Data Science Capstone Project

Victor Augusto
September 17, 2022

Outline

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

Executive Summary

Summary of methodologies

- Data Collection
- Data Wrangling
- Exploratory Data Analysis (EDA)
  - With SQL queries
  - Visualization
- Predictive Analysis

Summary of all results

- EDA results
- Geospacial Analysis
- Dashboard Presentation
- Machine Learning Model

Introduction

SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars, while 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, and this information can be used if an alternate company wants to bid against SpaceX for a rocket launch. Therefore, the main objective is to predict whether Space X's Falcon 9 spacecraft will land successfully. And for that, it is necessary to collect data from reliable sources, clean them, analyze them to extract powerful insights, and prepare for the machine learning model. For this purpose, tools and techniques will be used, which will be presented throughout this report.

Section 1

Methodology

Methodology Summary

Data Collection Methodology

- Get request from Space X API
- Web Scrap with BeautifulSoup library

Data Wrangling

- Use of replace() function to deal with missing values (imputing the mean)
- Method value_counts() to determinate:
  - Number of launches on each site
  - Occurrences of each orbit
  - Number of Landing Outcomes
- Binning 'Outcome' column on 'Class' column for classification

Exploratory Data Analysis (EDA)

- Use of SQL queries to understand and manipulate the data
- Data Visualization with seaborn to extract insights from the data

Interactive Visual Analytics

- Geospacial Analysis with Folium
- Dashboard using Plotly Dash

Predictive analysis using classification models

- Use of scikit-lean to:
  - Preprocess the data with
  - Split the data into train and test sets with train_test_split() method
  - Train different machine learning classification models
  - Use of GridSearchCV to find the best parameters for each model
- Evaluate each model with to find the model that best fits the data using confusion matrix

Data Collection

Data were collected from two sources, from Space X's own API and also from its respective Wikipedia page.
In order to acquire data from Space X's RESP API, a get request was made where the response (in JSON format) was converted to Pandas Dataframe and later stored in a .csv file.
On the Wikipedia site, a Web Scraping of a page table was made using the BeautifulSoup library and the generated Dataframe saved in a .csv file.
In each of the cases, different variables were extracted.

Data Collection – SpaceX API

GET request to Space X REST AP

Decode the response data (JSON) to a Pandas dataframe

Deal with collected data (check Appendix)

Selecting variables to be stored

Storing the cleaned data in a dataset

Filtering the data with only Falcon 9 launches

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

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

```
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
unch_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,
'atitude': Latitude}
```

```
df = pd.DataFrame(launch_dict)
```

GitHub

Data Collection – Web Scraping

GET request from Falcon 9 Launch Wiki URL

Create a BeautifullSoup object from a response text (to help searching for information)

Extract of all variables names from the HTML table

Create a dictionary using table's names

Parsing all launch features and storing it in a Dataframe (check Appendix)

```
response = requests.get(static_url)
soup = BeautifulSoup(response.text, 'html5lib')
```

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

```
launch_dict= dict.fromkeys(column_names)
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
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']=[]
```

GitHub

Data Wrangling

After collecting the data, the following tasks were done to deal with them.

- Using the method value_counts() it was verified that the status of the missions had several ways of informing that there was or was not successful.
- As a result, a feature engineering was carried out to reduce them and facilitate the analysis.
- 0 was rated for all outcomes that failed and 1 for those that succeeded.
- Thus creating a new column - 'Class'
- The new dataset has been saved and stored in a .csv file.

GitHub

EDA with Data Visualization

Scatter, bar and line charts were used. Scatter charts are good for visualizing if there is a relationship between the variables of interest - correlation between them; bar charts facilitates the comparison of variables analyzed; line charts are useful for visualizing trends, which helps predict possible outcomes.

Scatter Chart:

- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Flight Number vs. Orbit Type
- Payload vs. Orbit Type

Bar Chart:

- Orbit Type vs. Success Rate

Line Chart:

- Year vs. Success Rate

GitHub

EDA with SQL

To better understand the data, the following queries were performed:

- Display the names of the unique launch sites in the space mission;
- Display 5 records where launch sites begin with the string 'CCA';
- Display the total payload mass carried by boosters launched by NASA (CRS);
- Display average payload mass carried by booster version F9 v1.1;
- List the date when the first succesful landing outcome in ground pad was acheived
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000;
- List the total number of successful and failure mission outcomes;
- List the names of the booster_versions which have carried the maximum payload mass;
- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015;
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

GitHub

Interactive Map with Folium

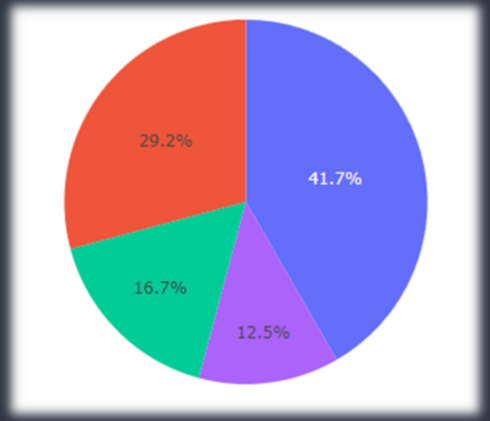
The following steps were followed for geospatial analysis:

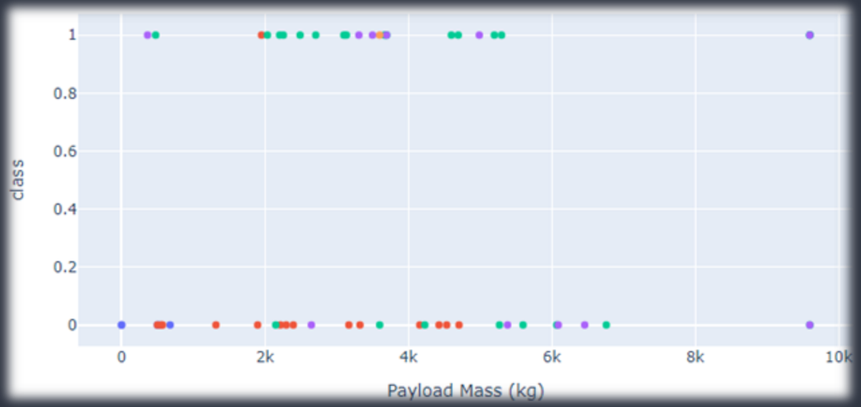
- Create a folium Map object – folium.Map
- Add all launch sites coordenates using its latitude and longitude
- Add circle and a marker to highlight each location site and a pop up text label with its name – folium.Circle and folim.Marker
- Create clusters with close launch sites – MakeCluster
- Encode Outcome column to 1 and 0 – green to success and red for failure
- Add a marker to clearly show sucess and failure with a click
- Add MousePosition to get coordenates fro a mouse over a pont on the map
- Then calculating distances between launch sites to hightway, railway and city

GitHub

Dashboard with Plotly Dash

The interactive dashboard consists of a pie chart and a scatter chart.
The pie chart provides an overview of the success of each release location and can be extracted using the dropdown menu - its construction can be verified on GitHub.
The scatterplot shows the relationship between payload mass and mission success or failure, which can be filtered by weight range via a slider - its construction can be verified on GitHub.





GitHub

Predictive Analysis (Classification)

Normalize the data with StandardScaler

Split the data into train and test sets with train_test_split

Predetermine some parameters for the models

Select classification algorithms (Decision Tree, KNN, SVM and Logistic Regression)

Create a GridSearchCV objet

Fit the models along with their specific parametes in the GridSerachCV

Use the training set to train the model

Use the testing set to evaluate every algorithm

Get the best score along with the accuray of the models

Plot confusion matrix to each model and analyse it's results

Determine which models best fit the data.

