
# Winning Space Race with Data Science

Produced by: Renzo Macedo
Date: September 2022

# Outline

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

# Executive Summary

- Summary of methodologies
  - Data Collection with Web Scraping
  - Data Collection through API
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Building a Dashboard with Ploty Dash
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

# Introduction

- Project background and context
  - 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.
- Problems you want to find answers
  - If we can determine if the first stage will land successfully, we can determine the cost of a launch

Section 1

# Methodology

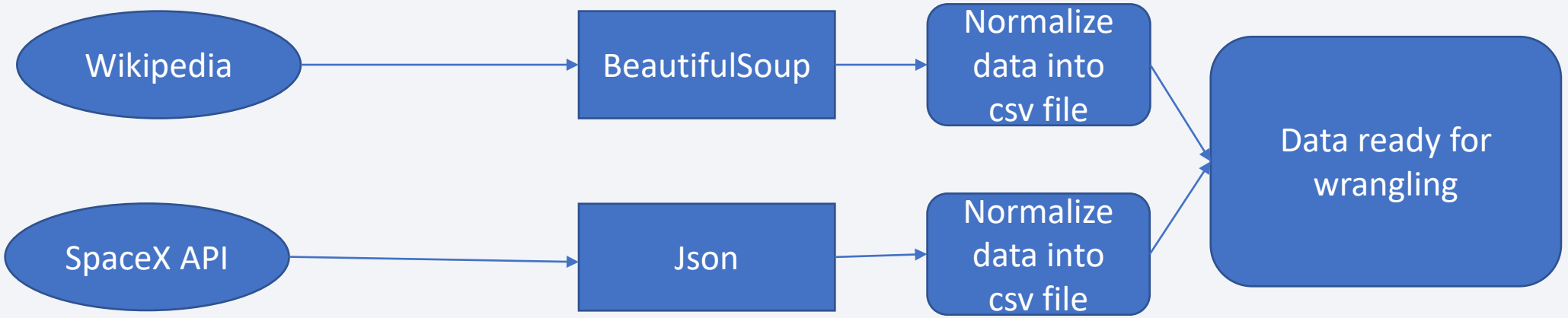
# Methodology

## Executive Summary

- Data collection methodology:
  - Web scraping: Extract a Falcon 9 launch records HTML table from Wikipedia
  - Data collection API: Request to the SpaceX API
- Perform data wrangling
  - Perform exploratory Data Analysis and determine Training Labels, Cleaning data of null values and 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
  - Create column class(succeed or fail) ,Standardize the data, split into training and test data, Find best Hyperparameter for SVM, Classification Trees and Logistic Regression and evaluate accuracy and tunes hyperparameters with GridSearchCV object

# Data Collection

- Extract a Falcon 9 launch records HTML table from Wikipedia and parse the table and convert it into a Pandas data frame with BeautifulSoup
- Data collection API: Request to the SpaceX API using Pandas and Numpy and use json to decode the response content and turn it into Pandas dataframe



# Data Collection – SpaceX API

- Data collection with SpaceX REST calls

https://github.com/renzomacedo/DataScienceCapstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

## 1° Get the data from SpaceX Rest

```
spacex_url="https://api.spacexdata.com/v4/launches/past"

response = requests.get(spacex_url)
response
```

## 2° Convert the json result into a data frame

```
data=pd.json_normalize(response.json())
```

## 3° Clean Data and take subset of our dataframe

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

## 4°Construct our dataset and combine into a dictionary

```
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}
```

## 5° Export to csv file

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

# Data Collection - Scraping

- Web scraping: Extract a Falcon 9 launch records HTML table from Wikipedia

https://github.com/renzomaceo/DataScienceCapstone/blob/main/jupyter-labs-webscraping.ipynb

## 1° Getting response from html web and create coup object

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
response=requests.get(static_url)
response
soup=BeautifulSoup(response.text, "html.parser")
```

## 2° Finding tables

```
html_tables=soup.find_all("table")
html_tables
```

## 3° Getting columns

```
column_names = []

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

## 4° Create a dictionary

```
# Let's initial the launch_dict with each
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'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

## 5° Appending data to dictionary keys

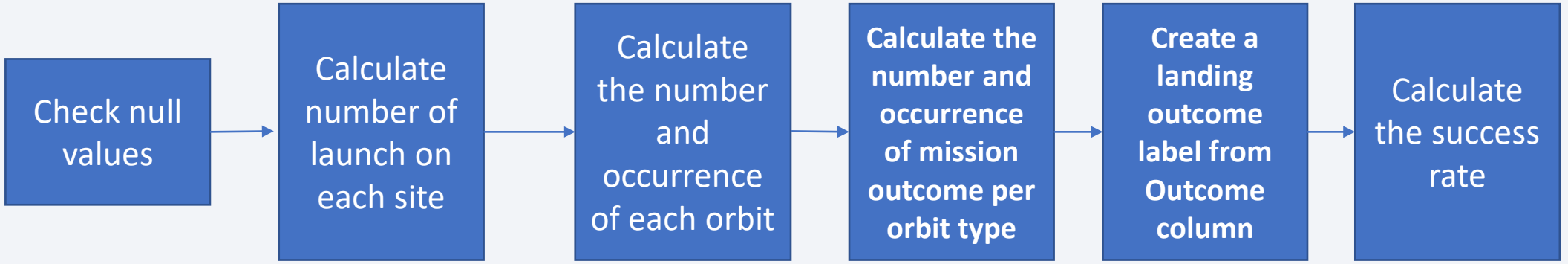
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
```

