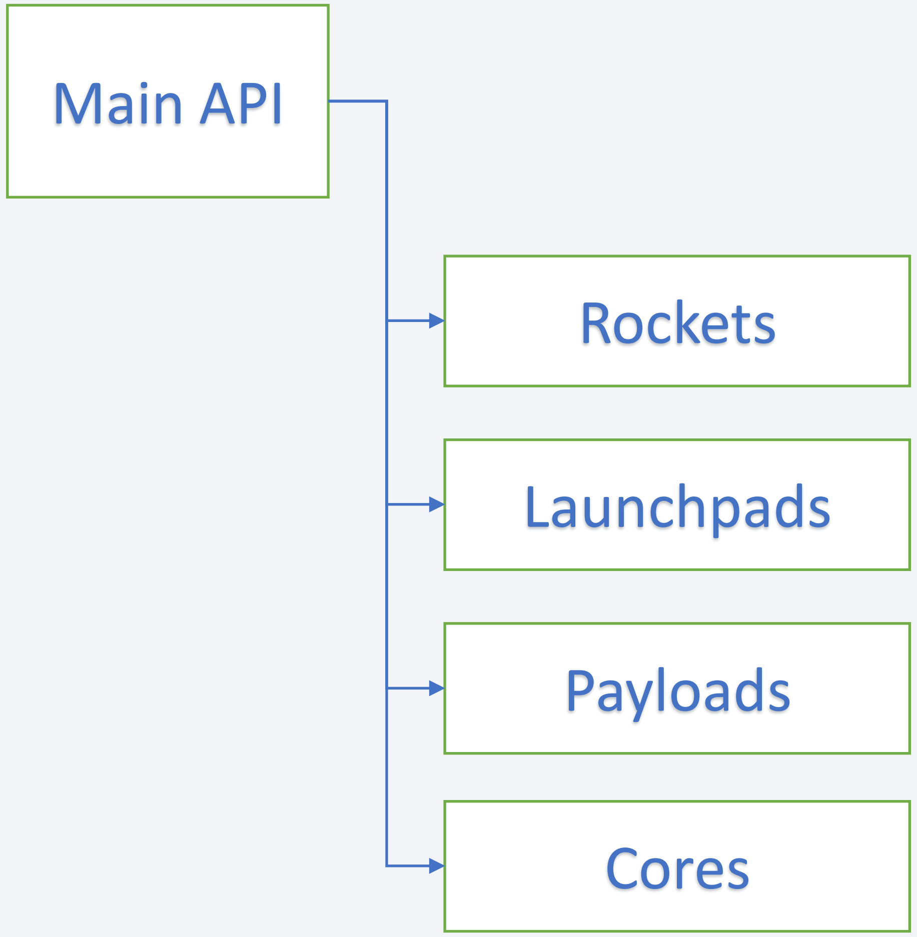
- Data collection started by using the requests and pandas libraries to turn data from several SpaceX REST APIs into Pandas Dataframes
  - spacex_url=https://api.spacexdata.com/v4/launches/past
  - Using the data from this API, we can then use several other REST API sites to get more detail on rockets, launchpads, payloads and rocket cores
- The Beautiful Soup library was then used to parse an HTML table of SpaceX launch information from Wikipedia page.
  - Once extracted, the data was moved into Pandas Dataframes and csv files for future use

## Data Collection – SpaceX API

The main SpaceX API site provides basic information and keys that can be used to gather more detailed information: https://api.spacexdata.com/v4/launches/past

- https://api.spacexdata.com/v4/rockets/*
- https://api.spacexdata.com/v4/launchpads/*
- https://api.spacexdata.com/v4/payloads/*
- https://api.spacexdata.com/v4/cores/*

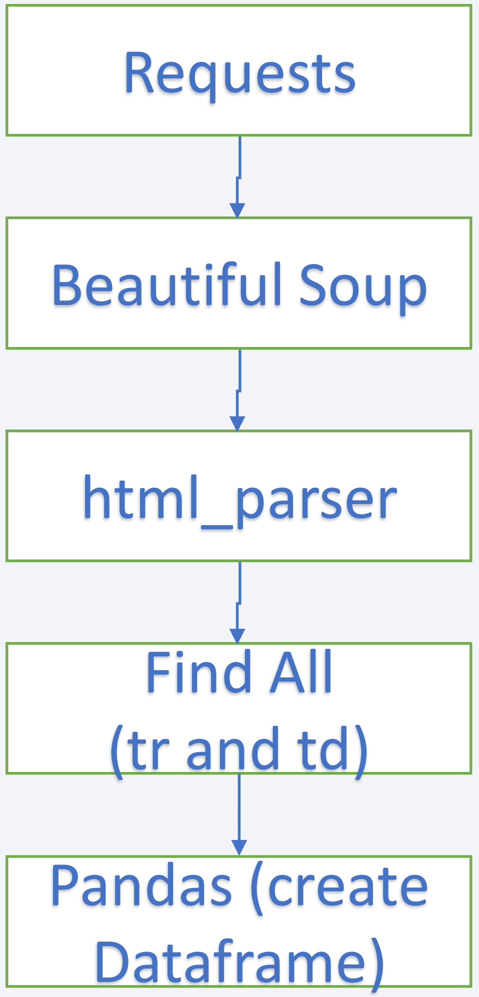
- Requests, Pandas and Numpy were used to extract json files and stage the data.  Fillna() was used to replace the null values in Payload Mass with the mean mass
- Detailed work can be found in: https://github.com/nolimpers/IBM-Class-Capstone/blob/main/Lab%201%20-%20jupyter-labs-spacex-data-collection-api.ipynb



## Data Collection - Scraping

- Requests was used to extract HTML data from the SpaceX Wikipedia page.
- Beautiful Soup was used parse the html.
- The find_all functionality enabled the extraction of table rows

- Detailed work can be found in: https://github.com/nolimpers/IBM-Class-Capstone/blob/main/Lab%202%20-%20jupyter-labs-webscraping.ipynb