GitHub

Results

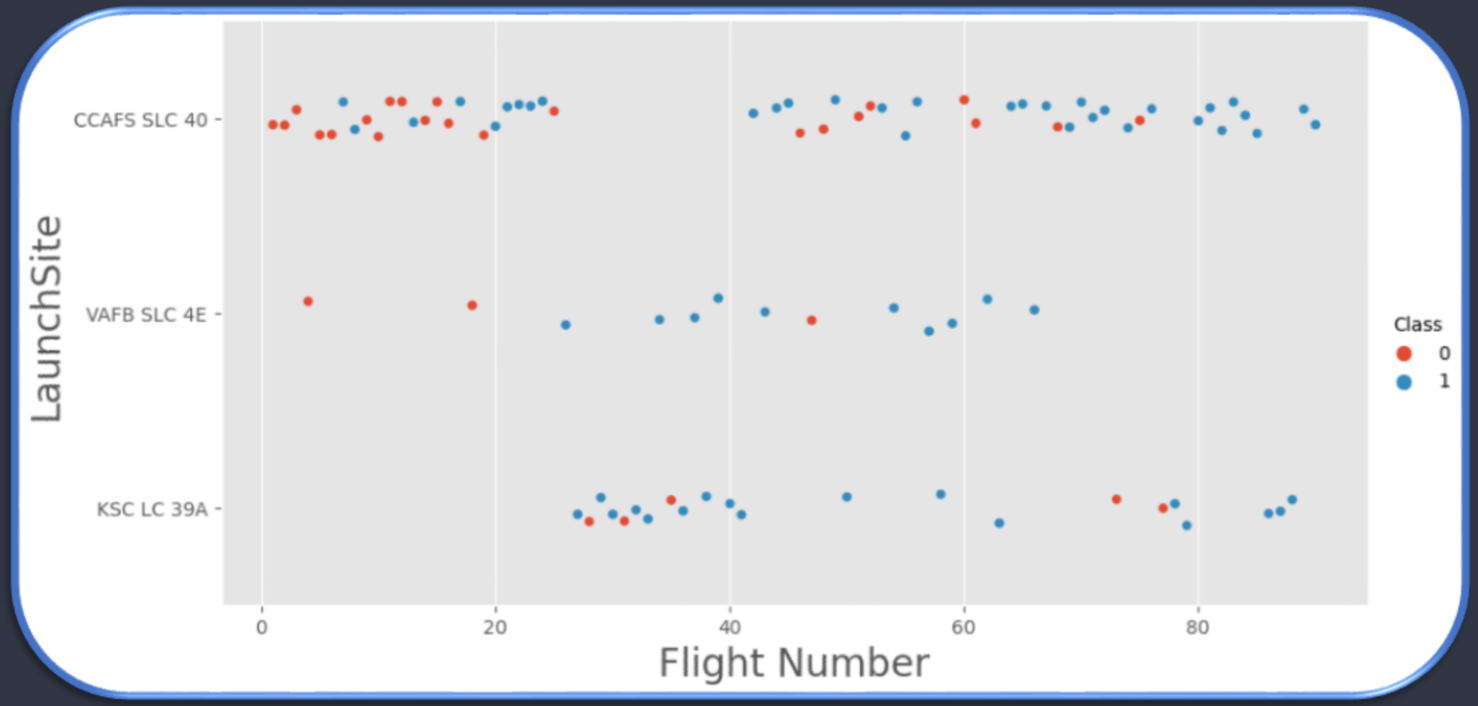
- Exploratory Data Analysis
- Interactive Analytics
- Predictive Analysis

Section 2

Insights drawn from EDA

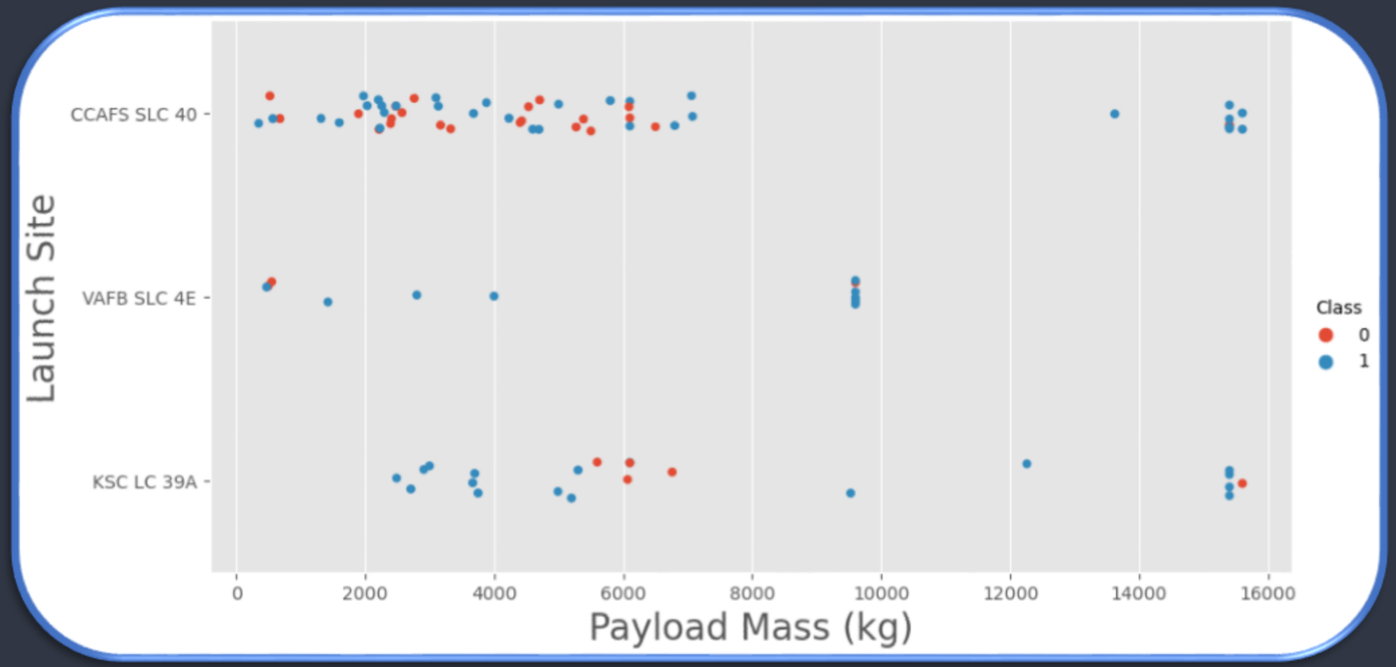
Flight Number vs. Launch Site

- Most of the first launch attempts failed, and almost all of them were from CCAFS SLC40;
- Apparently the first launch sites was where the project started and, as there was no experience, it is expected that these have failed;
- The releases on KSC LC 39A had a higher success rate in their first attempts, which can be assumed to have been lessons from the failures on the other sites;
- So much so that the latest releases (from the 80's onwards) were only successful.



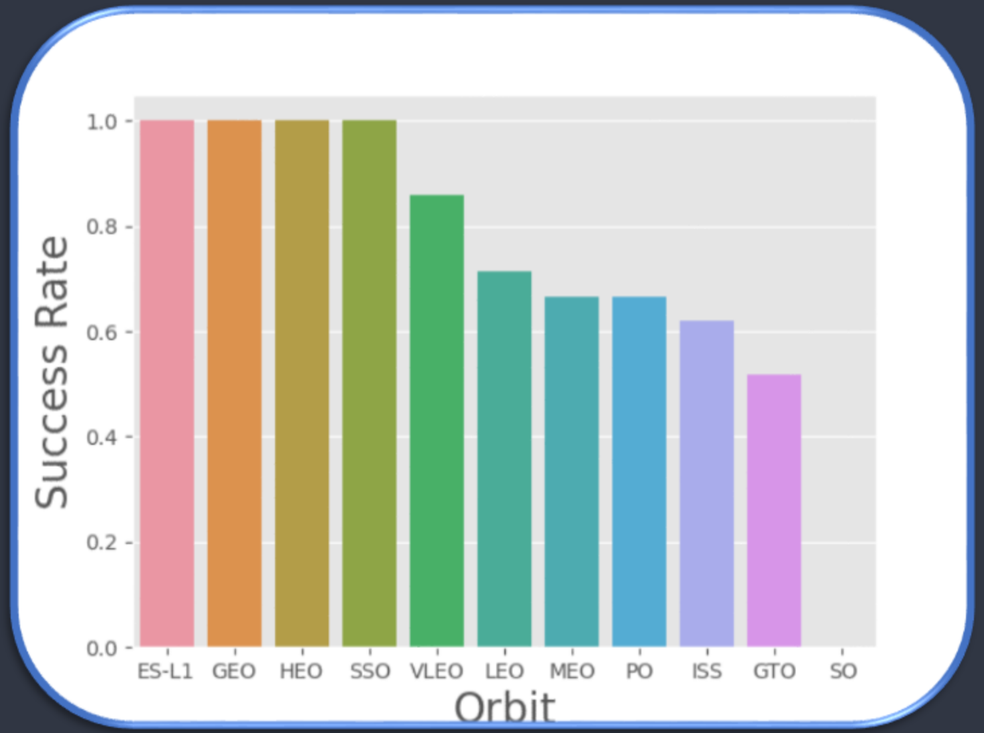
Payload vs. Launch Site

- Apparently, with payload above 8 tons, the greater the chances of success;
- Also, no correlation can be assumed between payload mass and launch site due to the fact that the variation is doesn't seem to have a observable variation.