## 6° Dictionary to dataframe and dataframe to csv

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df.to_csv('spacex_web_scraped.csv', index=False)
```

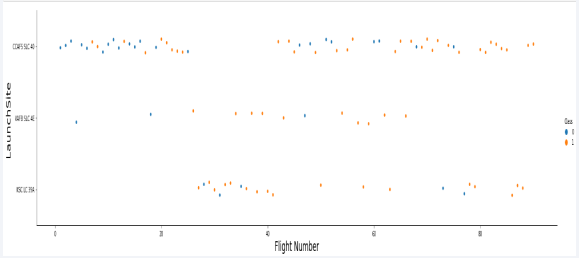
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

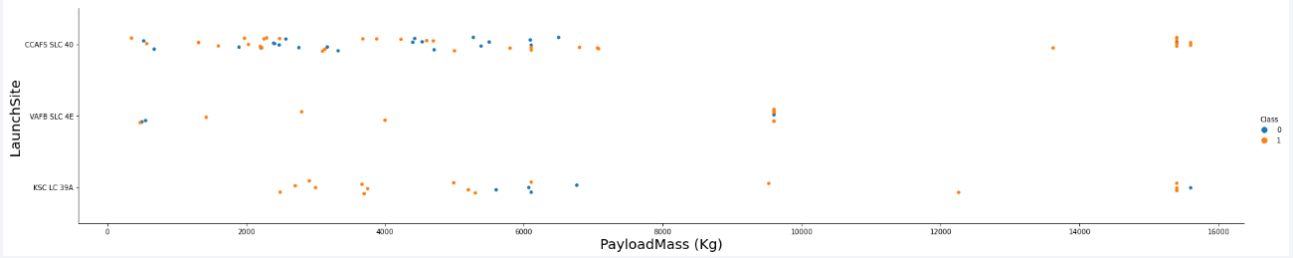
# Data Wrangling

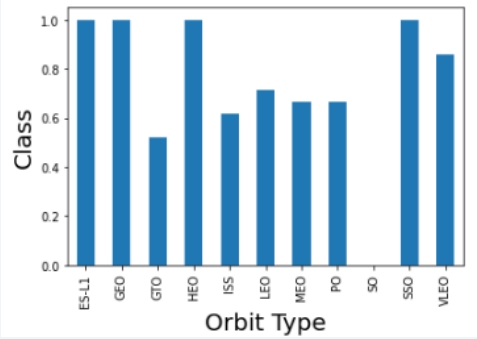


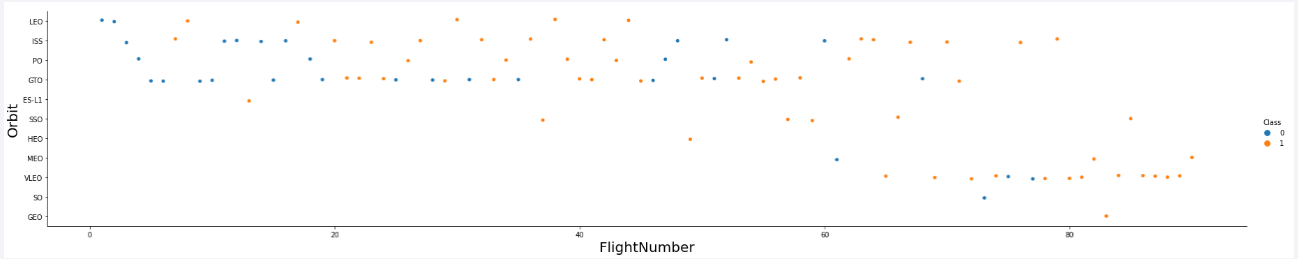

- https://github.com/renzomacedo/DataScienceCapstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

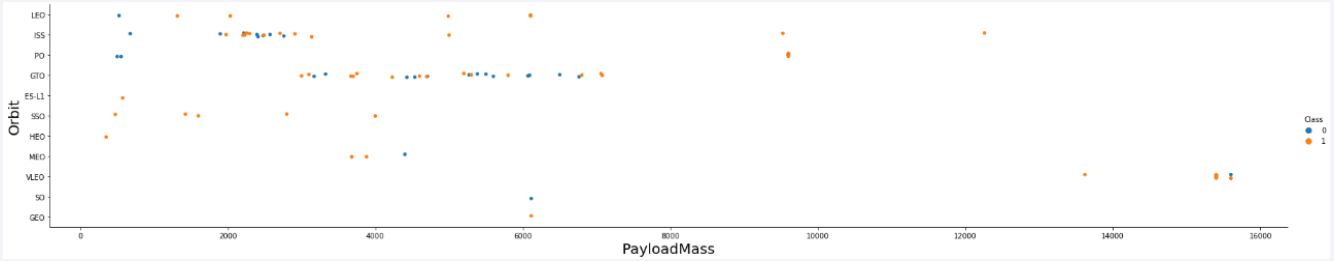
# EDA with Data Visualization

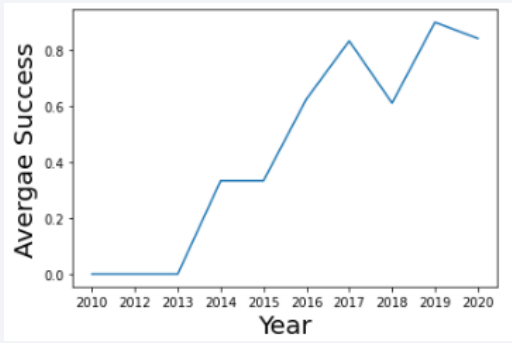


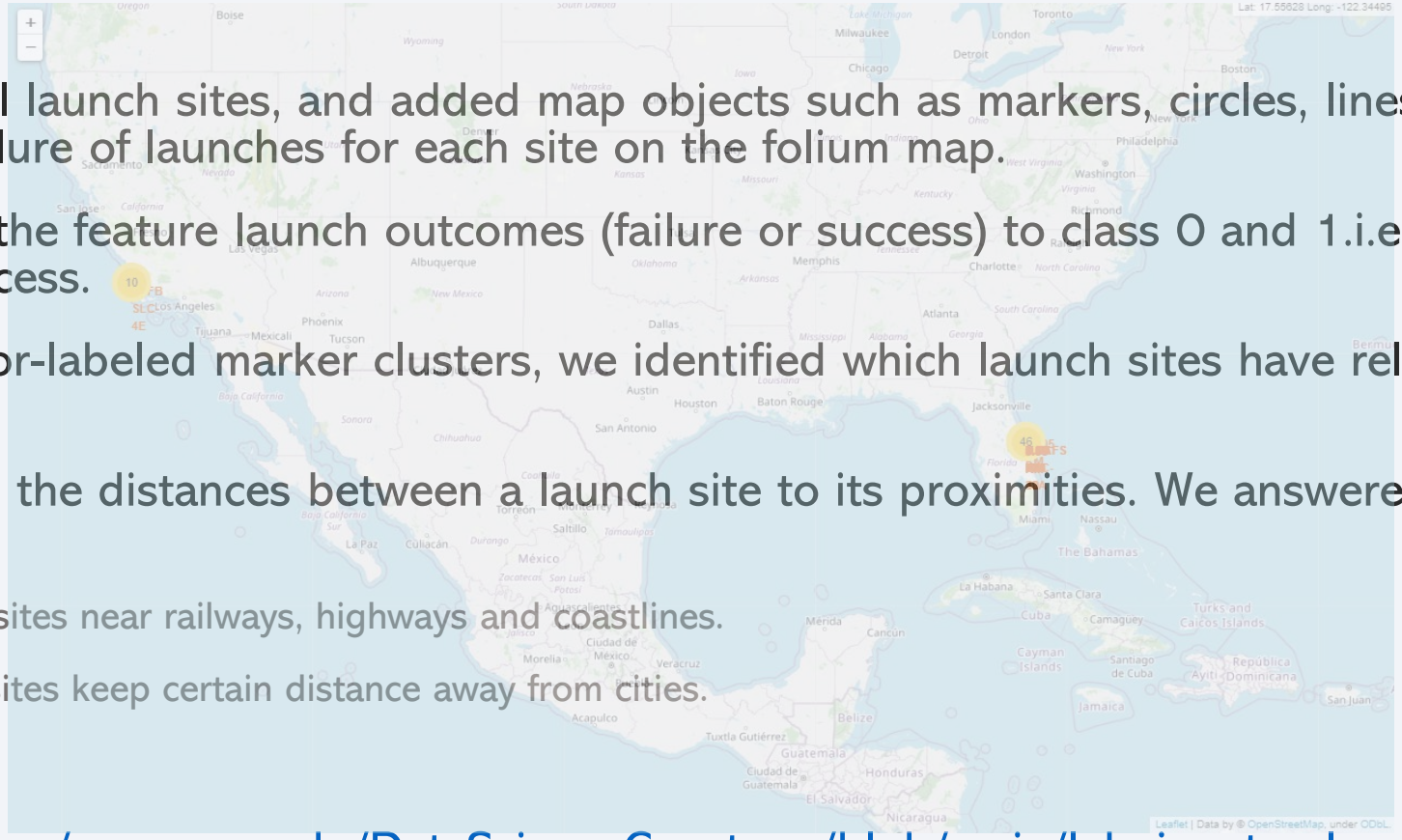

https://github.com/renzomacedo/DataScienceCapstone/blob/main/jupyter-labs-eda-dataviz.ipynb

# EDA with SQL

- SQL queries performed
  - 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.

https://github.com/renzomacedo/DataScienceCapstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

# Build an Interactive Map with Folium

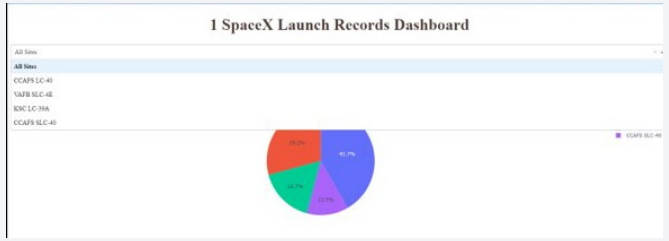