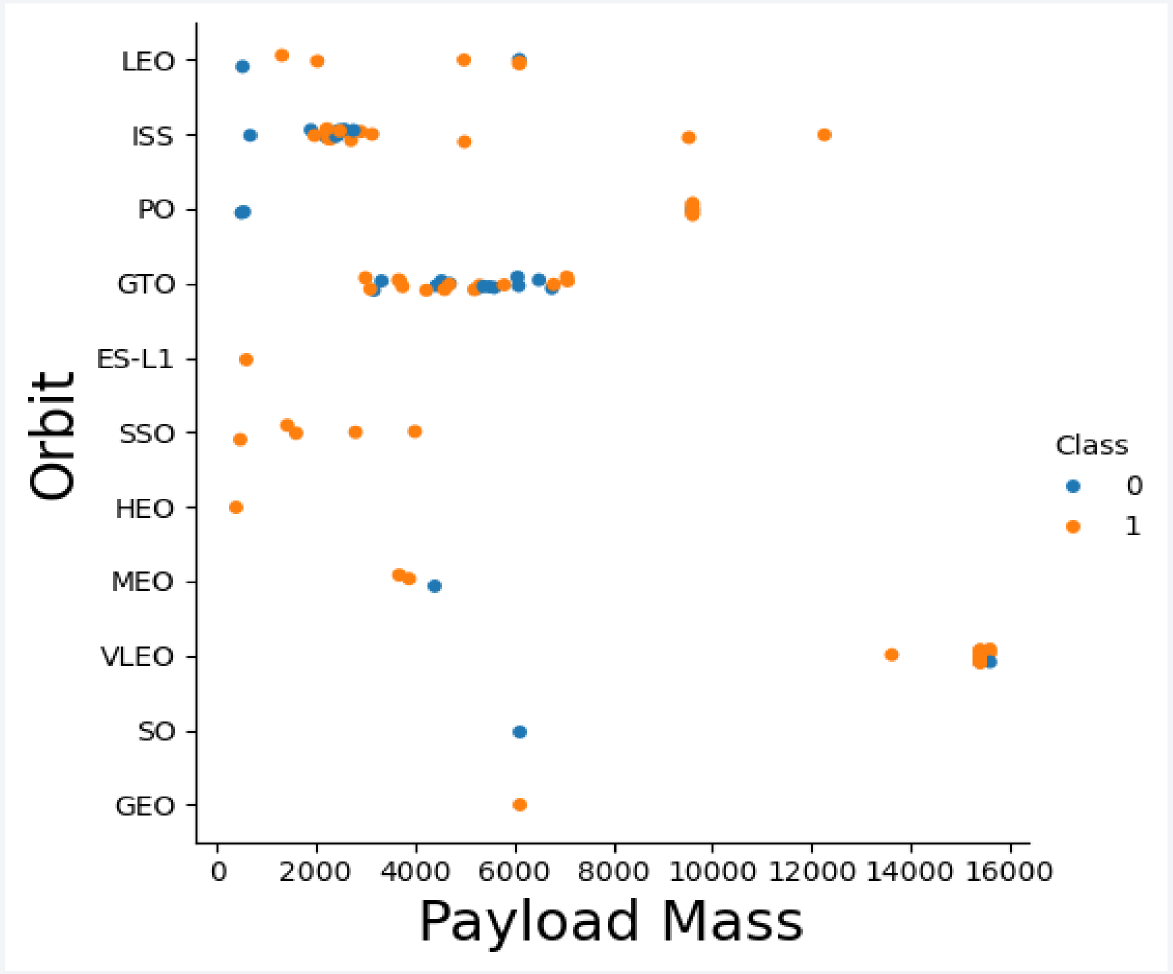
## Data Wrangling

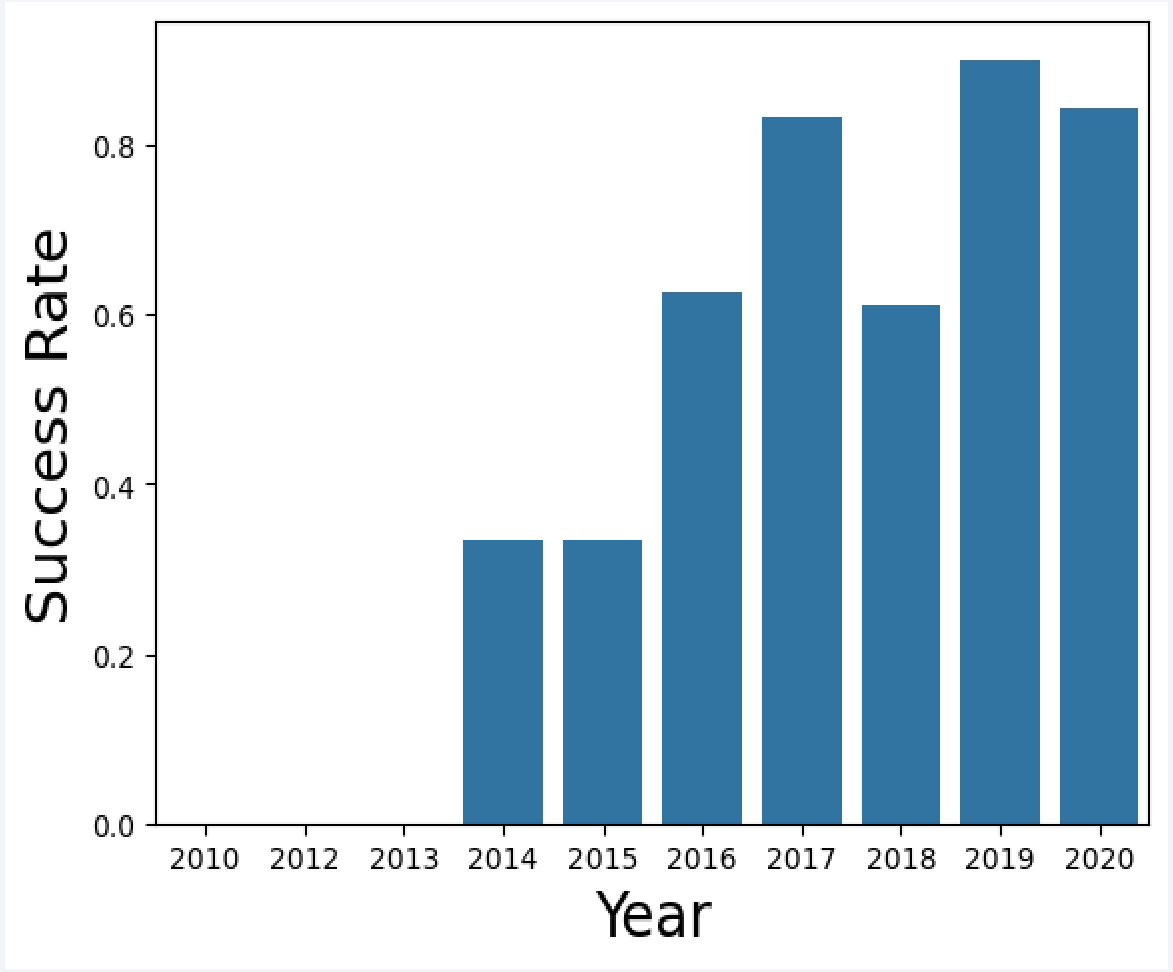
- Data from the API and Web Scraping processes were stored as .csv files, which were loaded into Pandas using the read_csv method.
- Fields were searched for null values and found that only LandingPad had nulls (29% of records). These were allowed to remain because that is a true reflection of the data (no Landing Pad was used)
  - Note: The earlier missing values on Payload Mass had already been corrected.
- Launch outcomes (later referred to as 'Class') were simplified from 8 values down to a simple 1 = Success, 2 = Failure field.
- Detailed work can be found in:
  - https://github.com/nolimpers/IBM-Class-Capstone/blob/main/Lab%203%20-jupyter-spacex-Data%20wrangling.ipynb

## EDA with Data Visualization - Process

- A variety of charts were created to explore the data and get a sense for which fields were the most important:
  - Scatterplots (with colored markers where needed) were used to evaluate the relationship between 2-3 columns at a time, including Payload vs Flight, Launch Site vs Flight, Launch Site vs Payload, Orbit vs Payload, and others
  - Bar Charts were used to evaluate the success rate of missions by Orbit and Year
  - The following tab contains a few informative charts, showing:
    - The Geosynchronous Orbit (GTO) and missions to the International Space Station (ISS) have the most failures
    - Success Rate of missions has increased significantly over time
- Detailed work can be found in : https://github.com/nolimpers/IBM-Class-Capstone/blob/main/Lab%205%20-%20jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

# EDA with Data Visualization - Charts





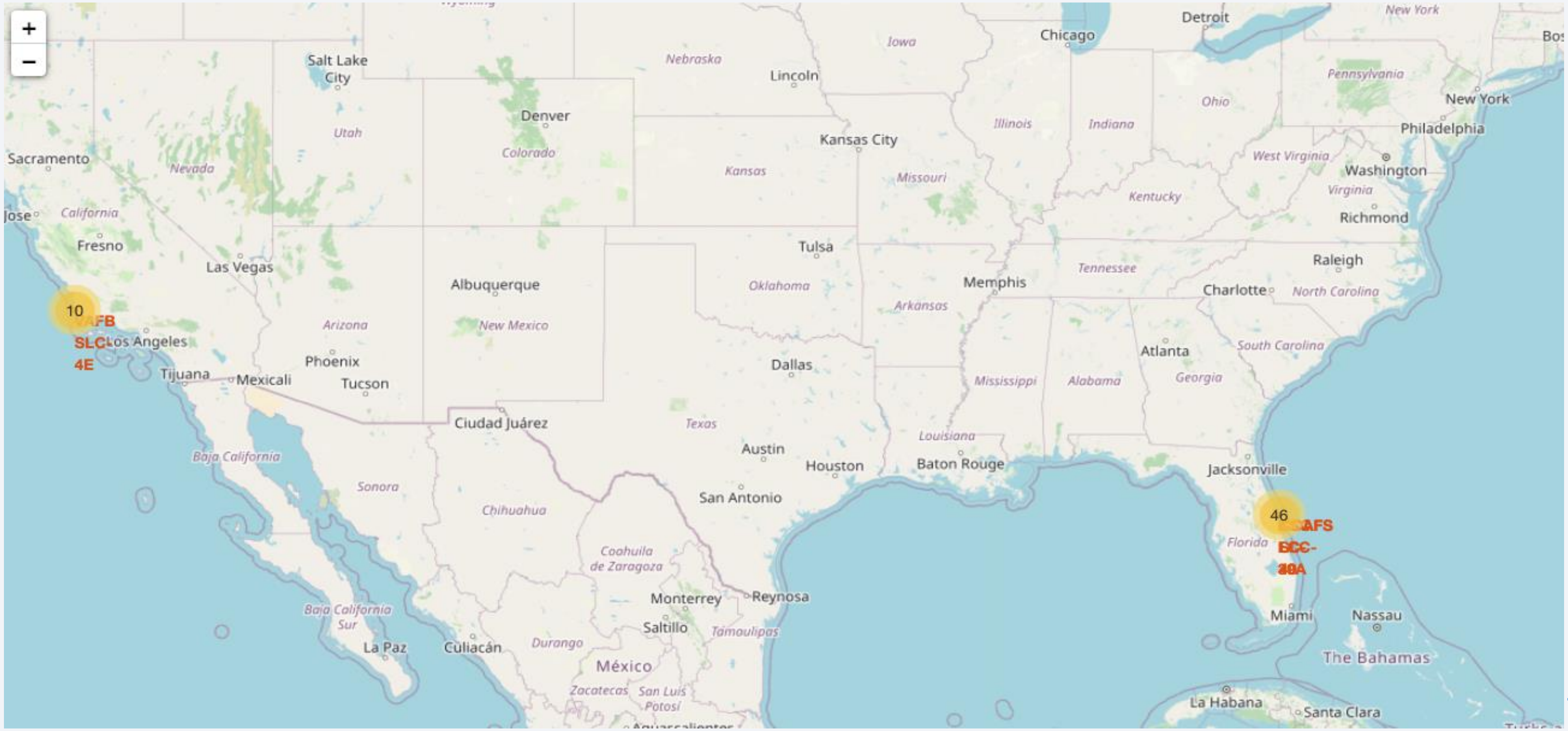
## EDA with SQL

- Many pieces of investigation were done using SQL queries, including:
  - Identifying unique Launch Sites
  - Finding the total Payload Mass associated with NASA (CRS) missions
  - The average payload mass for F9 v1.1 boosters
  - The date of the first successful Landing Outcome
  - Mission Outcome Frequencies
  - Booster versions that carry the maximum payload
  - Landing Outcomes (which includes the attempted method, e.g. drone ship, ocean…)
- Detailed work can be found in: https://github.com/nolimpers/IBM-Class-Capstone/blob/main/Lab%204%20-%20jupyter-labs-eda-sql-coursera_sqllite.ipynb

## Build an Interactive Map with Folium

- Folium was used to map the launch site locations to a map of North America. It shows the locations in Florida and California.
- Markers for each launch site were added and the Folium MarkerCluster plugin was used to add the launch result to be available interactively for each site.
- The MousePosition plugin was used to identify the latitudes and longitudes of nearby areas like oceans, cities and railways. Markers and lines (using the PolyLine) were added to highlight those distances.
- Detailed work can be found in: https://github.com/nolimpers/IBM-Class-Capstone/blob/main/Lab%206%20-%20jupyter_launch_site_location.jupyterlite.ipynb

## Build an Interactive Map with Folium



# Build a Dashboard with Plotly Dash

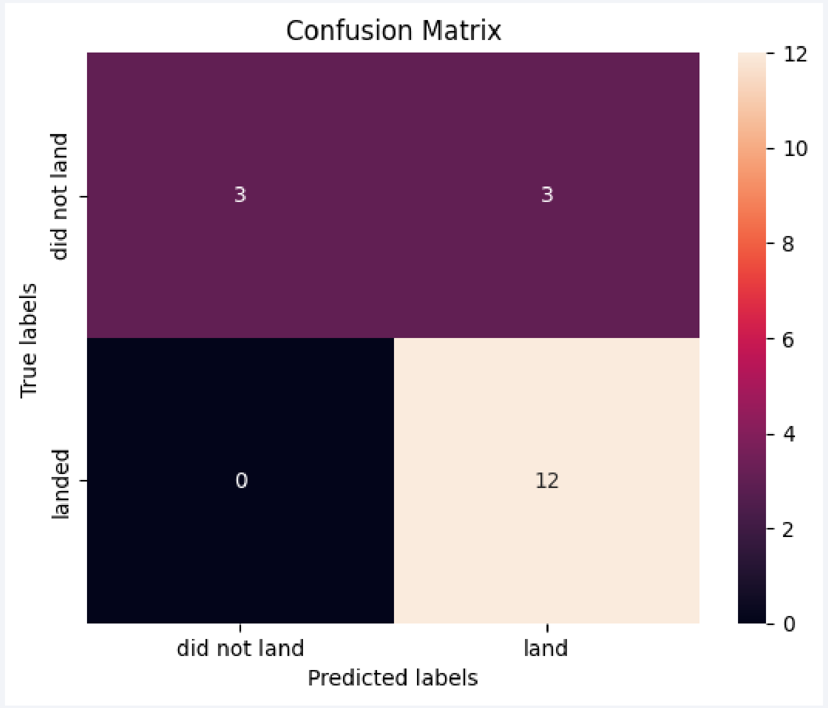
- An interactive dashboard was created using Plotly (for chart creation) and Dash (for interactivity and web hosting). This also involved:
    - Creating a Launch Site dropdown and Payload Mass Slider for interactive use
    - Creating Callback functions to link the values in the dropdown and slider to the charts themselves
- A pie chart was created so that users can see the Launch Site distribution, and then see the success rate of launches if they choose to drill into a specific site.
- A scatter chart was create to show the correlation between payload and launch success, with colored markers to show the Booster Versions.
- Detailed code can be found in: https://github.com/nolimpers/IBM-Class-Capstone/blob/main/Interactive%20Dashboard%20Code.py.txt
- An image of the dashboard can be found at: https://github.com/nolimpers/IBM-Class-Capstone/blob/main/Plotly%20Interactive%20Dashboard.png