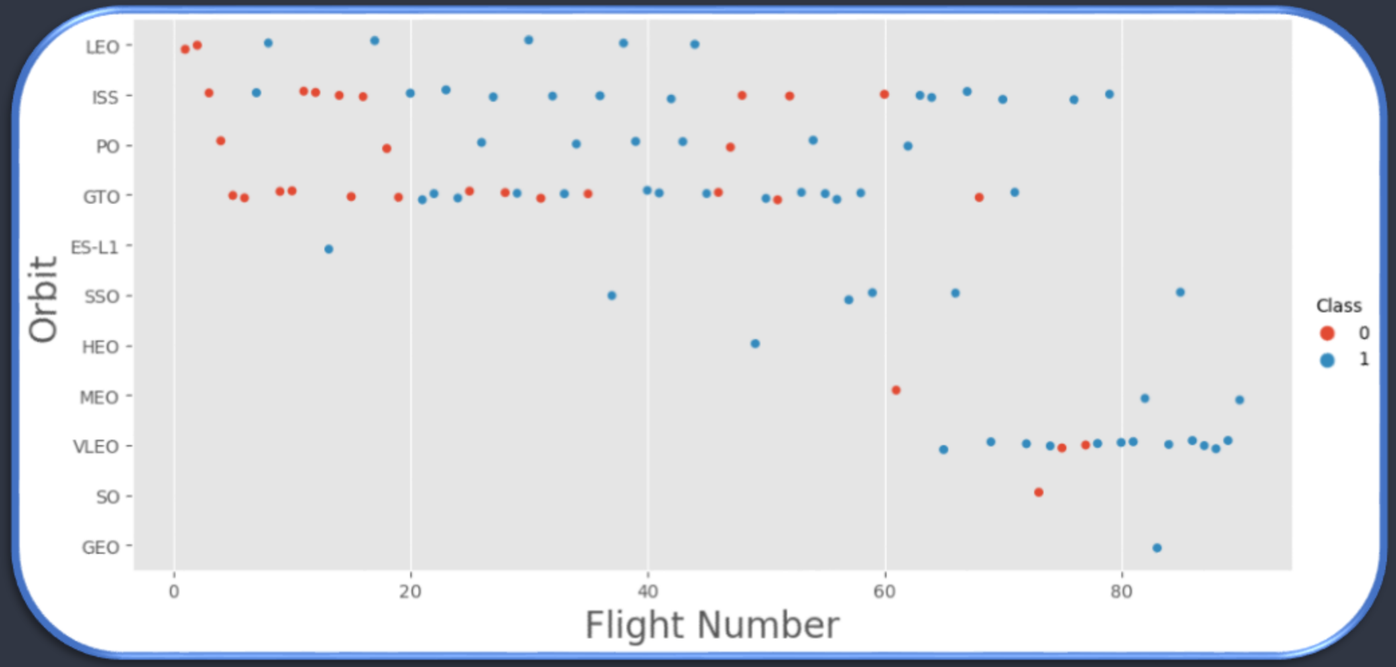
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO and SSO orbits have 100% of success rate whereas SO orbit have 0%;
- The others have a success rate between 40% and 80%



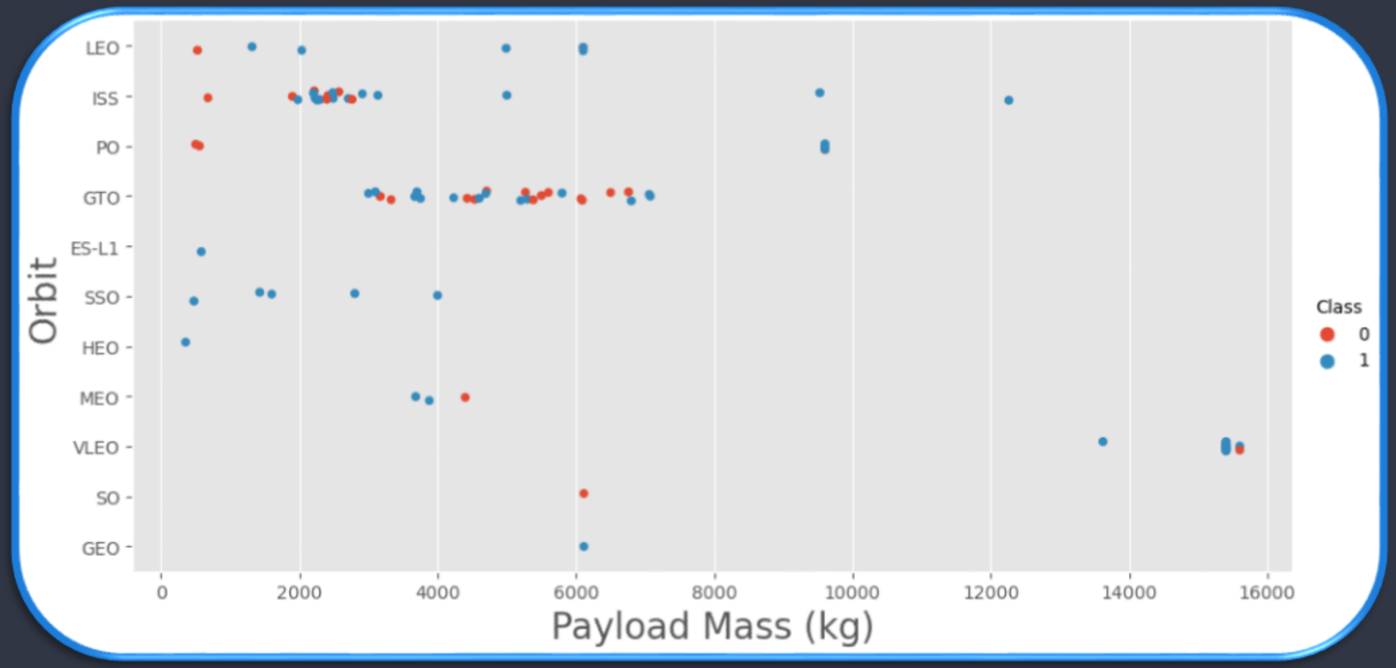
Flight Number vs. Orbit Type

- HEO and GEO orbit have 100% success rate due to the fact that there was only one launch for these orbits;



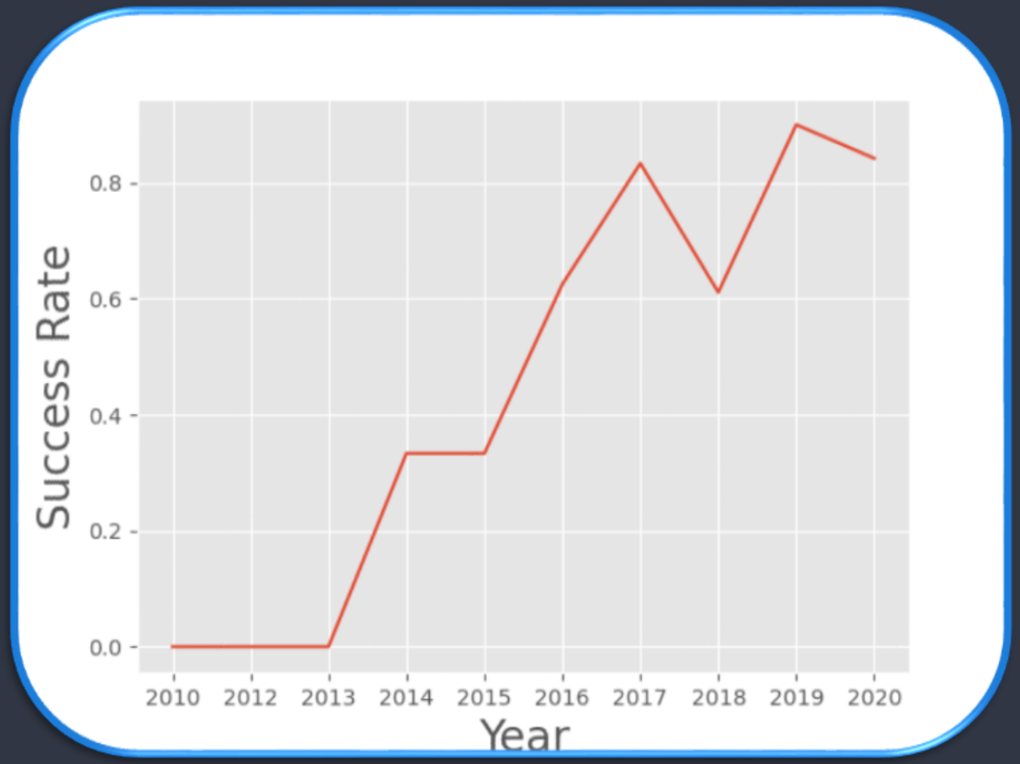
Payload vs. Orbit Type

- Most spacecraft were launched with payload between 2 and 6 tons;
- ISS and GTO were the orbits from which the most spacecraft were launched.