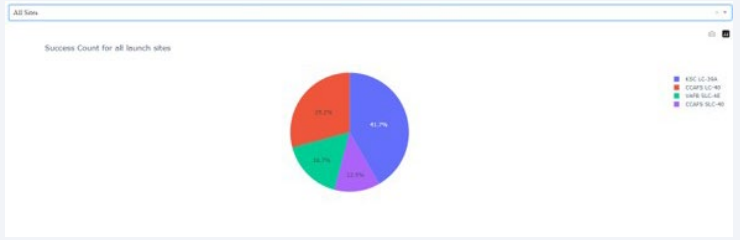
- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

https://github.com/renzomacedo/DataScienceCapstone/blob/main/lab_jupyter_launch_site_location.ipynb

# Build a Dashboard with Plotly Dash

- Built an interactive dashboard with Plotly dash
- Plotted pie charts showing the total launches by a certain sites
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.



https://github.com/renzomacedo/DataScienceCapstone/blob/main/spacex_dash_app.py

# 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.

https://github.com/renzomacedo/DataScienceCapstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

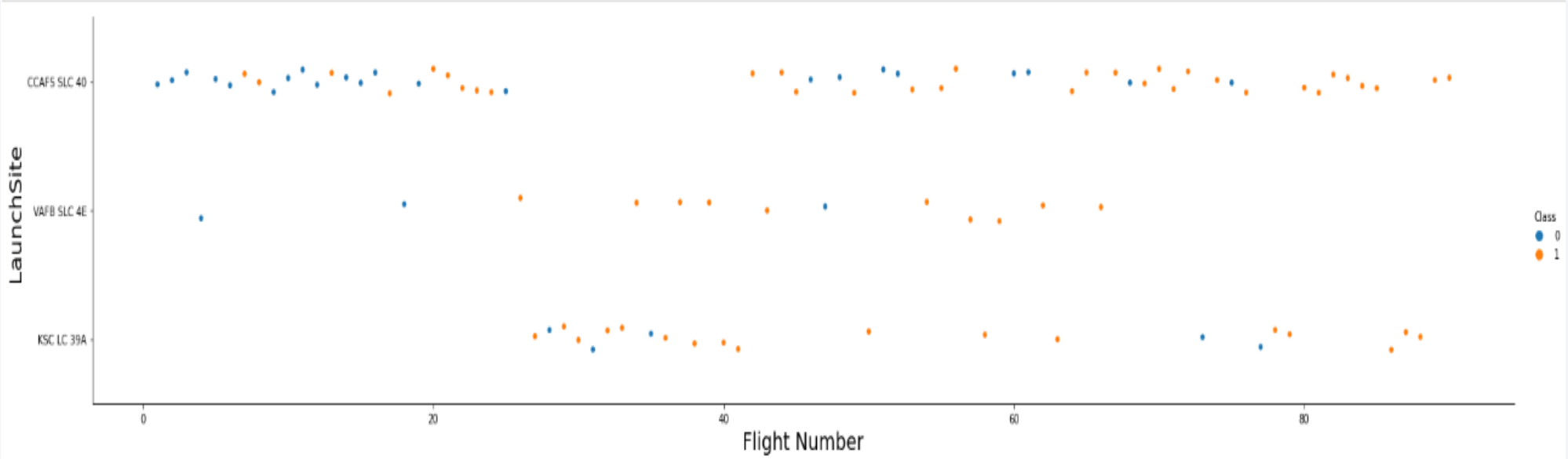
# Results

- The Logistic Regression, SVM and KNN models are the best in prediction accuracy
- Low weighted payloads perform better than heavier payloads
- The success rate of SpaceX launches is directly proportional to the time in years due to the refinement of the launches.
- The launcher place KSC LC 39A had the most successful launches site