## Predictive Analysis (Classification) Part 1

- A dataframe that had been one-hot-encoded was combined with a dataframe showing landing success were used for model development and testing.
- The numpy function train_test_split was used to create an 80% training sample and 20% testing sample.
- The sample was the normalized using Preprocessing and StandardScalar,then placed into a GridSearchDV object.
- From there, four different methodologies were used to model the training data and score the test data:
    - Logistic Regression
    - Support Vector Machine
    - Decision Trees
    - K Nearest Neighbor

## Predictive Analysis (Classification) - continued

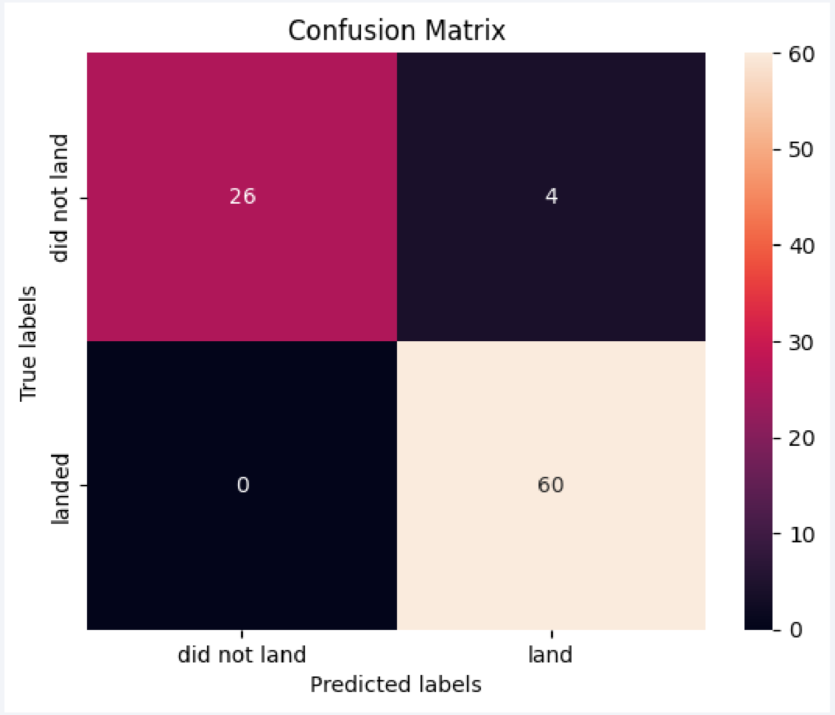
- All four methodologies came up with models with exactly the same score of 0.833.
- All four methodologies also sorted the data into the same confusion matrix with 3 records mis-predicted as landing when they actually did not.



- To break the tie, when each model was tested on the training population, the Support Vector Machine ended up being the most predictive:
  - Support Vector Machine: 0.888
  - Logistic Regression: 0.875
  - Decision Tree: 0.861
  - K-Nearest Neighbor: 0.861

# Predictive Analysis (Classification) - continued

- Because all 4 methods produced comparable results, all 4 were scored on the whole population.
- Again, the Support Vector Machine had the highest score. Interestingly, the kernel changed from sigmoid to linear, though. The confusion also looks improved with only 4 false positives.



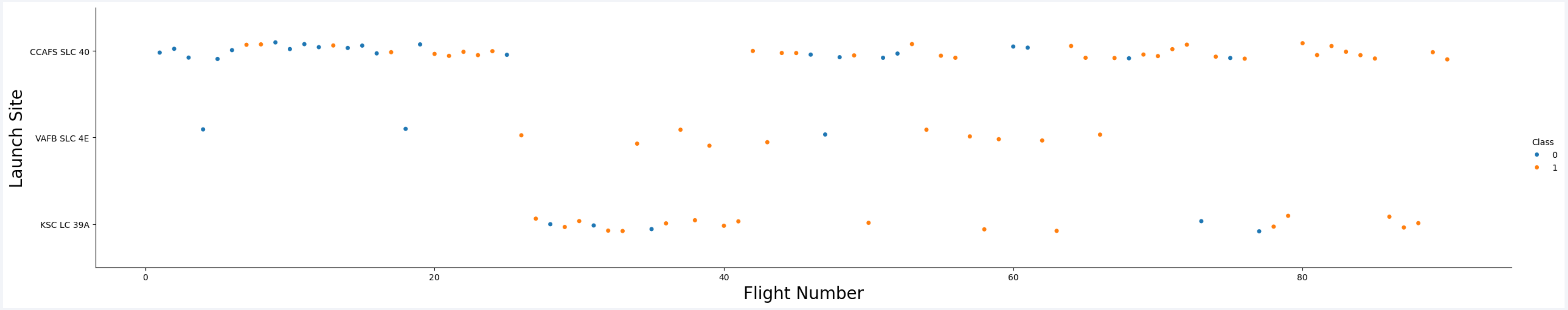
SVM Score on whole population: 0.877

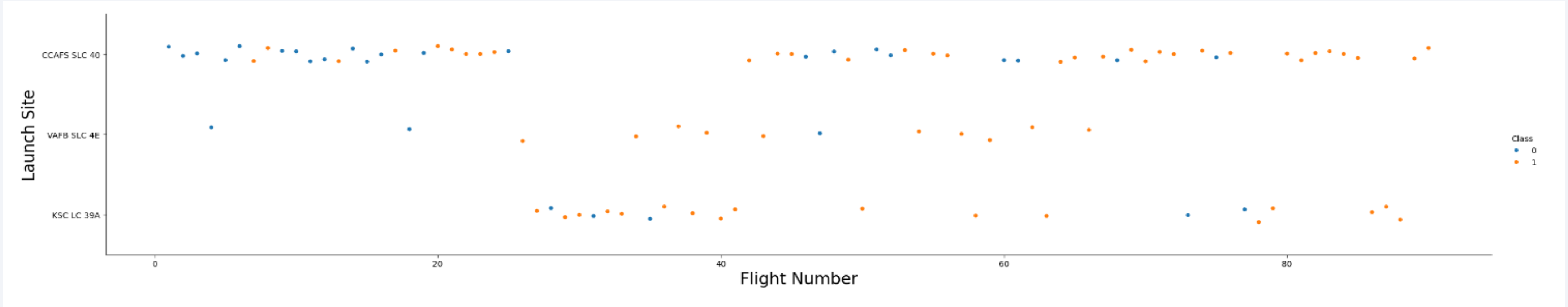
Detailed code can be found in: https://github.com/nolimpers/IBM-Class-Capstone/blob/main/Lab%208%20-%20SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Section 2

## Insights drawn from EDA

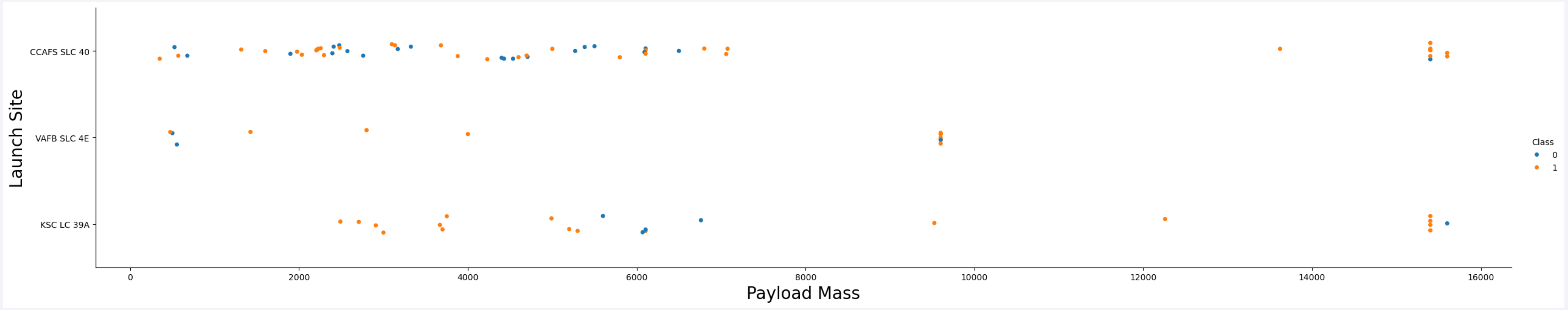
# Flight Number vs. Launch Site

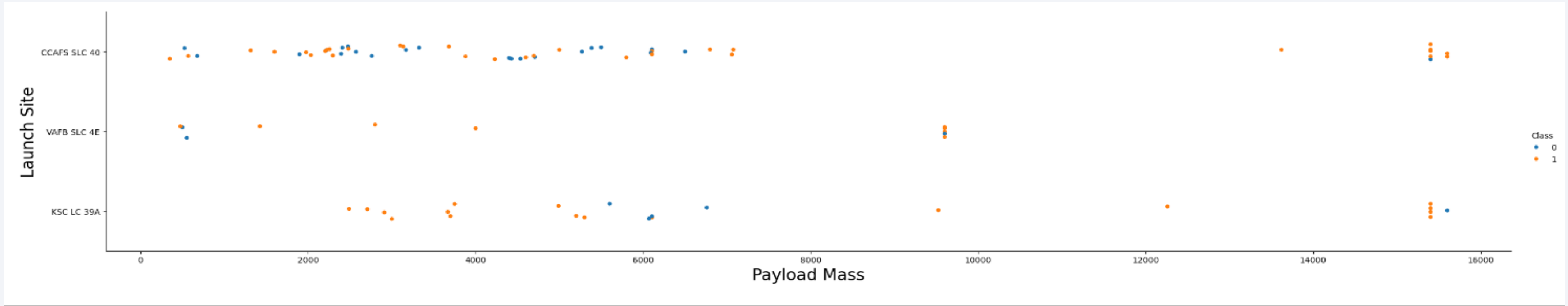




**### Insights: CCAFS SLC 40 was the main launch site early on, back when failures were frequent.**
**#### Success becomes more regular around flight 20.**
**#### Other sites became more regularly used around flight 28, so their success rate is higher as expected.**