Launch Success Yearly Trend

- Until 2013 no launch was successful;
- From 2013 until 2017 the success rate only increased or remained stable;
- 2018 was not a good year, there is a drop of approximately 20% in the success rate.



All Launch Site Names

Find the names of the unique launch sites

```
%sql select distinct(Launch_Site) from SPACEXTBL;
```

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

This query returns all unique values in the Launch Site column, via the DISTINCT command.

Launch Site Names Begin with 'CCA'

Find 5 records where launch sites begin with `CCA`

```
%sql select * from SPACEXTBL where Launch_Site like "CCA%" limit 5;
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
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

This query returns all records that have 'CCA + something' in column Launch_Site by using %.

Total Payload Mass

Calculate the total payload carried by boosters from NASA

```
%sql select sum(PAYLOAD_MASS__KG_) as TOTAL_PAYLOAD_MASS from SPACEXTBL where Customer = "NASA (CRS)";
```

TOTAL_PAYLOAD_MASS
45596

This query sums all 'NASA (CRS)' customer mass values, adds in TOTAL_PAYLOAD_MASS using as.

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_) as AVERAGE_PAYLOAD_MASS from SPACEXTBL where Booster_Version = "F9 v1.1";
```

AVERAGE_PAYLOAD_MASS
2928.4

This query calculates the mass average payload of Booster_Version F9 v1.1 and name it AVERAGE_PAYLOAD_MASS.

First Successful Ground Landing Date

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

```
%sql select min(Date) from SPACEXTBL where "Landing _Outcome" like 'Success (ground pad)';
```

min(Date)
01-05-2017

This query returns the first data that successfully grounded in pad, and for that keyword min() was used.

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 "Booster_Version" from SPACEXTBL where ("Landing _Outcome"='Success (drone ship)') AND ("PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000);
```

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

This query returns all successful drone shipped missions where the spacecraft was between 4000 kg and 8000 kg using the command BETWEEN.

Total Number of Successful and Failure Mission Outcomes

Calculate the total number of successful and failure mission outcomes

```
%sql select "Mission_Outcome", count(Mission_Outcome) as COUNT from SPACEXTBL group by "Mission_Outcome";
```

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

This query counts every unique value in Mission_Outcome column and returns it as COUNT column.

Boosters Carried Maximum Payload

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

```
%sql select "Booster_Version", "PAYLOAD_MASS__KG_" from SPACEXTBL where "PAYLOAD_MASS__KG_" = (select max("PAYLOAD_MASS__KG_") from SPACEXTBL);
```

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

The query between brackets is used to get the maximum value in PAYLOAD_MASS_KG_ column – that is a subquery. The main query return all Booster_Version and PAYLOAD_MASS_KG_ column in which the payload is equal to that value found in the subquery.

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, 4, 2) as MONTH, "Landing _Outcome", "Booster_Version", "Launch_Site" from SPACEXTBL where (substr(Date,7,4)='2015') AND ("Landing _Outcome" like "%Failure%");
```

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

Substr() was used first to get the month and in the subquery to get the year. This query shows that in January and April of 2015 had failures.