Section 2

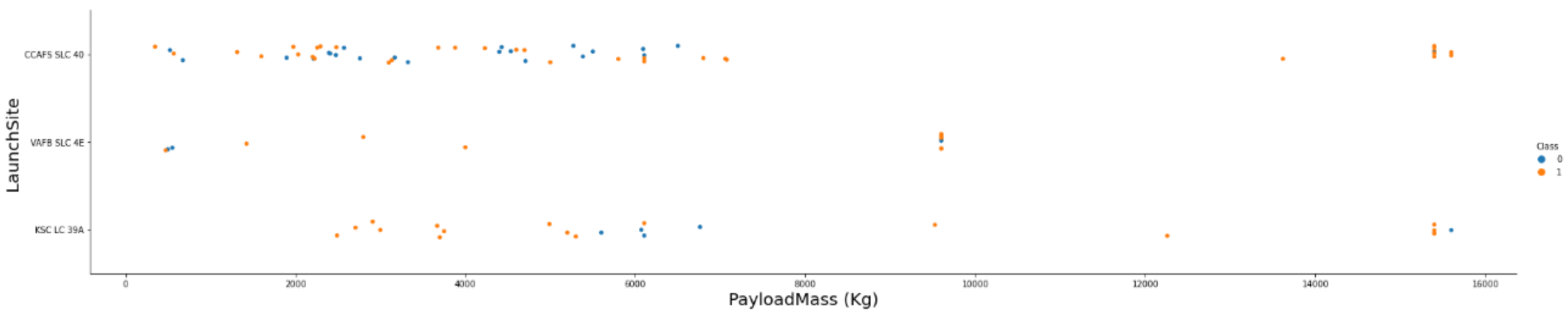
# Insights drawn from EDA

# Flight Number vs. Launch Site




Launches from CCAFS scl 40 are higher than launches form other sites

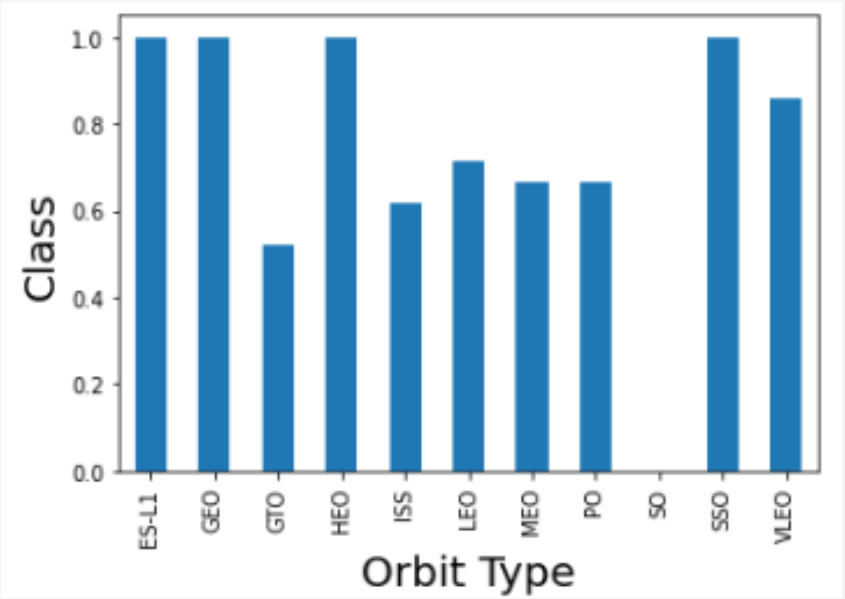
# Payload vs. Launch Site



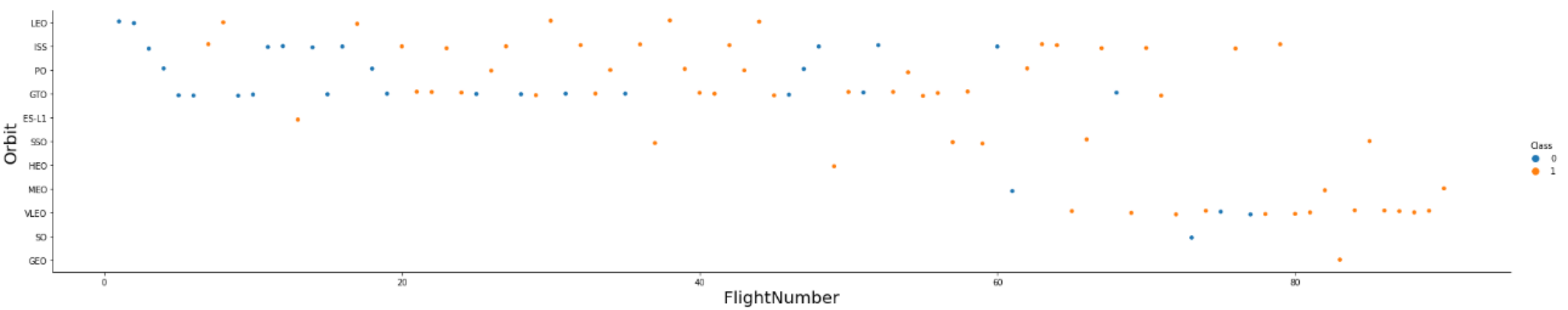

Higher the payload mass for the CCAFS SLC 40 launch site, the higher the success rate of the rocket

# Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



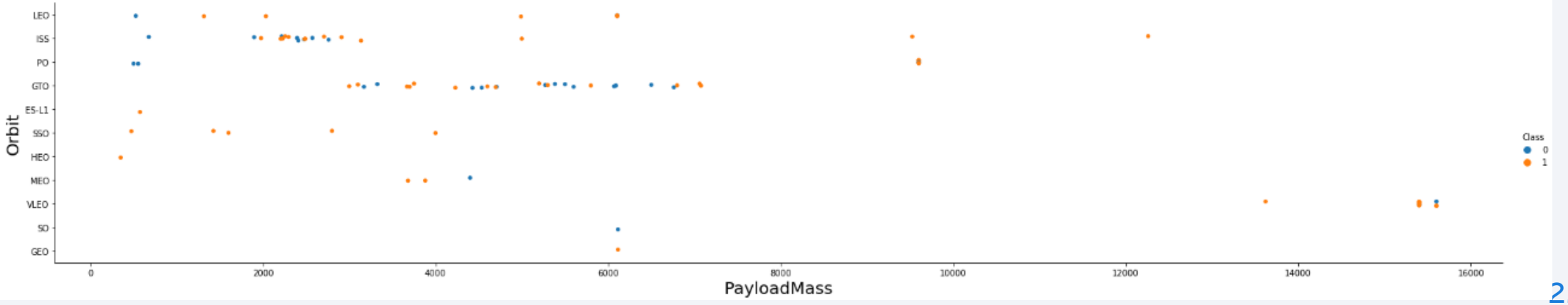
# Flight Number vs. Orbit Type




- We observe that in the LEO orbit, success is related to the number of flights

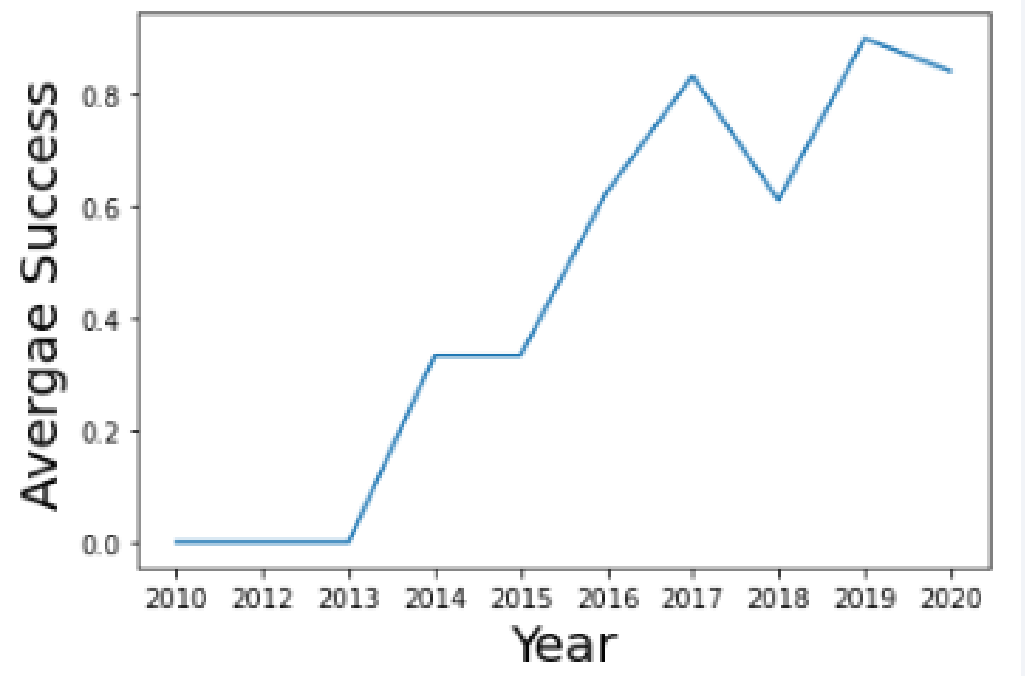
# Payload vs. Orbit Type

- There are strong correlation between ISS and Payload at the range around 2000, as well as between GTO and the range of 4000-8000



# Launch Success Yearly Trend

- Launch success rate has increased significantly since 2013 and stablised since 2019, due to advance in technologic



# All Launch Site Names

We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