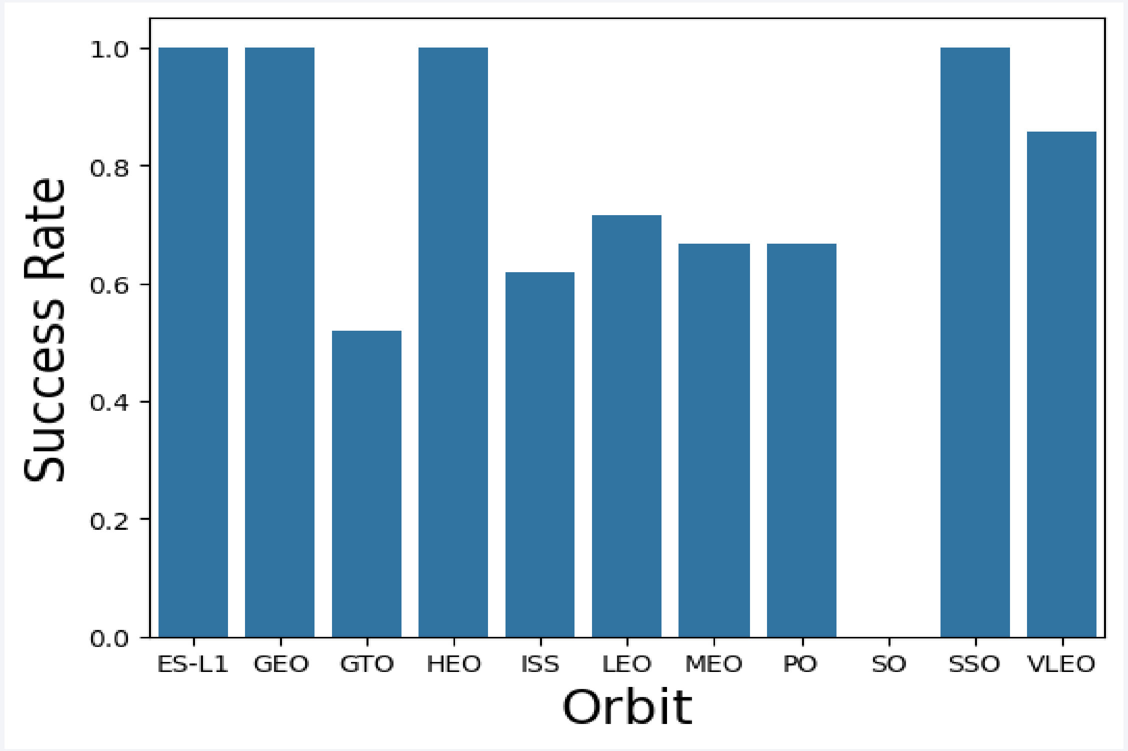
# Payload vs. Launch Site

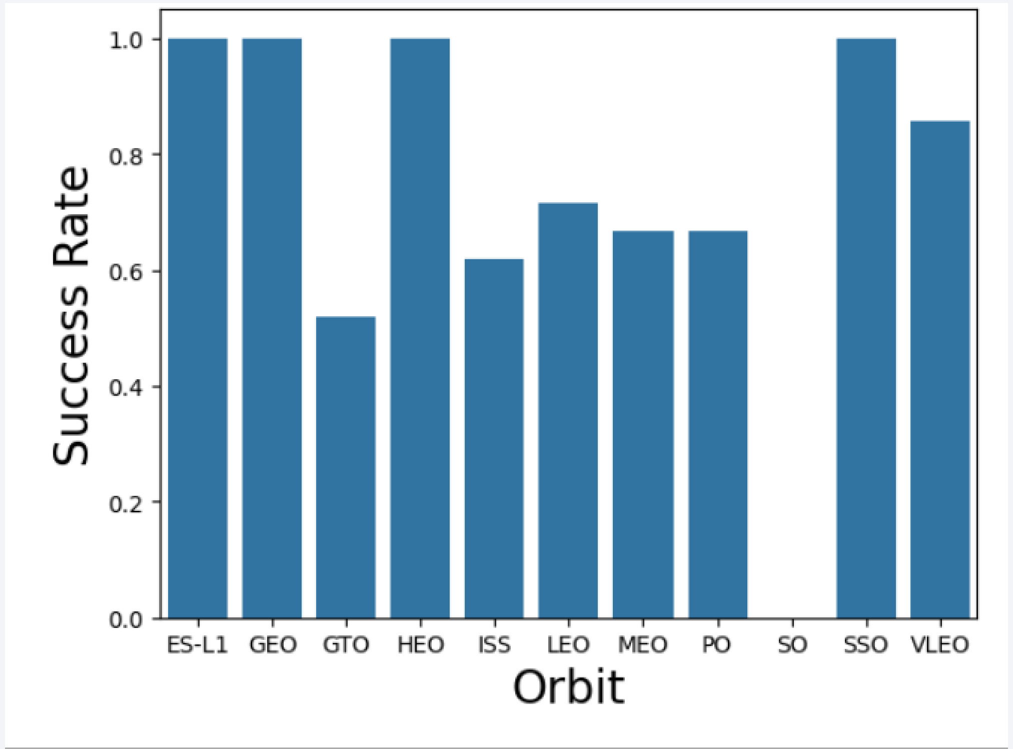




**### Insights: The CCAFS and KSC sites look capable of handling launches of any payload mass**
**#### The VAFB site looks to be useful only for payloads under 10k kg**
**#### Success rates seem fairly indepenent of payload**