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(*) as N_SUCCESS from SPACEXTBL
where ("Landing _Outcome" like "%Success%") AND (Date between "04-06-2010" and "20-03-2017")
group by "Landing _Outcome"
order by N_SUCCESS DESC;
```

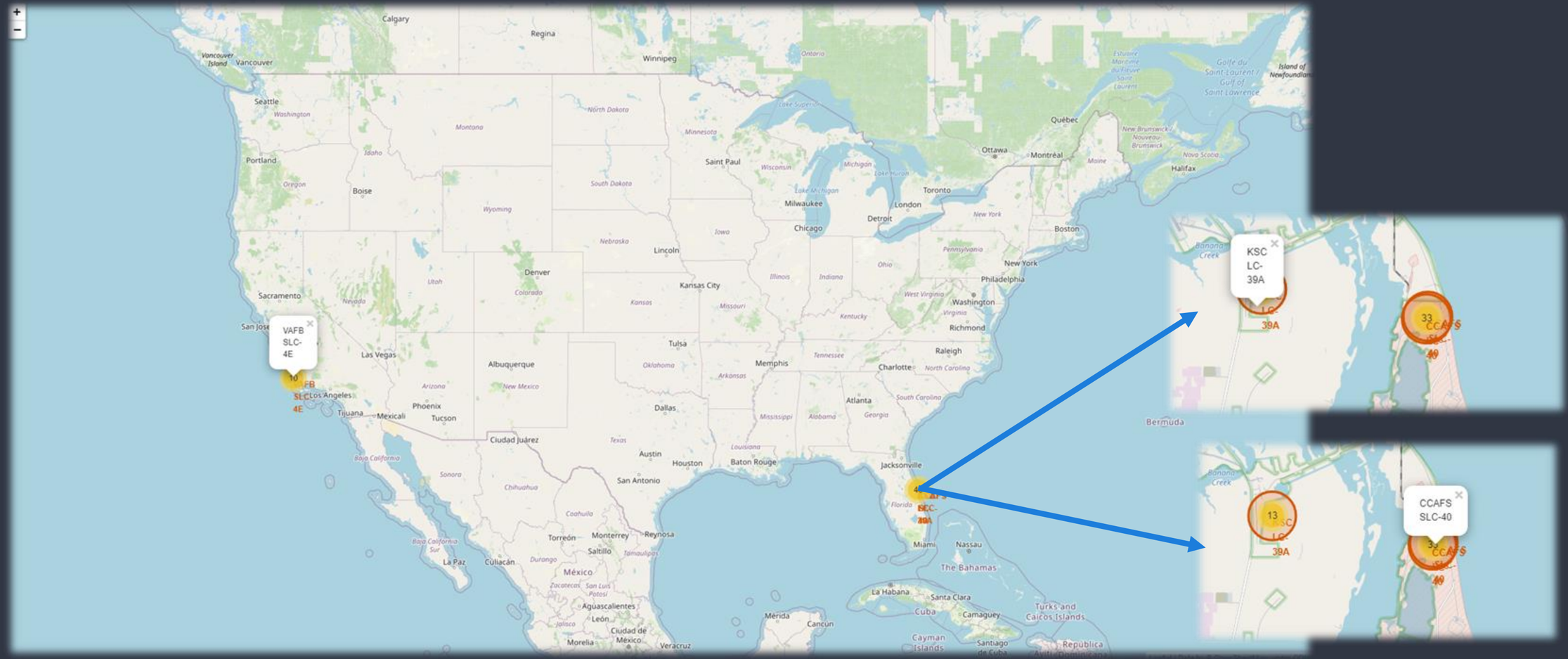
Landing _Outcome	N_SUCCESS
Success	20
Success (drone ship)	8
Success (ground pad)	6

This query contains two conditions – Success in Landing_Outcome (was used % in the beginning and end of Success to find all record with this word in it; and for the date range, the BETWEEN command was used. Group by is to group all unique values in Landing column and Order by command followed by DESC is to sort number of success in descendent order.

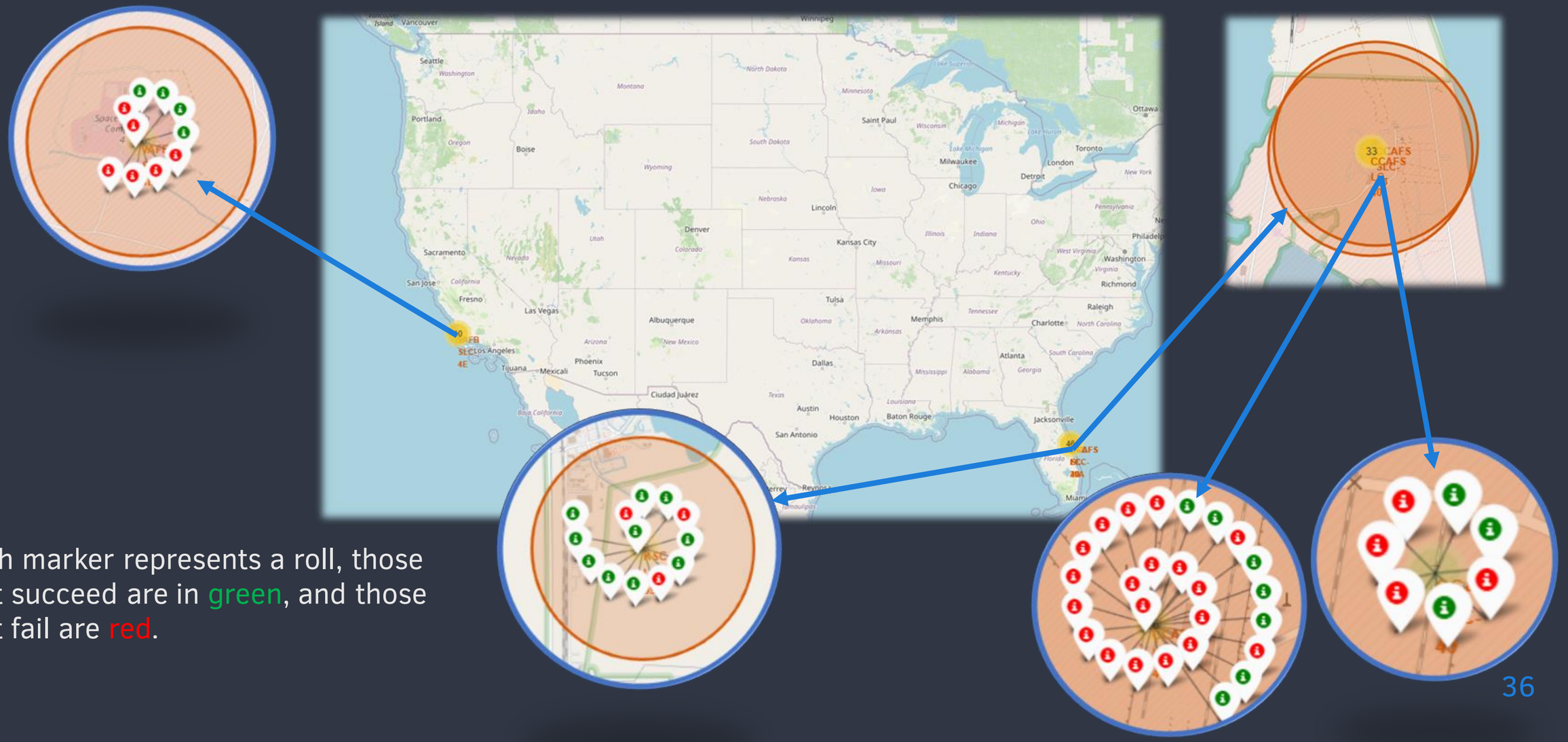
Section 3

Launch Sites Proximities Analysis

Launch Sites on Folium Map



Launch Sites Successes and Failures

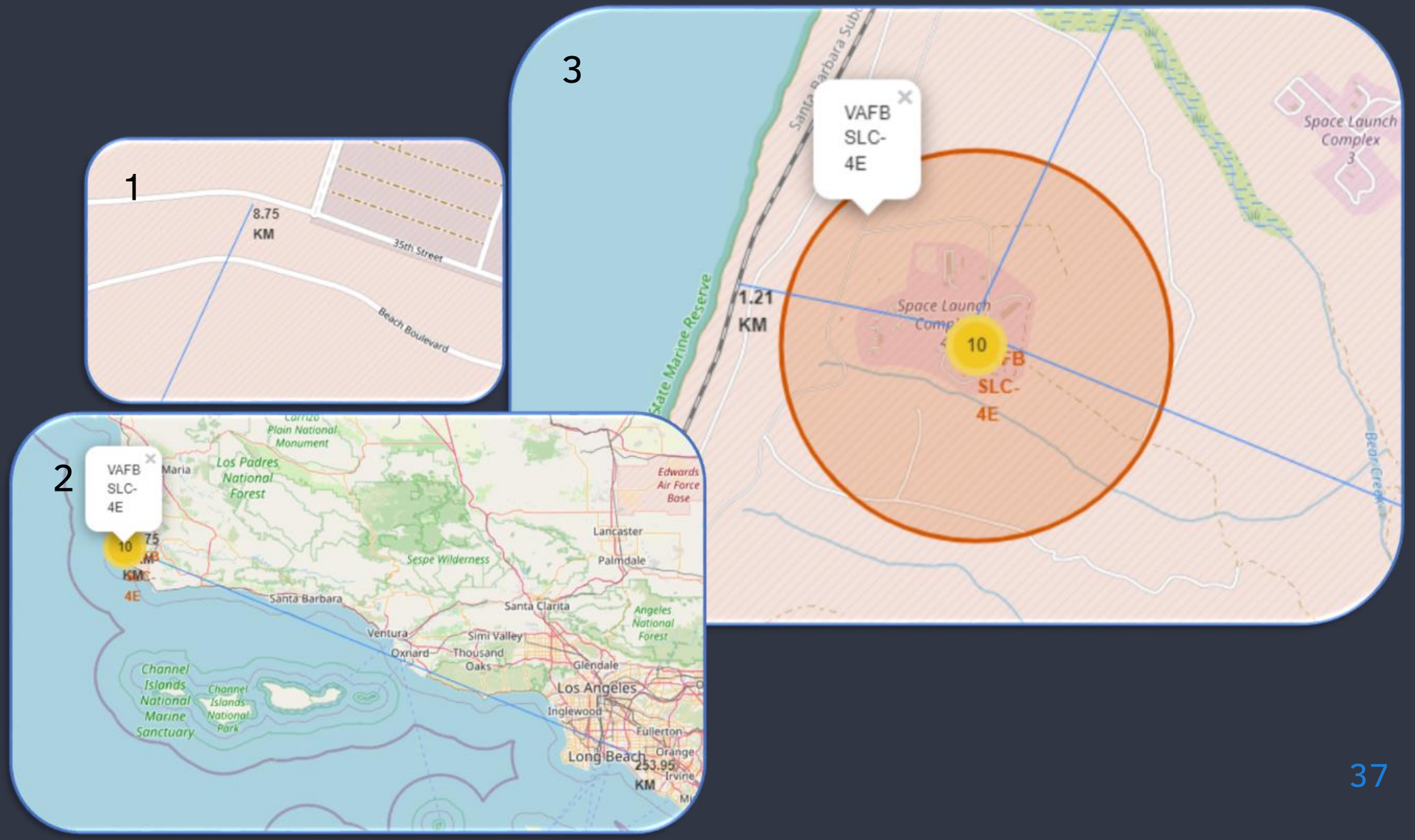


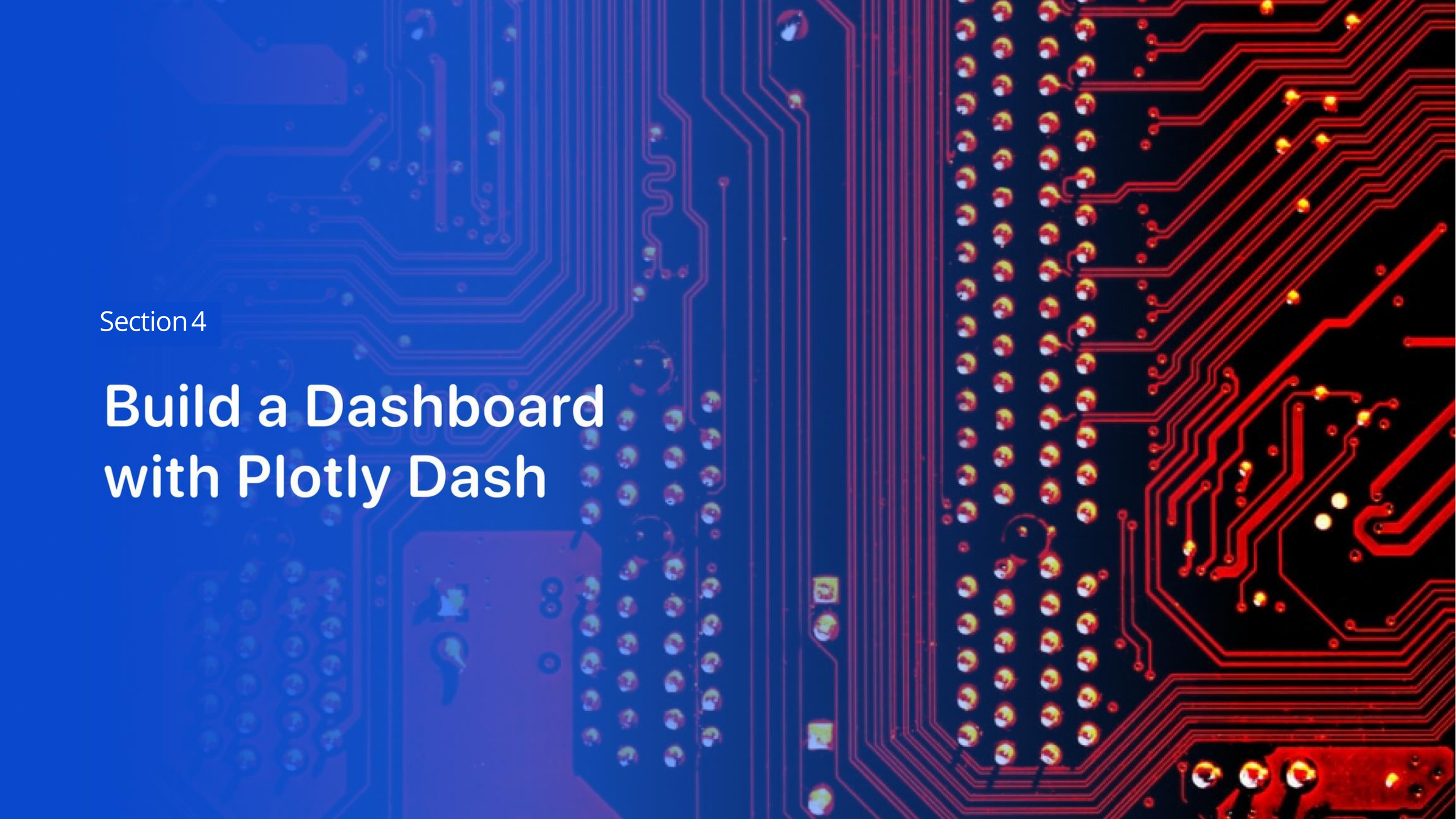
Each marker represents a roll, those that succeed are in green, and those that fail are red.

Distance from launch site to specific locations

Distance from launch site to specific locations was made by using latitude and longitude taken from the map coursor.

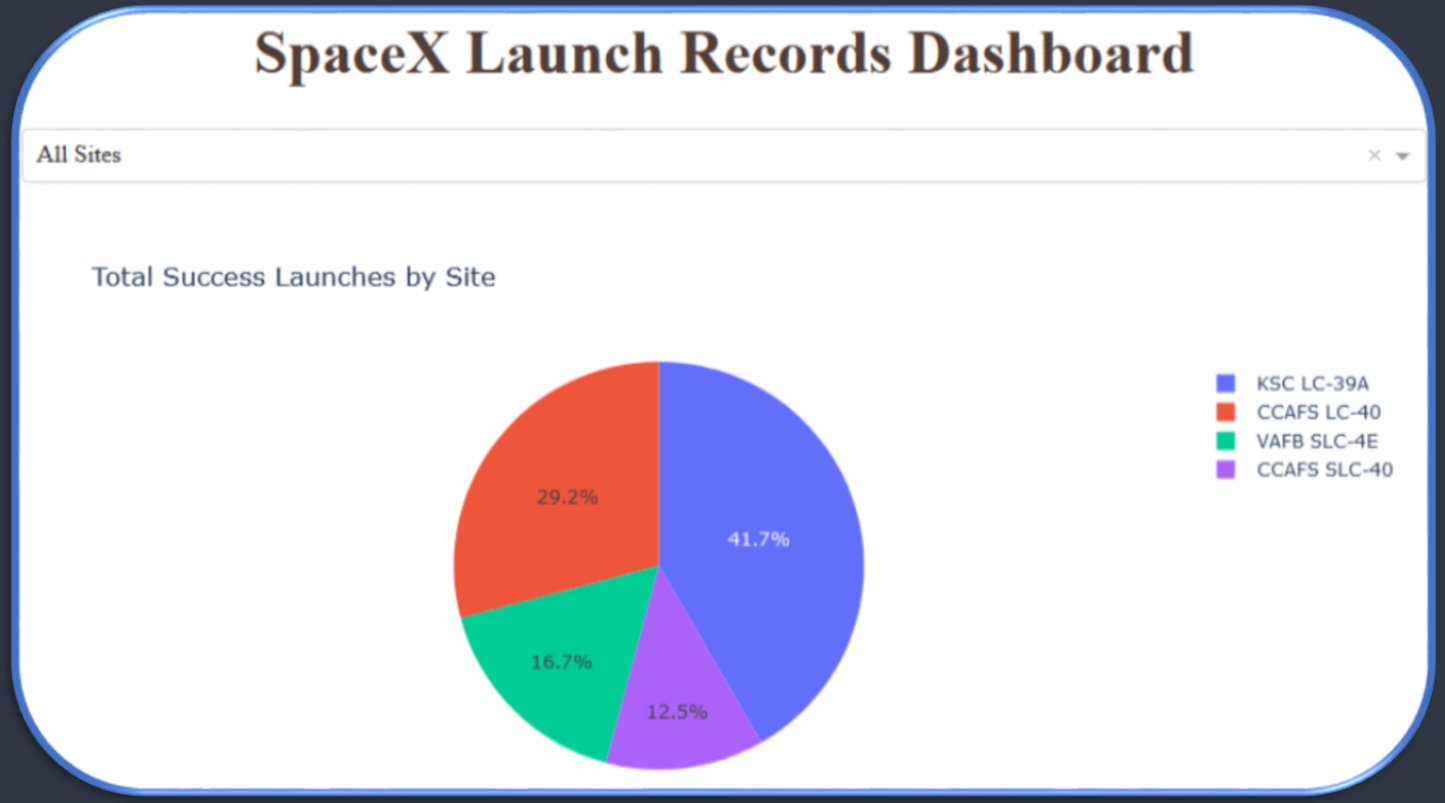
Figure 1 represents the distance to a highway.
Figure 2 represents the distance to a railway.
Figure 3 represents the distance to Los Angeles.




Section 4
