IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Facundo Sigal
2025-11-12

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- The commercial space age is here, companies are making space travel affordable for everyone. Starlink, a satellite internet constellation providing satellite Internet access. Sending manned missions to Space. One reason SpaceX can do this is the rocket launches are relatively inexpensive. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards 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. Spaces X's Falcon 9 launch like regular rockets.
- Stage two, or the second stage, helps bring the payload to orbit, but most of the work is done by the first stage.
- Space Y that would like to compete with SpaceX founded by Billionaire industrialist Allon Musk. In this project, I want to determine the price of each launch, by gathering information about Space X and creating dashboards. I will also determine if SpaceX will reuse the first stage, by training a machine learning model and use public information to predict if SpaceX will reuse the first stage.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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

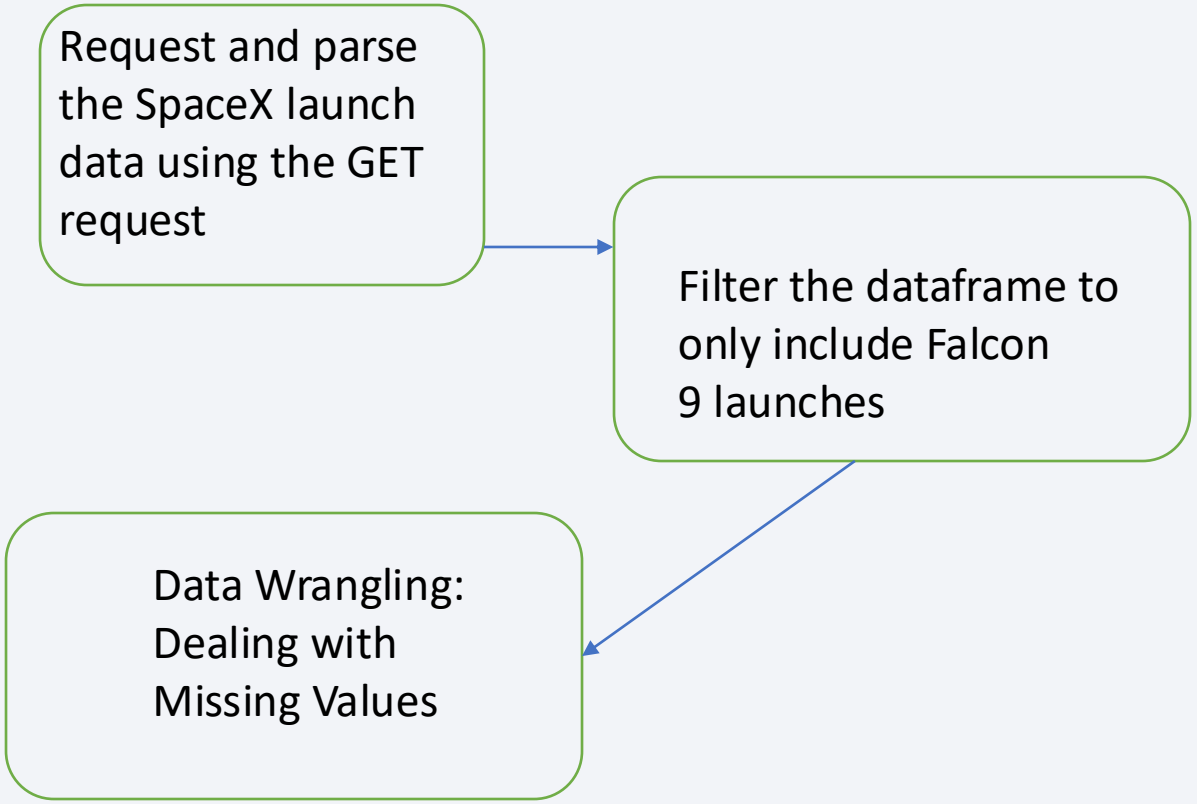
- I worked with SpaceX launch data that is gathered from an API, specifically the SpaceX REST API.
- This API provides data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.
- The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/. There are different end points, for example: /capsules and /cores We work with the endpoint api.spacexdata.com/v4/launches/past.
- I performed a get request using the requests library to obtain the launch data, which is used to get the data from the API.
- The response is in the form of a JSON, specifically a list of JSON objects.
- To convert this JSON to a dataframe, I used the json_normalize function, which allowed me to “normalize” the structured json data into a flat table.