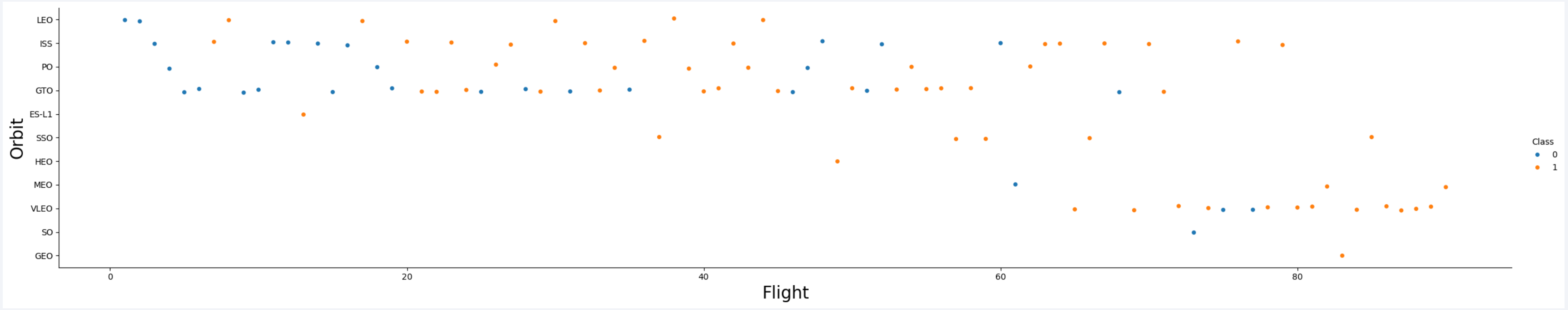
# Success Rate vs. Orbit Type

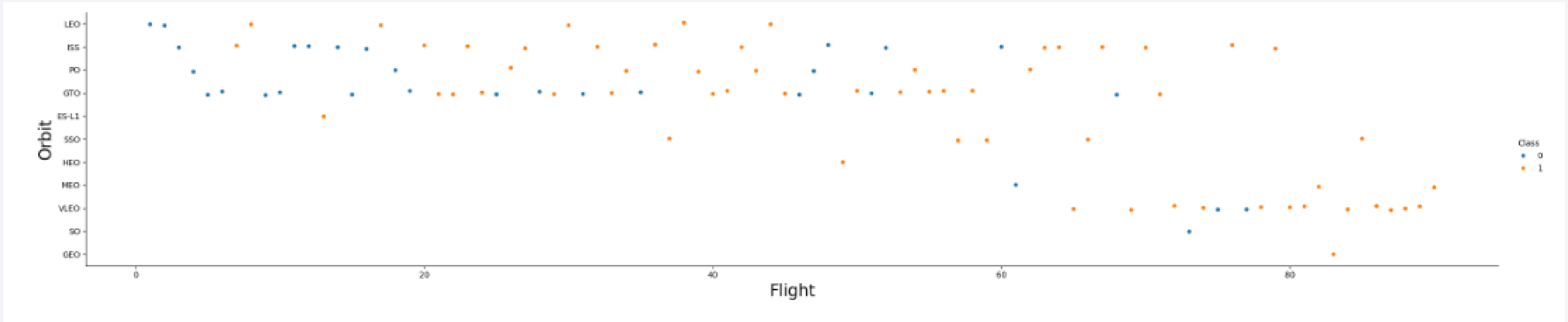




### Insights: Success rates vary heavily by Orbit
#### Geosynchronous and ISS are the most risky

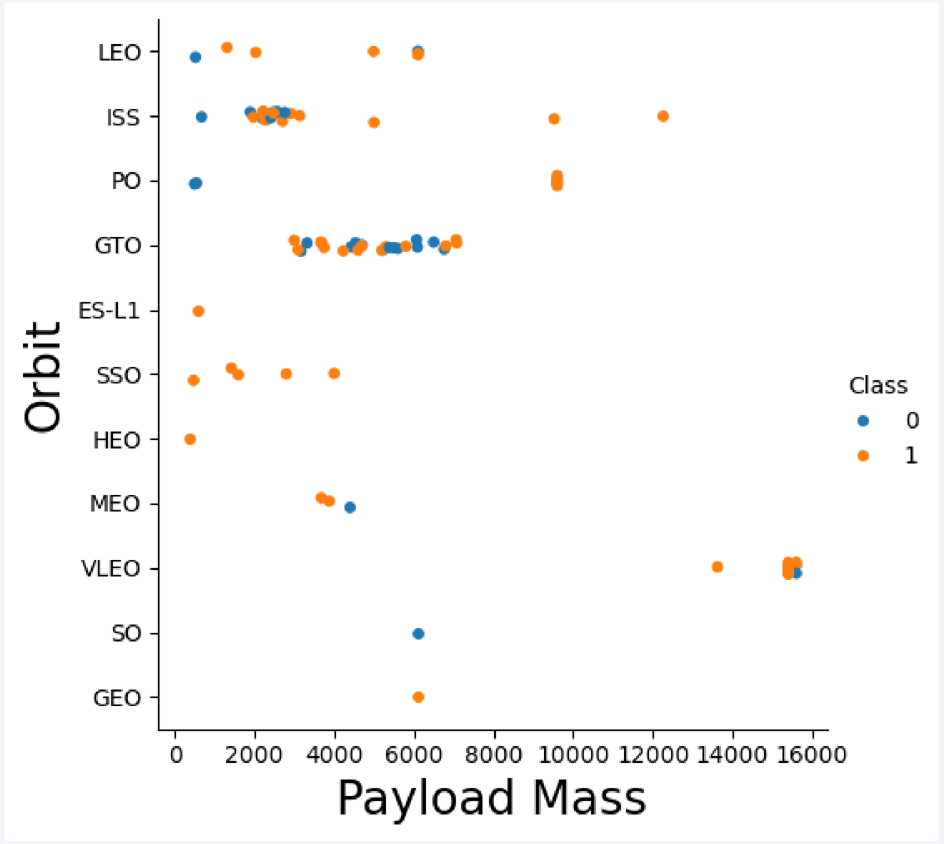
# Flight Number vs. Orbit Type

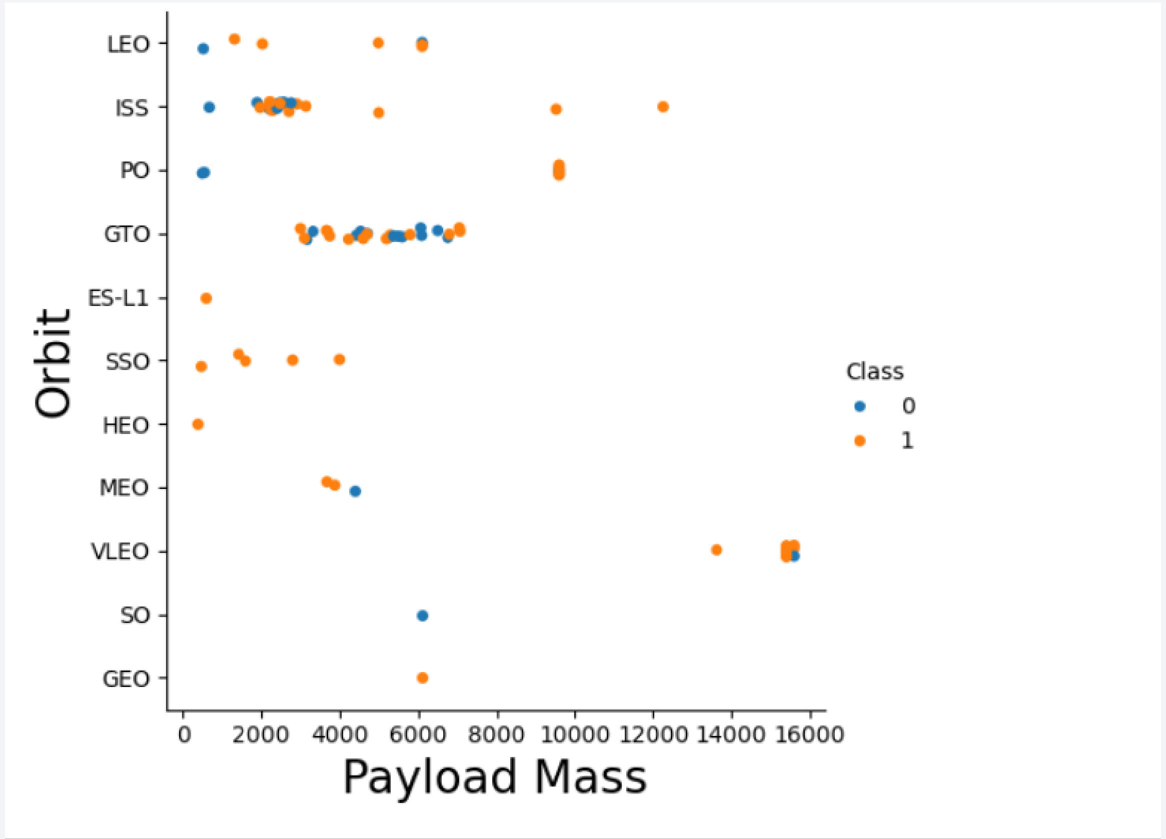




**### Insights:  The success rate of some orbits may be based on flight number versus an inherent low risk level**
**#### For instance, the Very Low Earth Orbit (VLEO) has a very good success rate, but didn't start until flight 67**

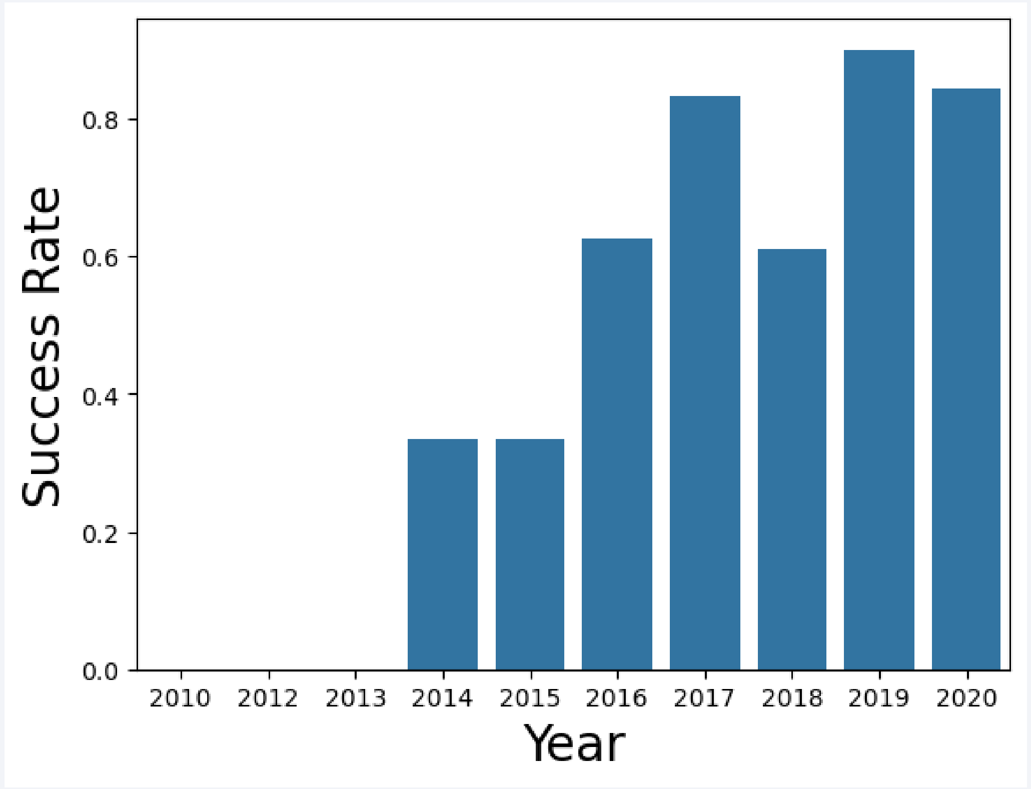
## Payload vs. Orbit Type

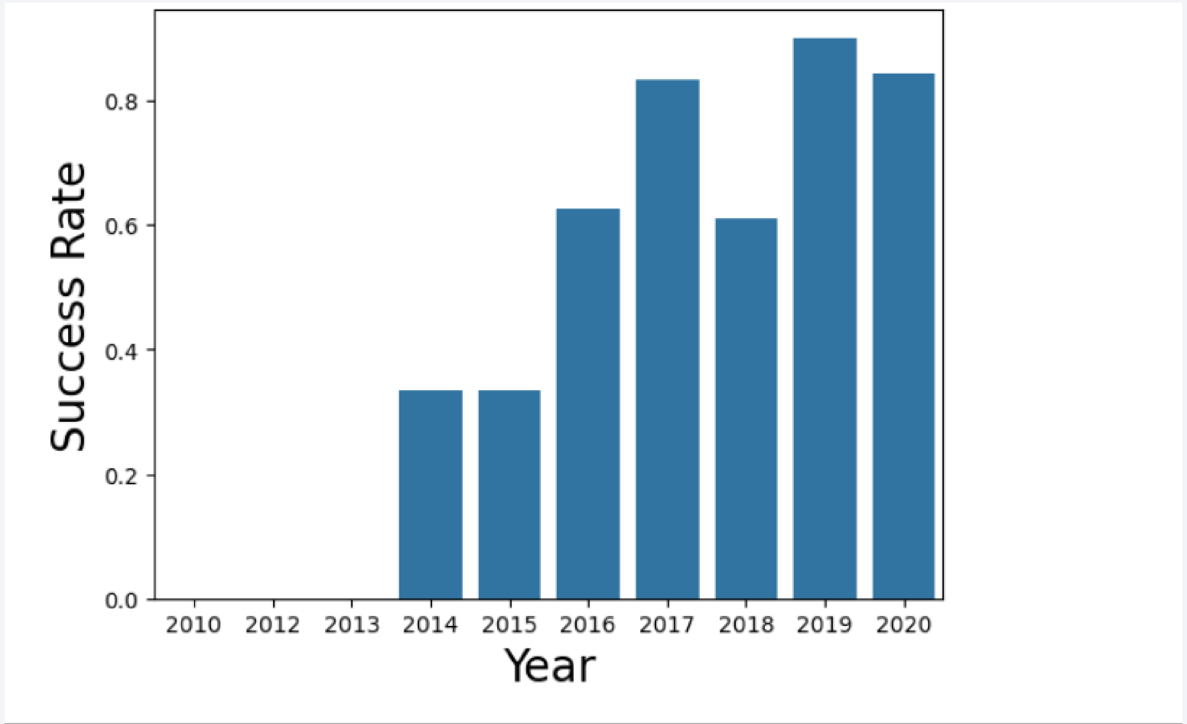




### Insights: Payload mass seems to have fairly low variability for payload mass.
#### For instance, most trips to the ISS have a payload mass between 2000 and 4000 kg