Build a Dashboard with Plotly Dash

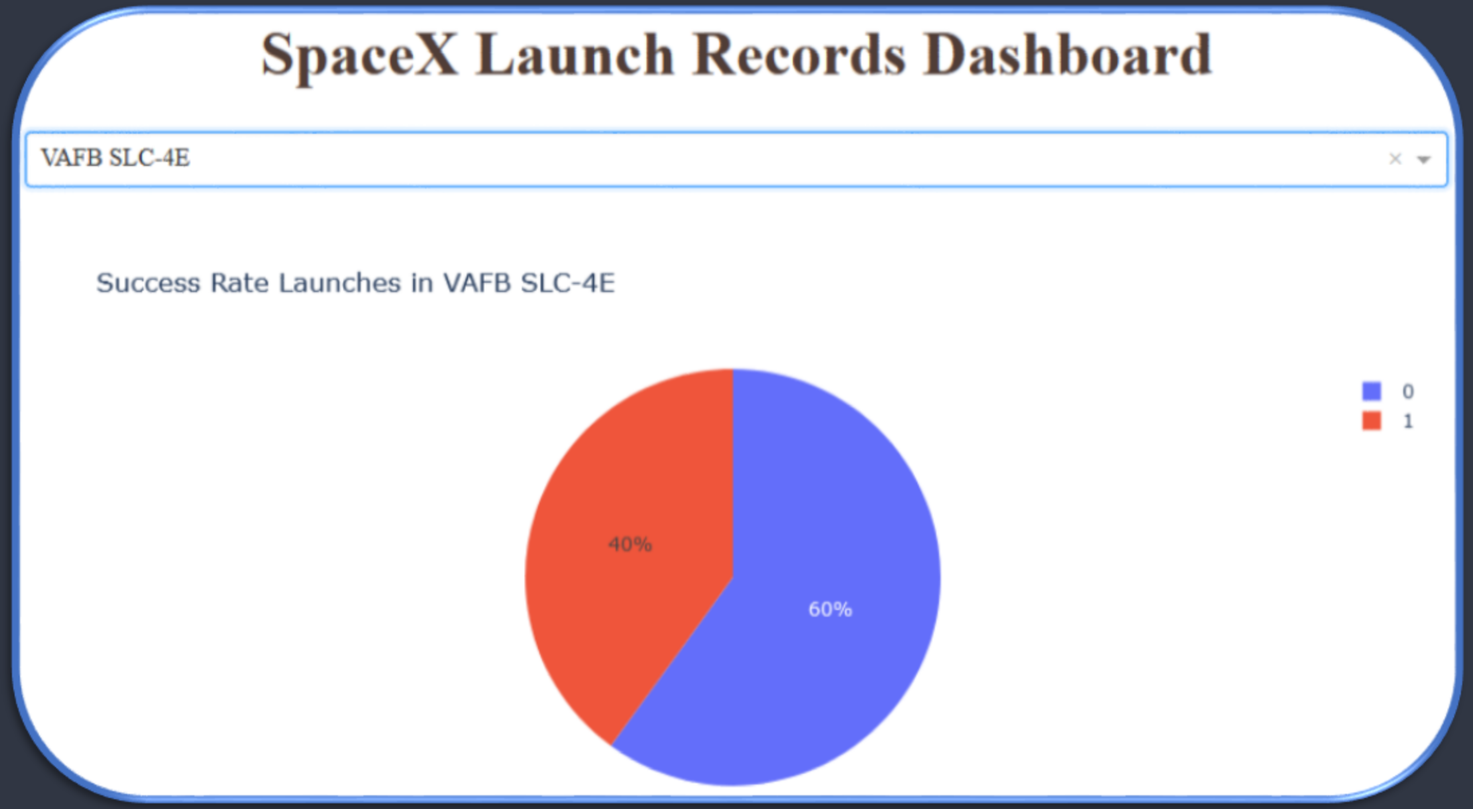
Launch Sites Successful Outcome

This pie chart presents the comparison of the success outcome at each launch site. It can be observed that KSC LC-39A has the most successfull outcome whereas CCAFS SLC-40 the worst successfull outcome.



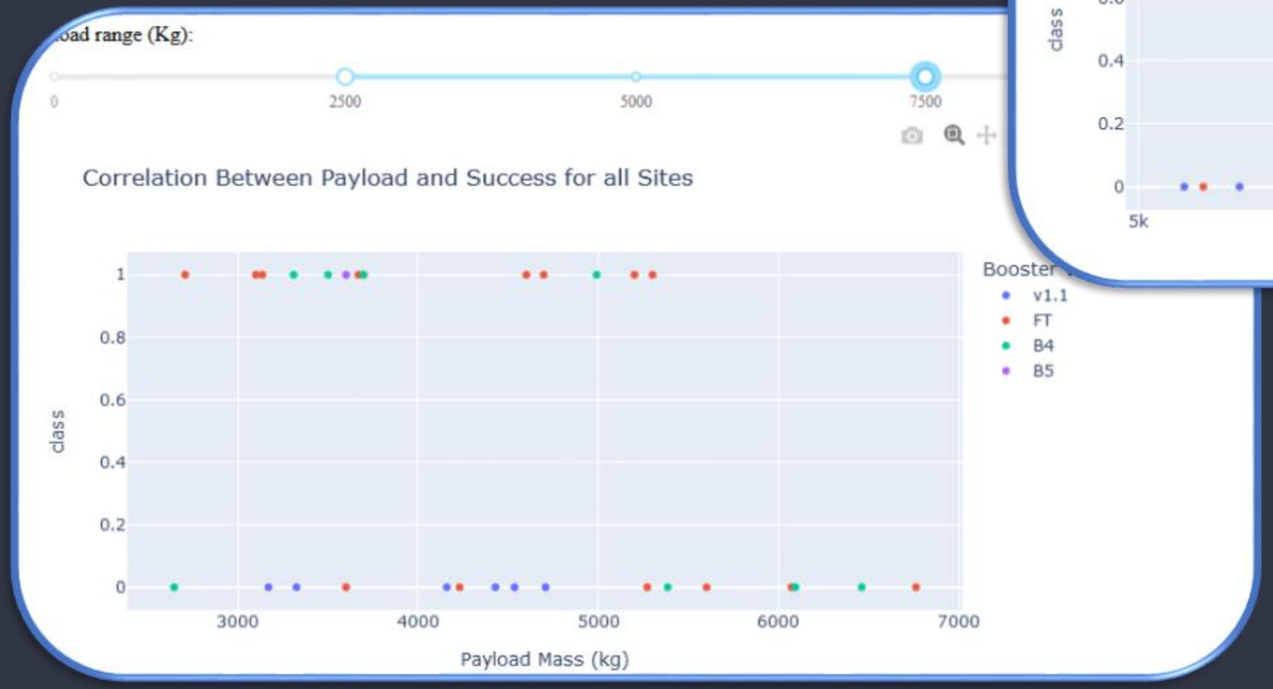
Launch Site VAFB SLC-4E Success Rate

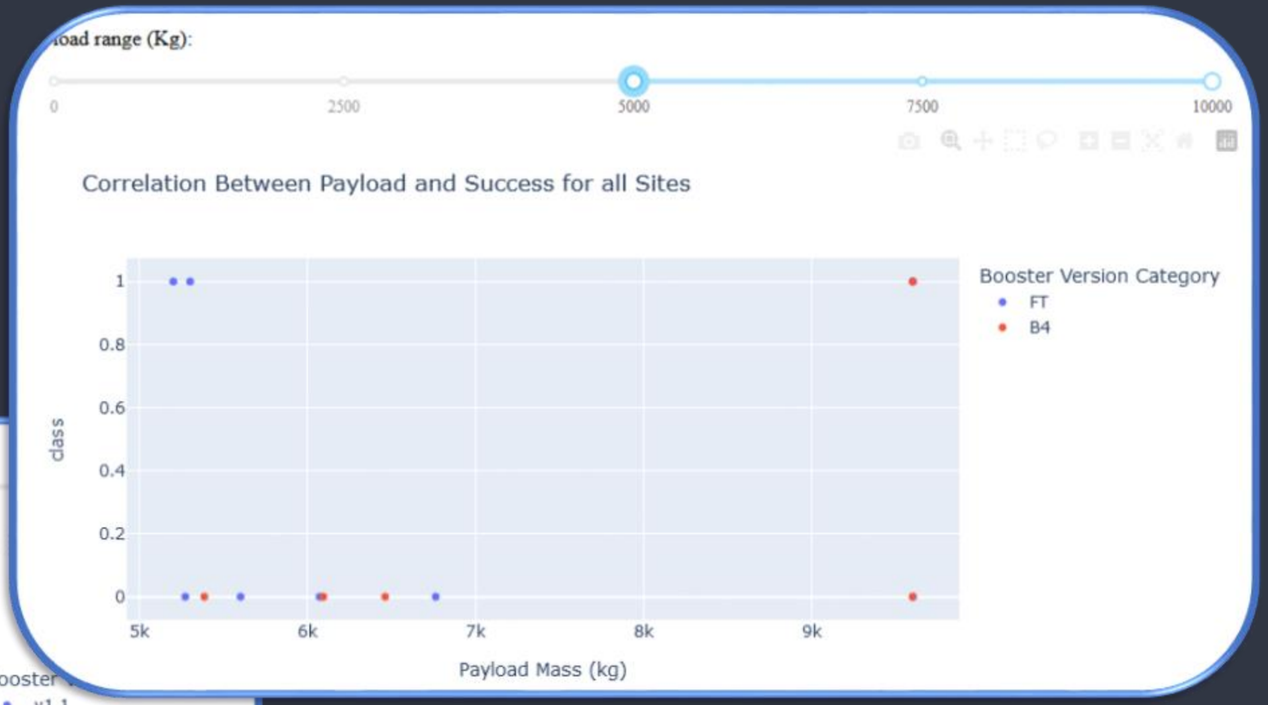
Although VAFB SLC-4E represents only 16% of total launches, its success rate is 60%.



Launch Outcome vs. Payload Mass by Booster Version

Apparently, spacecrafts with payload between 2.5 tons and 7.5 tons are more likely to succeed.
Only two Boost Version were released with payloads above 5 tons.



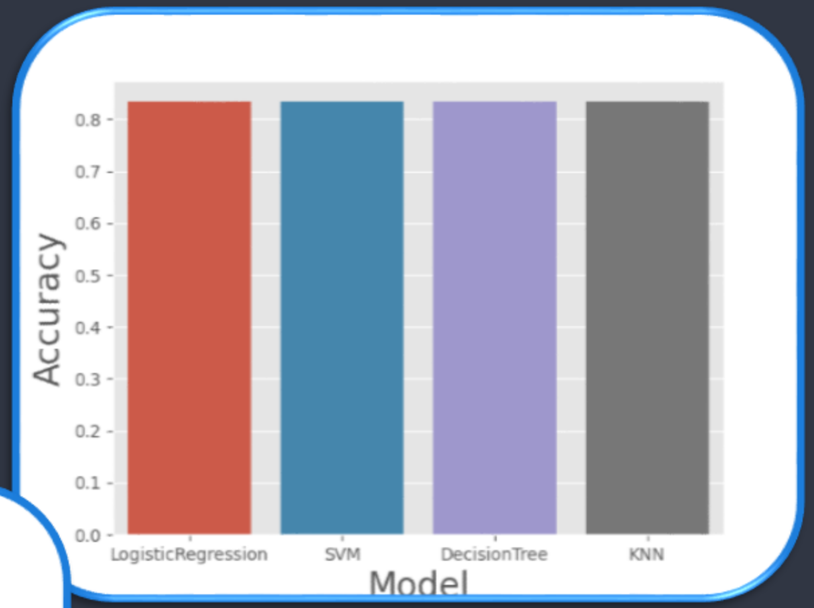


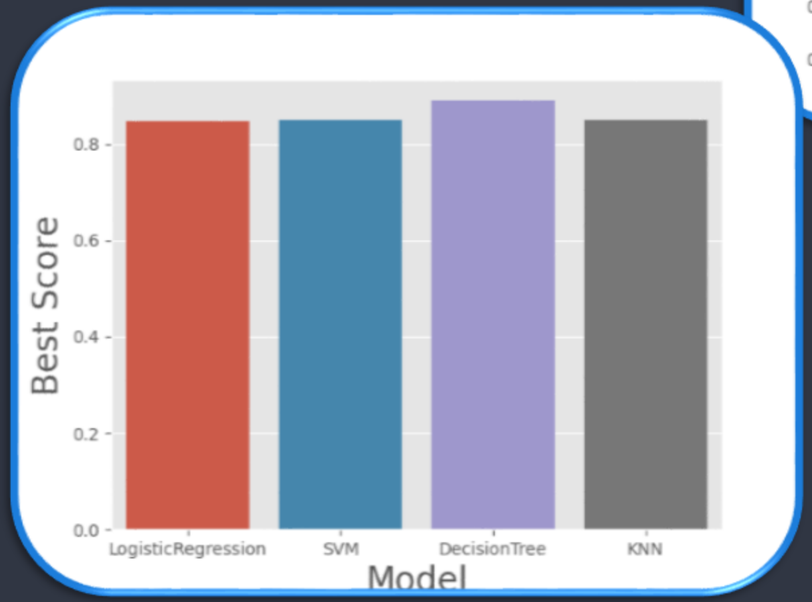
Section 5

Predictive Analysis (Classification)

Classification Accuracy

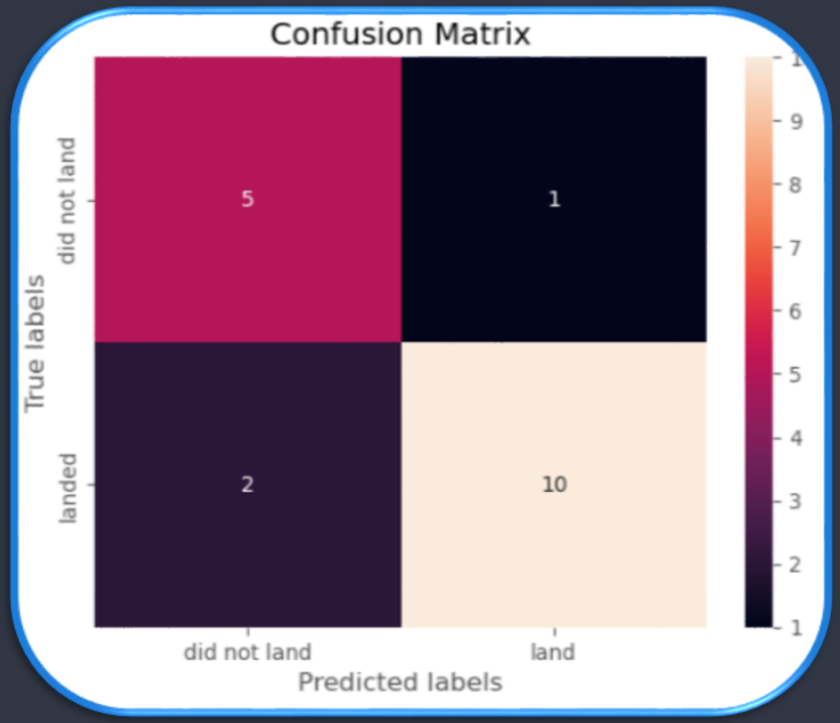
- The bar charts shows Accuracy and Best Score from each model;
- Decision Tree is the model that best fits the data is accuracy is equal to the others, but it was the one that best scored.





Confusion Matrix

- The main diagonal represents the correct predictions of the algorithm and the other diagonal represents the wrong ones;
- So it correctly predicted 10 times that the outcome was successfull and 5 times that the outcome was unsucessfull;