```
instruccion="SELECT DISTINCT Launch_Site FROM SPACEXTBL"
cur.execute(instruccion)
datos=cur.fetchall()
print(datos)
```

```
[('CCAFS LC-40',), ('VAFB SLC-4E',), ('KSC LC-39A',), ('CCAFS SLC-40',)]
```

# Launch Site Names Begin with 'CCA'

```
instruccion="SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%'"
cur.execute(instruccion)
datos=cur.fetchmany(5)
print(datos)
```

```
[('04-06-2010', '18:45:00', 'F9 v1.0  B0003', 'CCAFS LC-40', 'Dragon Spacecraft Qualification Unit', 0, 'LEO', 'SpaceX', 'Success', 'Failure (parachute)'), ('08-12-2010', '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)'), ('22-05-2012', '07:44:00', 'F9 v1.0  B0005', 'CCAFS LC-40', 'Dragon demo flight C2', 525, 'LEO (ISS)', 'NASA (COTS)', 'Success', 'No attempt'), ('08-10-2012', '00:35:00', 'F9 v1.0  B0006', 'CCAFS LC-40', 'SpaceX CRS-1', 500, 'LEO (ISS)', 'NASA (CRS)', 'Success', 'No attempt'), ('01-03-2013', '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 above to display 5 records where launch sites begin with `CCA`

# Total Payload Mass

- We sum all column data about Payload mass

```
instruccion="SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL"
cur.execute(instruccion)
datos=cur.fetchall()
print(datos)
```

```
[(619967,)]
```

# Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1 where Booster version is F9 v1.1

```
instruccion="SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version='F9 v1.1'"
cur.execute(instruccion)
datos=cur.fetchall()
print(datos)
```

```
[(2928.4,)]
```

# First Successful Ground Landing Date

- We get the dates of the first successful landing outcome on ground pad :

```
instruccion="SELECT min(Date) FROM SPACEXTBL WHERE [Landing _Outcome]='Success (ground pad)'"
cur.execute(instruccion)
datos=cur.fetchall()
print(datos)
```

```
[('01-05-2017',)]
```

# 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

```
instruccion="SELECT Booster_Version FROM SPACEXTBL WHERE [Landing _Outcome]='Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 "
cur.execute(instruccion)
datos=cur.fetchall()
print(datos)
```

```
[('F9 FT B1022',), ('F9 FT B1026',), ('F9 FT  B1021.2',), ('F9 FT  B1031.2',)]
```

- 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

# Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcome

```
instruccion="SELECT count([Landing _Outcome]) FROM SPACEXTBL where [Landing _Outcome] like 'succes%'"
instruccion2="SELECT count([Landing _Outcome]) FROM SPACEXTBL"
cur.execute(instruccion)
datos=cur.fetchall()
cur.execute(instruccion2)
datos2=cur.fetchall()
print("total :", datos2 )
print("succes :", datos )
print("failiure :", 40 )
```

```
total : [(101,)]
succes : [(61,)]
failiure : 40
```

# Boosters Carried Maximum Payload

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

```
instruccion="SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_=((SELECT max(PAYLOAD_MASS__KG_) from SPACEXTBL))"
cur.execute(instruccion)
datos=cur.fetchall()
print(datos)
```

```
[('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

```
instruccion="SELECT substr(Date,4,2),SUBSTR(DATE,7,4),[Landing _Outcome],Booster_Version,Launch_Site FROM SPACEXTBL WHERE SUBSTR(DATE,7,4)='2015' AND [Landing _Outcome]='Failure (drone ship)'"
cur.execute(instruccion)
datos=cur.fetchall()
print(datos)
```

```
[('01', '2015', 'Failure (drone ship)', 'F9 v1.1 B1012', 'CCAFS LC-40'), ('04', '2015', '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