## Launch Success Yearly Trend





### Insights: Launch Success Rate seems to have a very positive linear correlation

## All Launch Site Names

Find the names of the unique launch sites

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

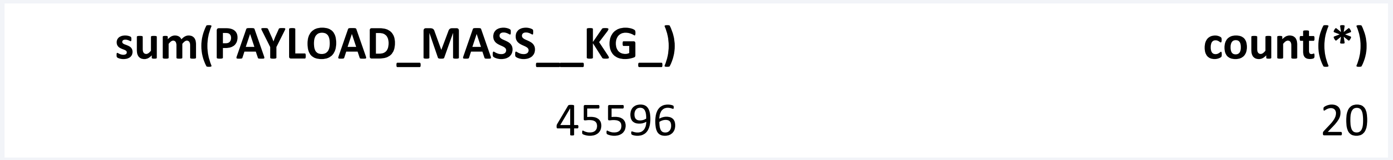
- %sql – SQL Magic operator to run SQL queries in Jupyter Notebooks
- 'distinct' gives you the unique values

## Launch Site Names Begin with 'CCA'

Find 5 records where launch sites begin with `CCA`

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

- The like operator is for pattern matching
- The % on either side of the CCA will match to anything (or nothing)
- Limit 5 reduces your output to only 5 records



| sum(PAYLOAD_MASS__KG_) | count(*) |
|---|---|
| 45596 | 20 |

# Total Payload Mass

Calculate the total payload carried by boosters from NASA (CRS)

%sql select sum(PAYLOAD_MASS__KG_), count(*) from SPACEXTABLE where customer ='NASA (CRS)’

- The sum function sums up the values in ()
- The count(*) counts the records (not required here but good to see)
- The where clause is needed limit the records to NASA (CRS) records

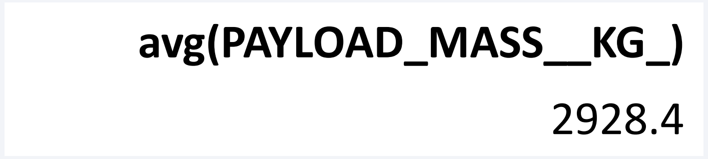
| sum(PAYLOAD_MASS__KG_) | count(*) |
|---|---|
| 45596 | 20 |

## Average Payload Mass by F9 v1.1

Calculate the average payload mass carried by booster version F9 v1.1

%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version ='F9 v1.1'

- The avg function finds the mean of the field in the ()



| avg(PAYLOAD_MASS__KG_) |
|---:|
| 2928.4 |

## First Successful Ground Landing Date

Find the dates of the first successful landing outcome on ground pad

%sql select min(date) from SPACEXTABLE where Landing_Outcome like ('%Success%')

- The min function finds the lowest value for the date field
- Landing Outcome has a lot of different values that show success, and they all have the word Success in them, so a like function is used to find them all