- Also, it wrongly predicted 2 times that the outcome was unsuccessfull and 1 time that the outcome was successfull when in fact it was the opposite.



Conclusions

- Falcon 9 launches took place at 4 different launch sites: CCAFS LC-40, VAFB SLC-4E, KSC LC-39A and CCAFS SLC-40;
- As expected, in the first 20 launches the level of success of the missions was very low, however with the gain of experience it is noticed that this level has increased considerably;
- Most spacecraft carry payloads between 2 tons and 7 tons, in addition, those that carry payloads above 7 tons were almost always successful;
- Launches to ES-L1, GEO, HEO and SSO orbits were 100% successful, while SO 0%;
- Launches to ISS and GTO orbits were the most frequent;
- VLEO orbit is associated with spacecraft with a large amount of payload (above 15 tons);
- As mentioned in the second point, first launches were not successful, so much so that from 2010 to 2013 the success rate was 0%;
- From 2013 onwards, better results occurred and the success rate from 2016 to 2020 remained above 60%;
- The launch sites that provided the best results were KSC LC-39A followed by CCAFS LC-40;
- When using the machine learning model created (Decision Tree) to predict whether the mission will be a success, it is expected to be correct by approximately 88%.

Appendix

Data Collection – SpaceX API

- On the right functions
- Below the code used to deal with collected data