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

In [19]:
```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

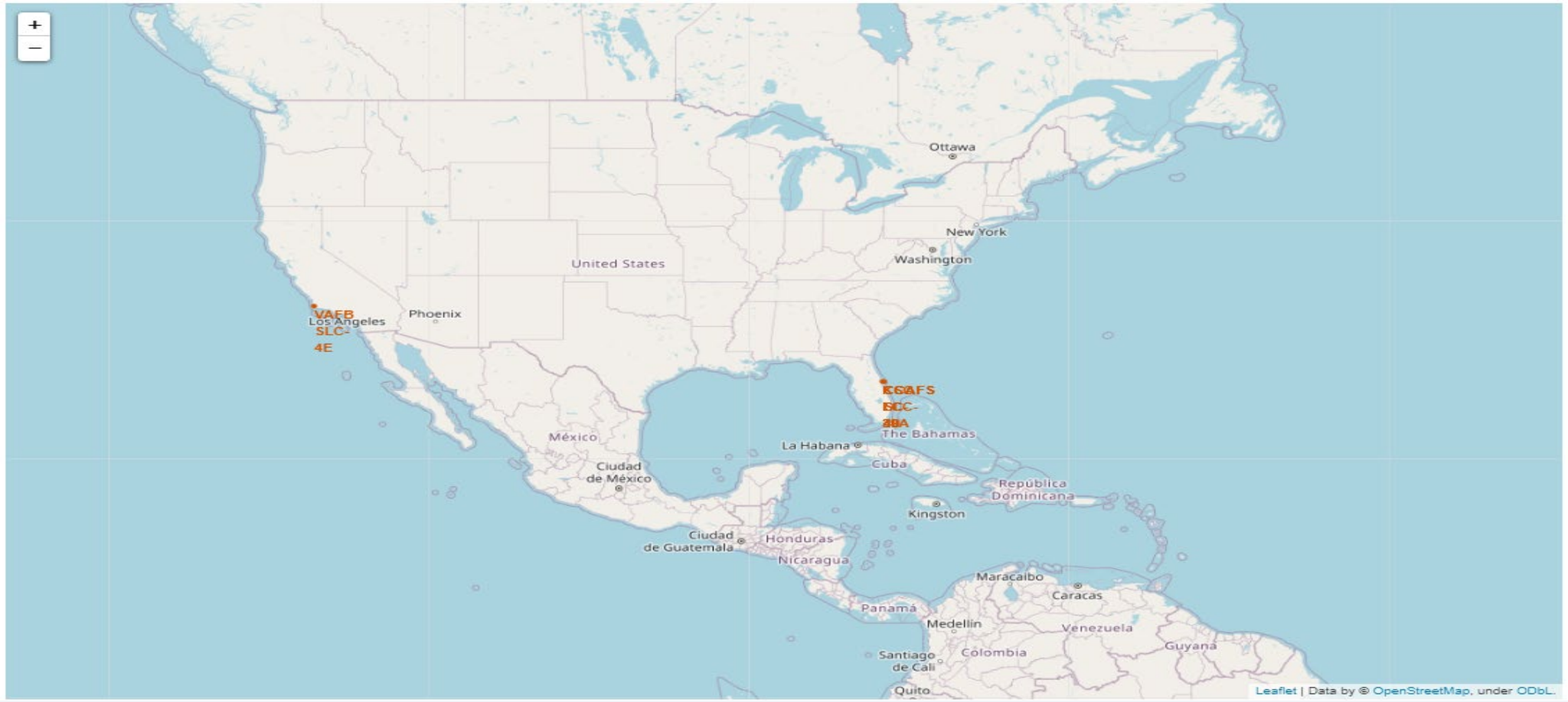
Out[19]:

| | landingoutcome | count |
|---|---|---|
| 0 | No attempt | 10 |
| 1 | Success (drone ship) | 6 |
| 2 | Failure (drone ship) | 5 |
| 3 | Success (ground pad) | 5 |
| 4 | Controlled (ocean) | 3 |
| 5 | Uncontrolled (ocean) | 2 |
| 6 | Precluded (drone ship) | 1 |
| 7 | Failure (parachute) | 1 |

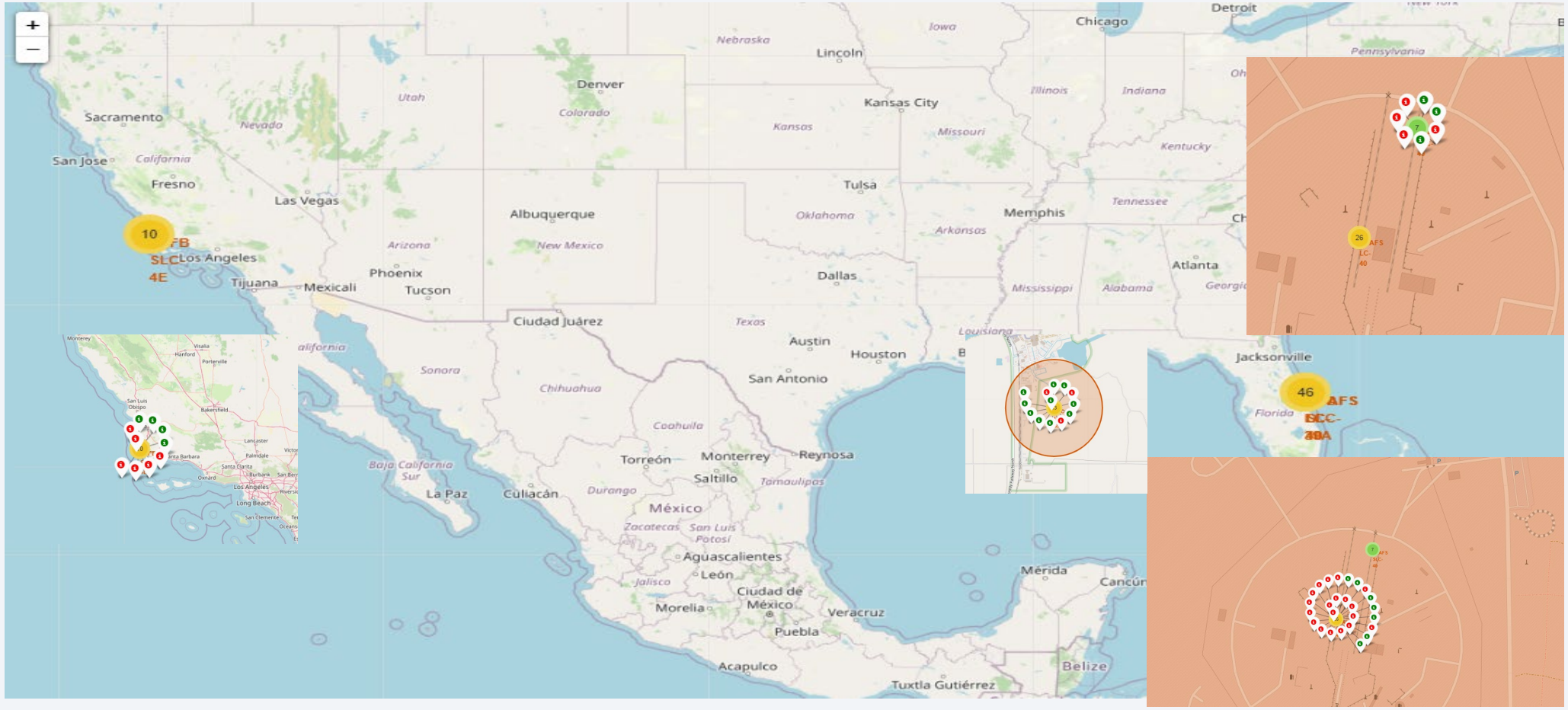

Section 3
Launch Sites
Proximities Analysis

# All launch sites global map markers

All launch site of SpaceX is located in United Stated coast, in Florida and California



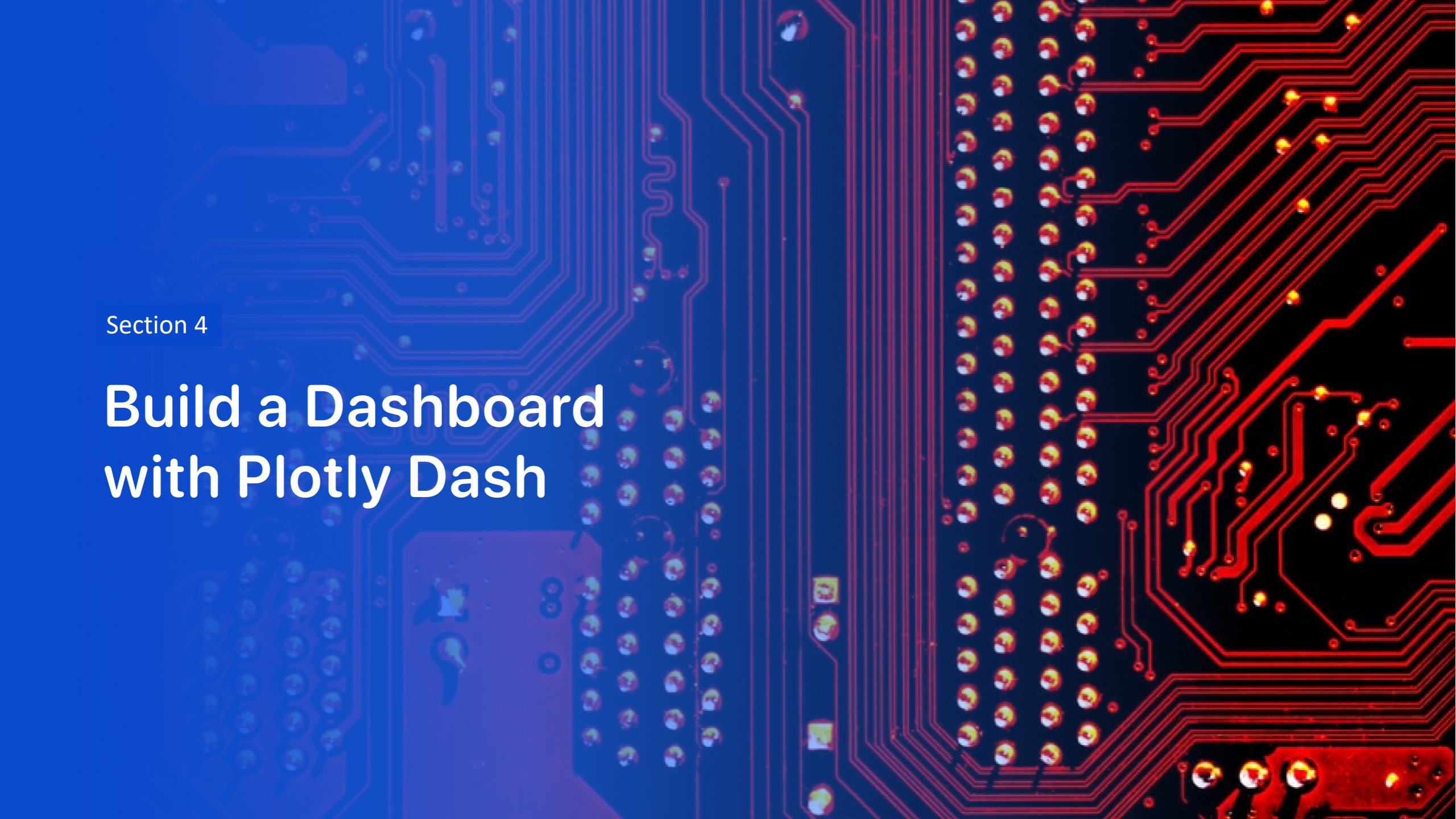