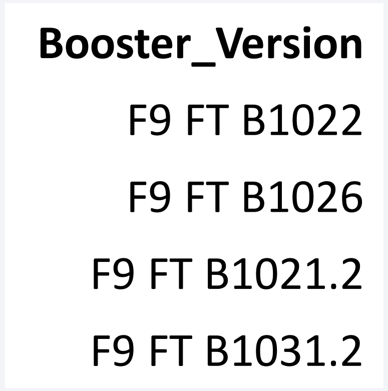
| min(date) |
|---:|
| 2015-12-22 |

## Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
%sql select distinct Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_<6000 and Landing_Outcome='Success (drone ship)'
```

- There isn't much new here except that I'm using a payload > 4000 and < 6000 combo (which could be done as a BETWEEN in some SQL platforms)



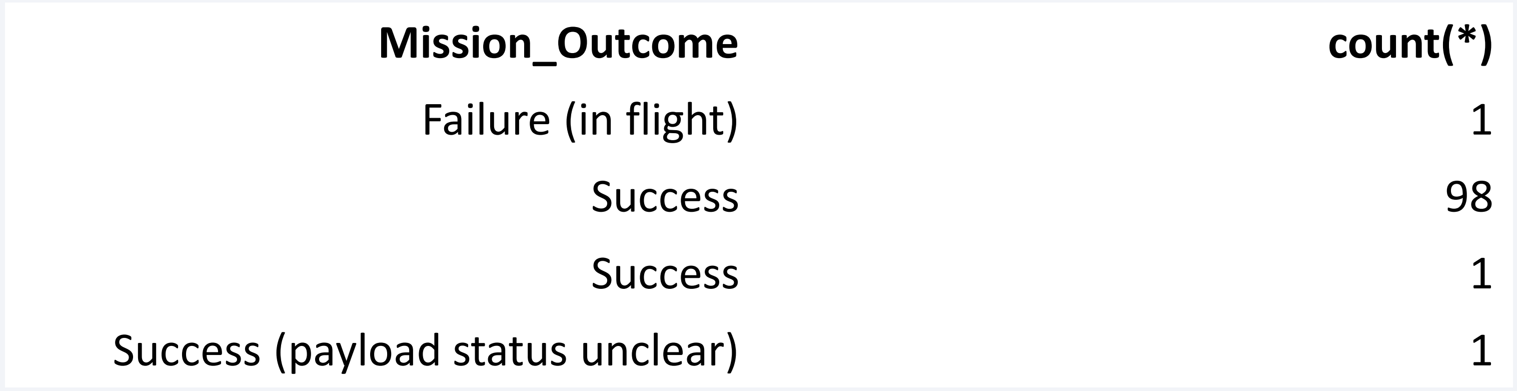
| Booster_Version |
|---|
| F9 FT B1022 |
| F9 FT B1026 |
| F9 FT B1021.2 |
| F9 FT B1031.2 |

## Total Number of Successful and Failure Mission Outcomes

Calculate the total number of successful and failure mission outcomes

%sql select Mission_Outcome, count(*) from SPACEXTBL group by 1 order by 1

- We have our first group by here, which is needed when using aggregate functions.
- We have our first order by here which sorts the records in ascending order
- Both clauses make use of '1' (group by 1 order by 1), which is a shortcut for saying the first field from the select list



| Mission_Outcome | count(*) |
|---:|---:|
| Failure (in flight) | 1 |
| Success | 98 |
| Success | 1 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload

List the names of the booster which have carried the maximum payload mass

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

- This query needs to be done with a subquery – the subquery finds the maximum value of the payload
- Then the result of that subquery is used in an “in” clause, which could have been an equal here as an alternative.  “In” is usually used for a list of items.

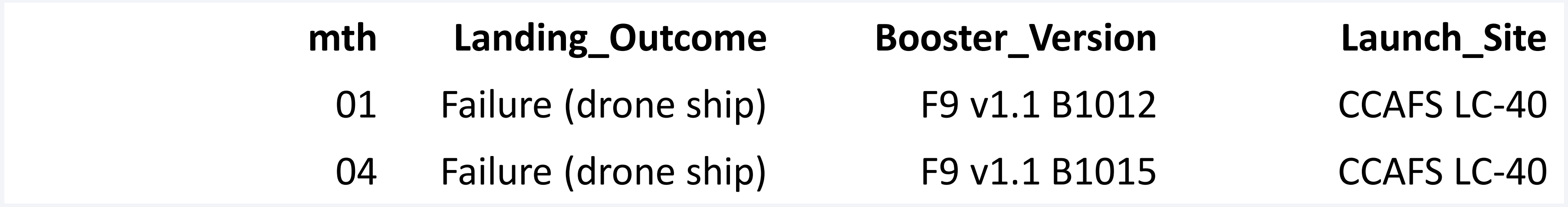
| **Booster_Version** |
|---|
| F9 B5 B1048.4 |
| F9 B5 B1049.4 |
| F9 B5 B1051.3 |
| F9 B5 B1056.4 |
| F9 B5 B1048.5 |
| F9 B5 B1051.4 |
| F9 B5 B1049.5 |
| F9 B5 B1060.2 |
| F9 B5 B1058.3 |
| F9 B5 B1051.6 |
| F9 B5 B1060.3 |
| F9 B5 B1049.7 |

# 2015 Launch Records

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

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

- The substring function is used twice here. That function is like slicing a string, in this case parsing out a date in YYYY-MM-DD format into:
  - The 6th and 7th bytes (Month) (start at byte 6 and take 2 bytes)
  - Bytes 0-4 (Year) – start at byte 0 and take 5 bytes (this was recommended by the class – I would normally do substr(1,4).



| mth | 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

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(*) from SPACEXTBL where Date >= '2010-06-04' and Date <= '2017-03-20' group by 1 order by 2 desc

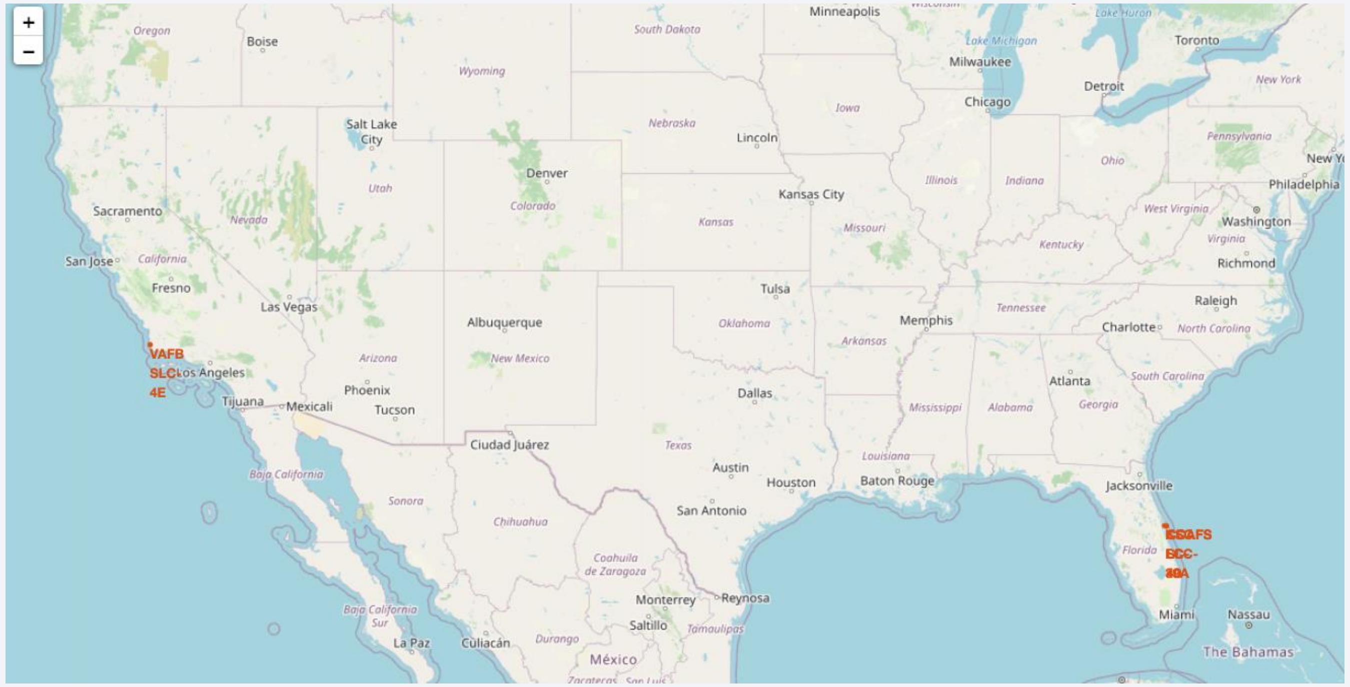
- The only new item here is the use of desc to note that the sort should be in descending order. In this case, it is on the count field, so it shows the largest values at the top

| Landing_Outcome | count(*) |
|---:|---:|
| No attempt | 10 |
| Success (drone ship) | 5 |
| Failure (drone ship) | 5 |
| Success (ground pad) | 3 |
| Controlled (ocean) | 3 |
| Uncontrolled (ocean) | 2 |
| Failure (parachute) | 2 |
| Precluded (drone ship) | 1 |

Section 3

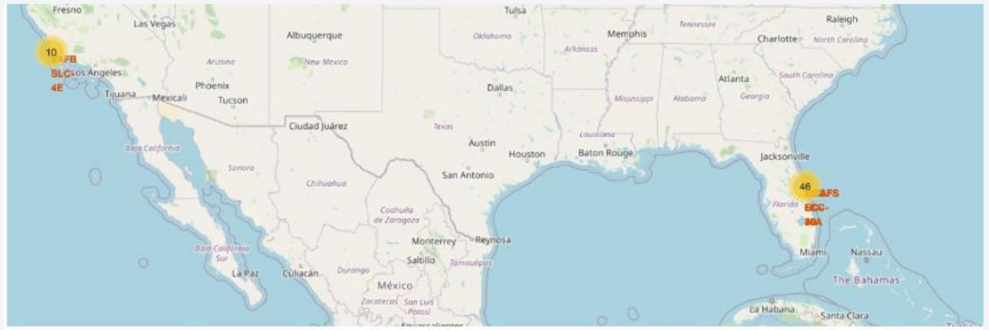
## Launch Sites Proximities Analysis

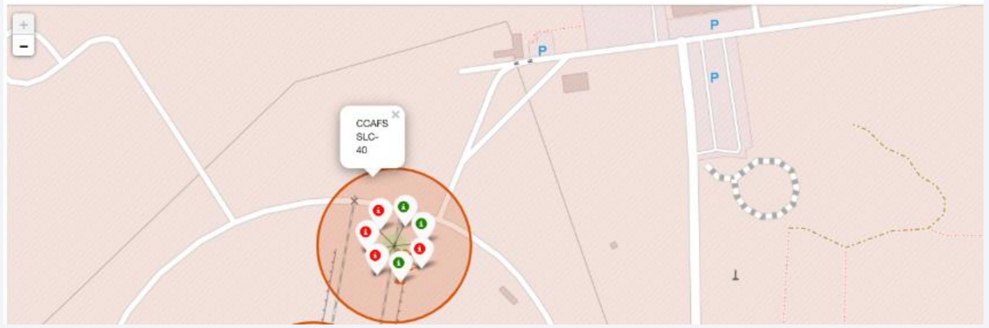
# SpaceX Launch Sites



Markers have been added at the launch sites

# <Folium Map Screenshot 2>



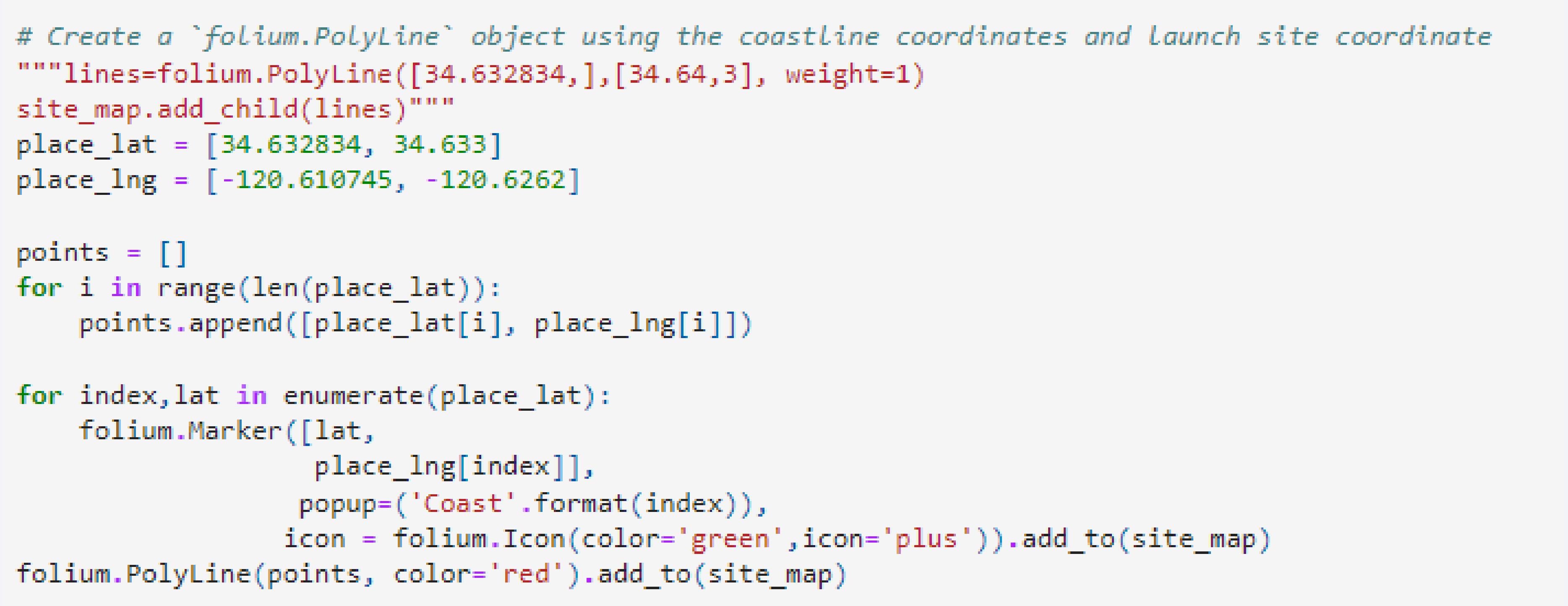


The Yellow groupings are the clusters assigned to each launch site.

When you zoom in and click on a cluster, it will show a circular set of green (success) and red(failure) values for each launch from there

## <Folium Map Screenshot 3>

Sorry, my maps didn't come along with my notebooks when I saved them. But I had a map of a California launch site showing a line of 1.9km to the shore, using this code:

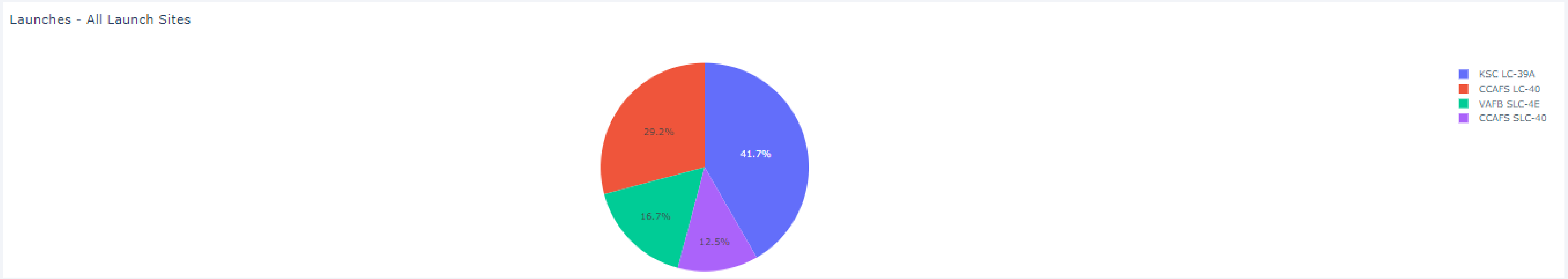