```
# 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)]
```

```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])

# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
        response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
        Longitude.append(response['longitude'])
        Latitude.append(response['latitude'])
        LaunchSite.append(response['name'])

# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])

# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Appendix

Data Collection – Web Scrap

- Parse and store the data

```
xtracted_row = 0
#Extract each table
or 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()
      else:
          flag=False
      #get table element
      row=rows.find_all('td')
      #if it is number save cells in a dictonary
      if flag:
          extracted_row += 1
          # Flight Number value
          # TODO: Append the flight_number into launch_dict with key `Flight No.`
          datatimelist=date_time(row[0])
          launch_dict['Flight No.'].append(flight_number)
          #print(flight_number)

          # Date value
          # TODO: Append the date into launch_dict with key `Date`
          date = datatimelist[0].strip(',')
          launch_dict['Date'].append(date)
          #print(date)

          # Time value
          # TODO: Append the time into launch_dict with key `Time`
          time = datatimelist[1]
          launch_dict['Time'].append(time)
          #print(time)

          # Booster version
          # TODO: Append the bv into launch_dict with key `Version Booster`
          bv=booster_version(row[1])
          if not(bv):
              bv=row[1].a.string
          launch_dict['Version Booster'].append(bv)
          #print(bv)

          # Launch Site
          # TODO: Append the bv into launch_dict with key `Launch Site`
          launch_site = row[2].a.string
          launch_dict['Launch site'].append(launch_site)
          #print(launch_site)
```

```
# Payload
# TODO: Append the payload into launch_dict with key `Payload`
payload = row[3].a.string
launch_dict['Payload'].append(payload)
#print(payload)

# Payload Mass
# TODO: Append the payload_mass into launch_dict with key `Payload mass`
payload_mass = get_mass(row[4])
launch_dict['Payload mass'].append(payload_mass)
#print(payload)

# Orbit
# TODO: Append the orbit into launch_dict with key `Orbit`
orbit = row[5].a.string
launch_dict['Orbit'].append(orbit)
#print(orbit)

# Customer
# TODO: Append the customer into launch_dict with key `Customer`
if (row[6].a is not None):
    customer=row[6].a.string
else:
    customer=''
launch_dict['Customer'].append(customer)
#print(customer)

# Launch outcome
# TODO: Append the launch_outcome into launch_dict with key `Launch outcome`
launch_outcome = list(row[7].strings)[0]
launch_dict['Launch outcome'].append(launch_outcome)
#print(launch_outcome)

# Booster landing
# TODO: Append the launch_outcome into launch_dict with key `Booster landing`
booster_landing = landing_status(row[8])
launch_dict['Booster landing'].append(booster_landing)
#print(booster_landing)
```

Thank you!