# Success/failed launches for each site on the map



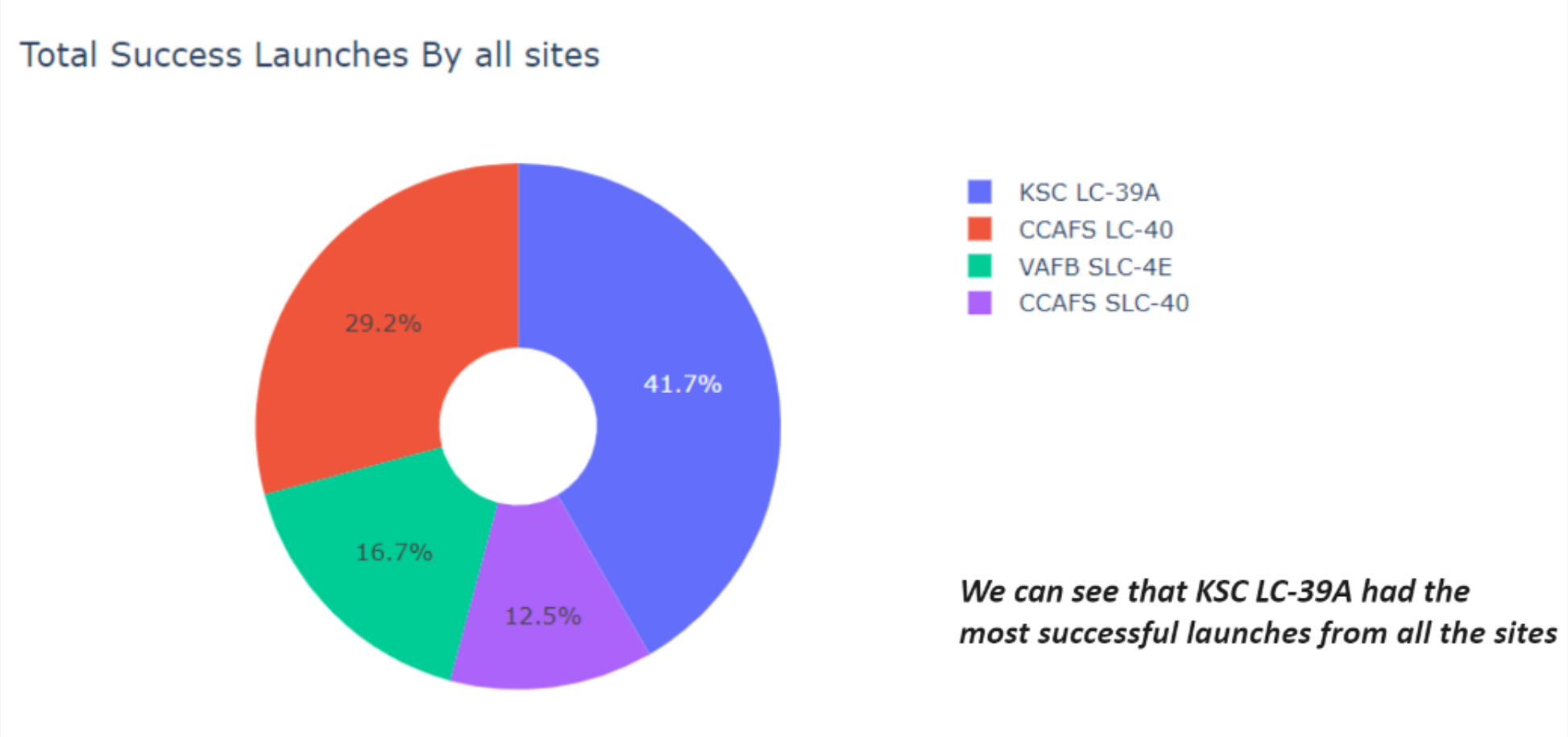

Green marker show successful launches and red mark show failiure

# Distances between a launch site to its proximities

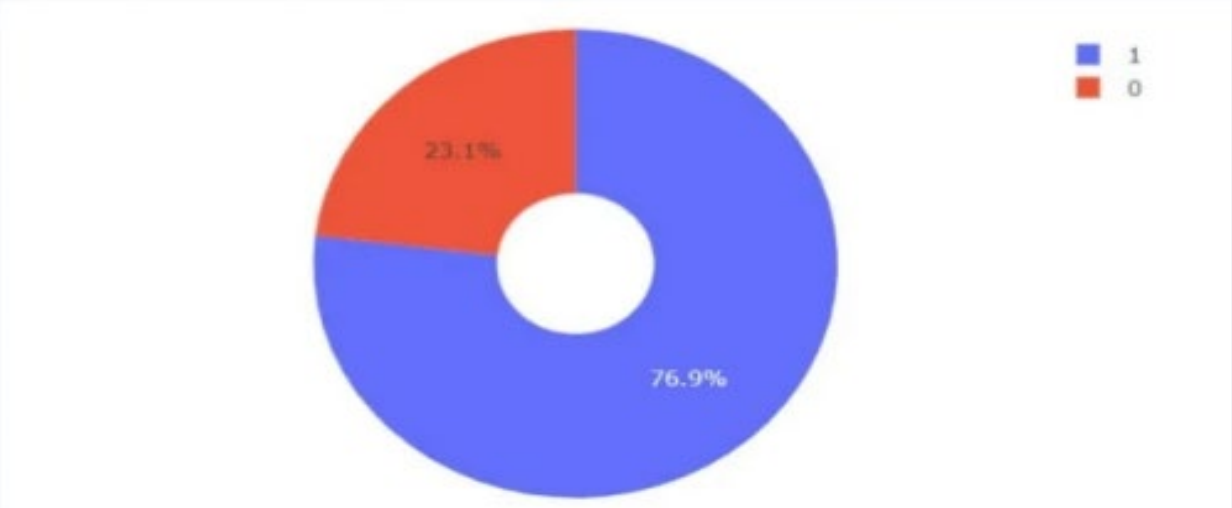

The lines show distances and kilometers to city, railways and coasts


Section 4
Build a Dashboard with Plotly Dash

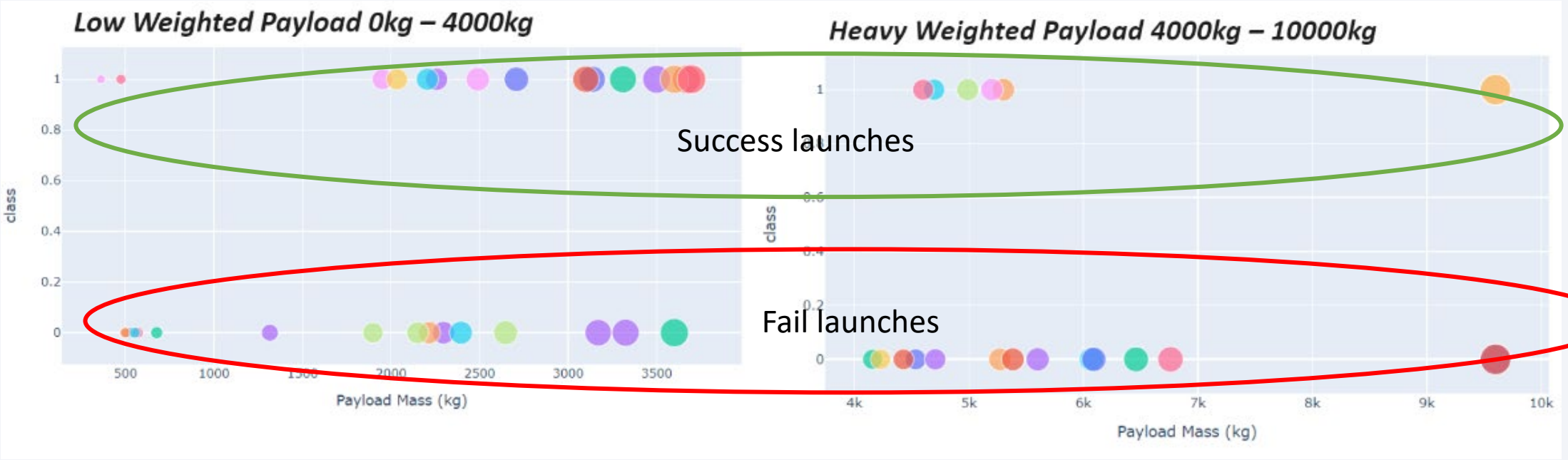
# Total success launches by all sites



# Success rate by site



# Payload vs launch outcome for all sites

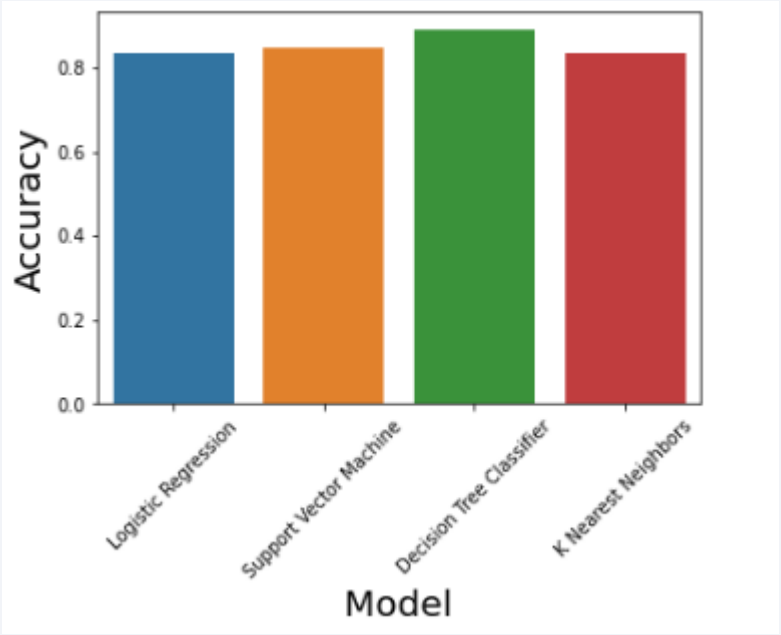


Section 5

# Predictive Analysis (Classification)

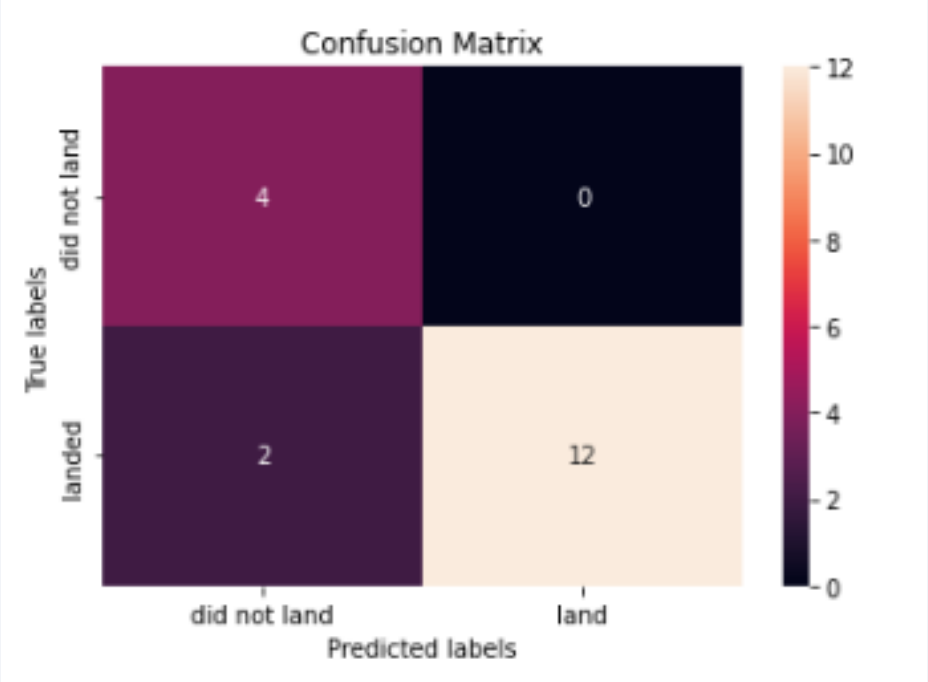
# Classification Accuracy

- All model are similar Accuracy
- The decision tree classifier is the model with the highest classification accuracy



# Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false negative .i.e., unsuccessful landing marked as successful landing by the classifier.



# Conclusions

- The SVM, KNN, and Logistic Regression models are the best in terms of prediction accuracy for this dataset, being the most precise the decision tree classifier.
- Low weighted payloads perform better than the heavier payloads.
- The success rates for SpaceX launches is directly proportional time in years, because they will perfect the launches and the technology.
- KSC LC 39A had the most successful launches from all the sites.
- Orbit GEO,HEO,SSOES L1 has the best Success Rate.

# 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!