```
# Create a `folium.PolyLine` object using the coastline coordinates and launch site coordinate
"""lines=folium.PolyLine([34.632834,],[34.64,3], weight=1)
site_map.add_child(lines)"""
place_lat = [34.632834, 34.633]
place_lng = [-120.610745, -120.6262]

points = []
for i in range(len(place_lat)):
    points.append([place_lat[i], place_lng[i]])

for index,lat in enumerate(place_lat):
    folium.Marker([lat,
                   place_lng[index]],
                  popup=('Coast'.format(index)),
                 icon = folium.Icon(color='green',icon='plus')).add_to(site_map)
folium.PolyLine(points, color='red').add_to(site_map)
```

Section 4

## Build a Dashboard with Plotly Dash

# Launch Success by Launch Site



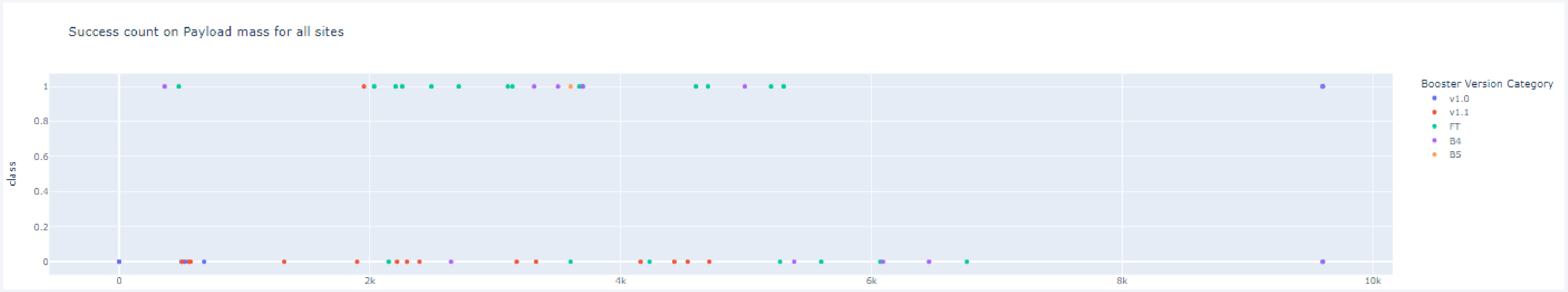
The pie chart shows the share of launches from each site. This is the option that appears when the dropdown box selects All Sites

## <Dashboard Screenshot 2>

Sorry, my maps didn't come along with my notebooks when I saved them.  But I had a map of a California launch site showing a line of 1.9km to the shore, using this code:

- But what happened on the interactive dashboard was that a selection in the dropdown box of any of the individual sites would change the pie chart from a "count by site" into a Success vs Failure pie chart.

# Success Count by Payload Mass for All Sites



The successful launches are at the top of the chart (Class = 1)

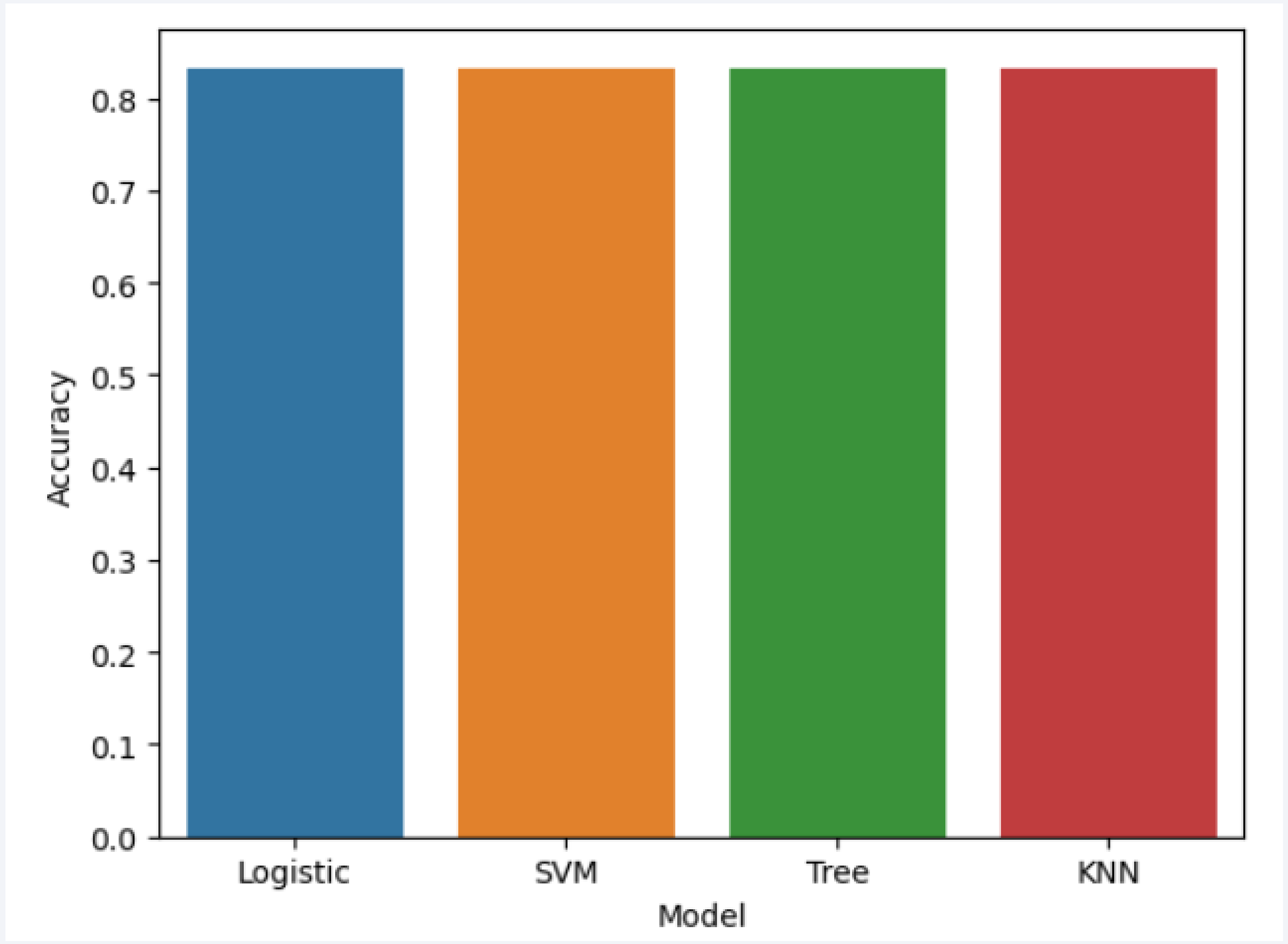
The failed launches are at the bottom of the chart (Class = 0)

The colors signify the Booster Version of the Falcon 9 Rocket

Section 5

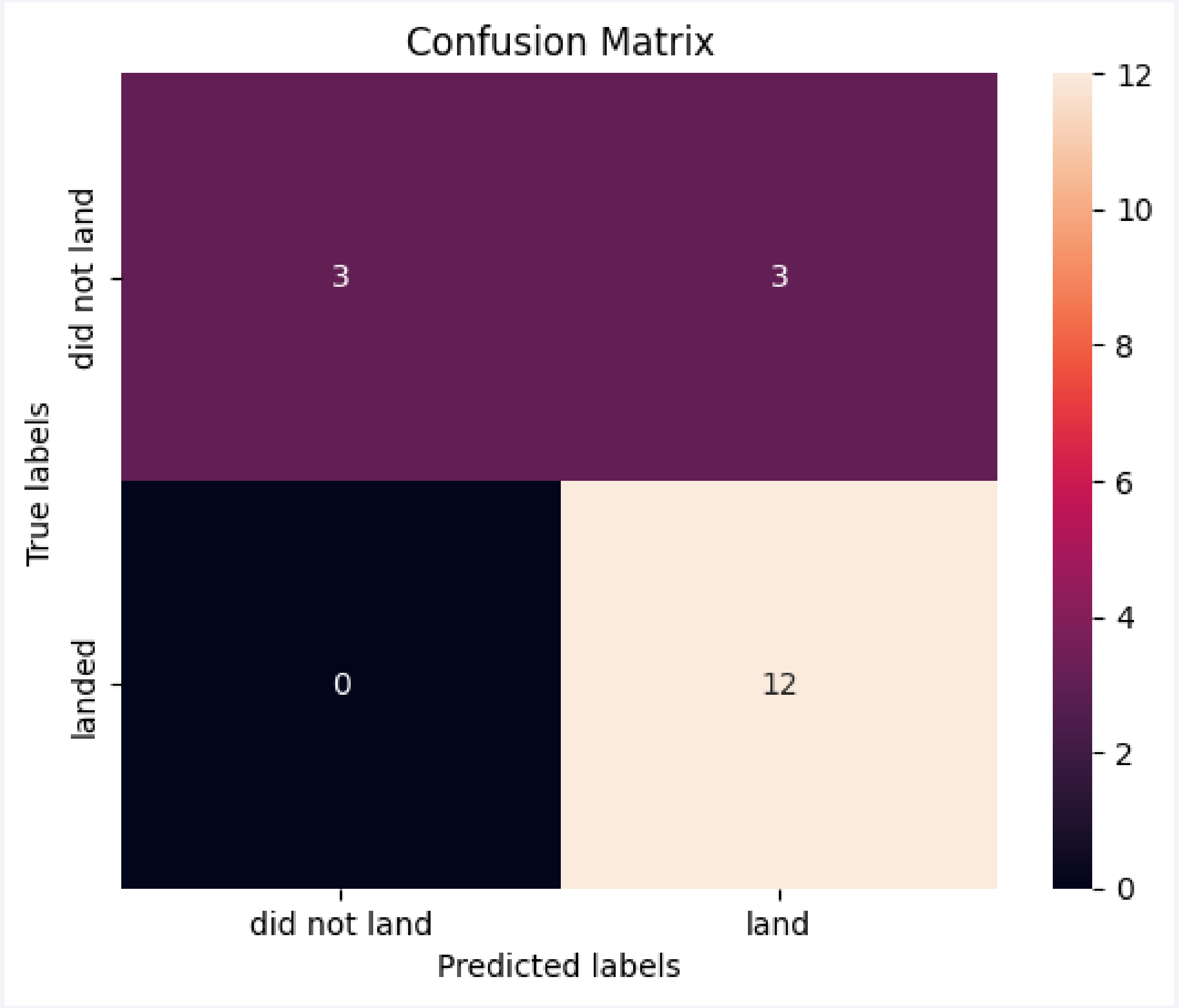
## Predictive Analysis (Classification)

## Classification Accuracy



- As you can see, all four models scored exactly the same (0.833) when modeling the SpaceX data.

## Confusion Matrix



The Confusion Matrix for all 4 models is also identical.

- For all 12 actual landings, all 12 were correctly predicted.
- For the 6 failed landings, the models got 3 correct and got 3 wrong.
- This would be a False Negative problem.

## Conclusions

- The data set of landings is limited with under 100 records.  While that is not a large sample, it was enough to allow our models to be accurate with scores of 0.833
- Despite all models trained on the training data performing equally well on the test data, the Support Vector Machine model was slightly more accurate when rebuilt using the full set of data, achieving a score of 0.877 with only 4 False Positives.
- Through exploratory data analysis, it is clear that Launch Number (or Date) plays a major role in the success or failure.  Early launches rarely succeeded, while later launches are succeeding at a rate of nearly 90%
- For a competitor to enter the market, the early failure of SpaceX landing may be more indicative of early results for a competitor.  So, the pricing of early launches will be closer to $165MM before success becomes regular, leading to the re-use of the Falcon 9 Rocket

## Appendix

- All project elements can be found at:
- https://github.com/nolimpers/IBM-Class-Capstone/tree/main

Thank you!