Data Collection – SpaceX API

- These are the REST calls used for data collection:

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

response = requests.get(spacex_url)

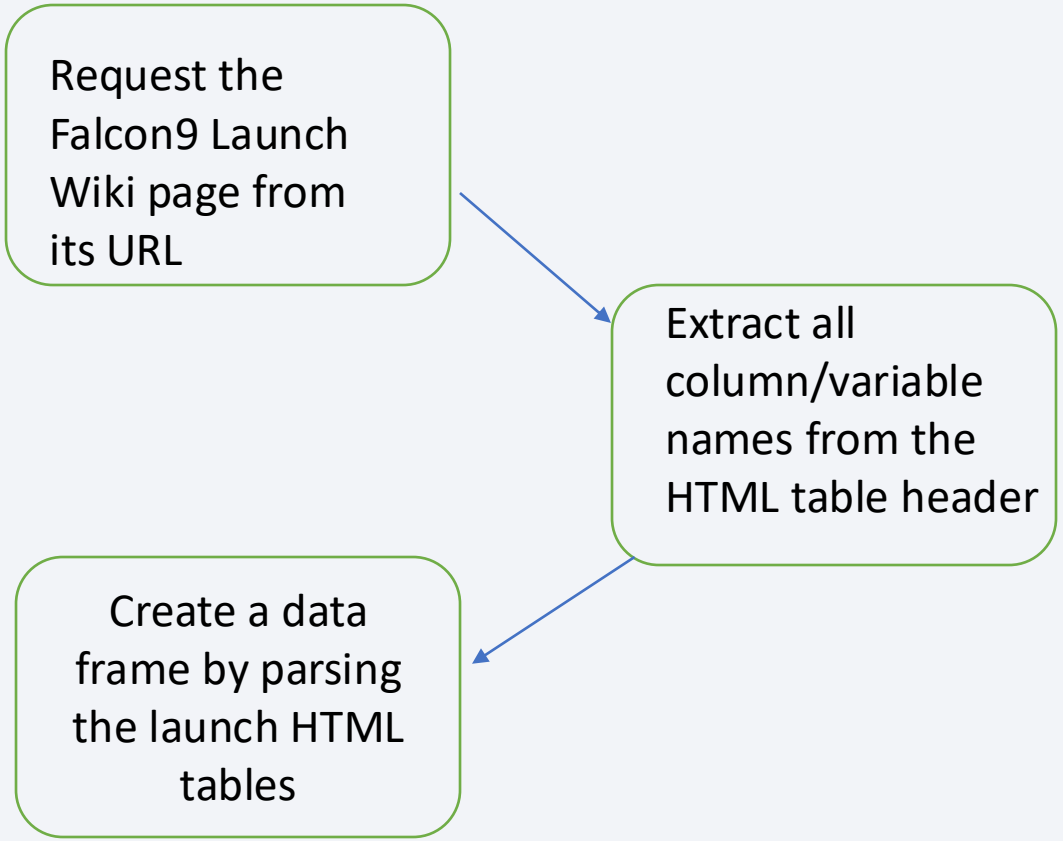


- GitHub URL of the completed SpaceX API calls notebook:

jupyter-labs-spacex-data-collection-api.ipynb

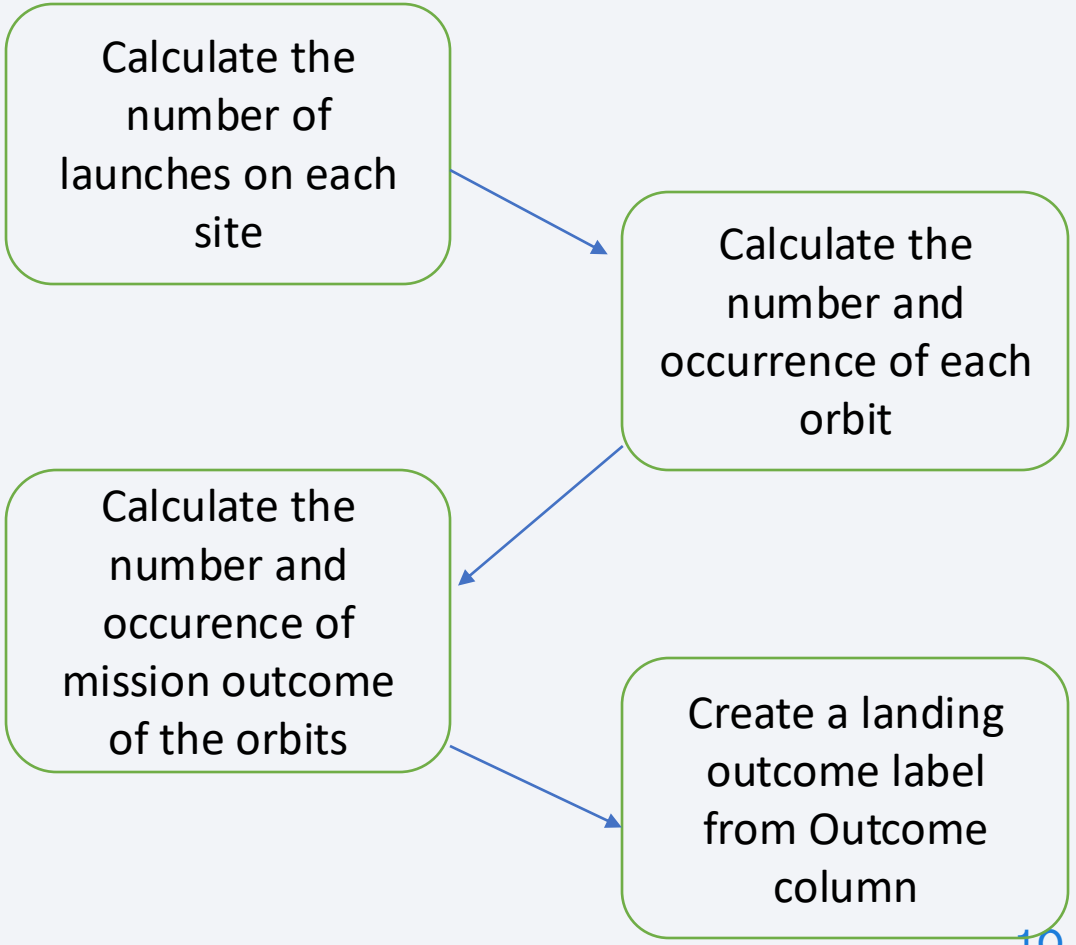
Data Collection - Scraping

- Web scraping of related Wiki pages.
- I used the Python BeautifulSoup package to web scrape some HTML tables that contain valuable Falcon 9 launch records.
- Then I parsed the data from those tables and convert them into a Pandas data frame for further visualization and analysis.
- GitHub URL of the completed web scraping notebook:
- https://github.com/fsigal/DataScienceCapstoneIBM/blob/main/jupyter-labs-webscraping.ipynb



Data Wrangling

- I performed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.
- In this phase I converted the outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful
- GitHub URL of my completed data wrangling related notebooks:
- https://github.com/fsigal/DataScienceCapstonelBM/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



EDA with Data Visualization

- I performed some Exploratory Data Analysis using a database. Then I researched if the data could be used to automatically determine if the Falcon 9's first stage will land.
- I used the following plots:
  - Scatter plots, to see how the response in one variable is affected by changes in another one)
  - Bar chart, to compare success rate between different orbits
  - Line chart, to describe the evolution of success rate through time.
- The GitHub URL of my completed EDA with data visualization notebook is not presented because it could not be downloaded from Jupyter Lab.

EDA with SQL

- I performed the following SQL queries:
  - 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 all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.
  - 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 landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.
- GitHub URL of my completed EDA with SQL notebook: https://github.com/fsigal/DataScienceCapstoneIBM/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- I created these map objects:
  - a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
  - a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
  - For each launch site, a Circle object based on its coordinate (Lat, Long) values and Launch site name as a popup label
  - markers for all launch records. If a launch was successful then I use a green marker and if a launch was failed, I use a red marker
  - a marker with distance to a closest city, railway, highway, etc and a line between the marker to the launch site
- I added those objects in order to mark all launch sites on a map, mark the success/failed launches for each site on the map and calculate the distances between a launch site to its proximities
- The GitHub URL of my completed interactive map with Folium map is not presented because it could not be downloaded from Jupyter Lab.

Build a Dashboard with Plotly Dash

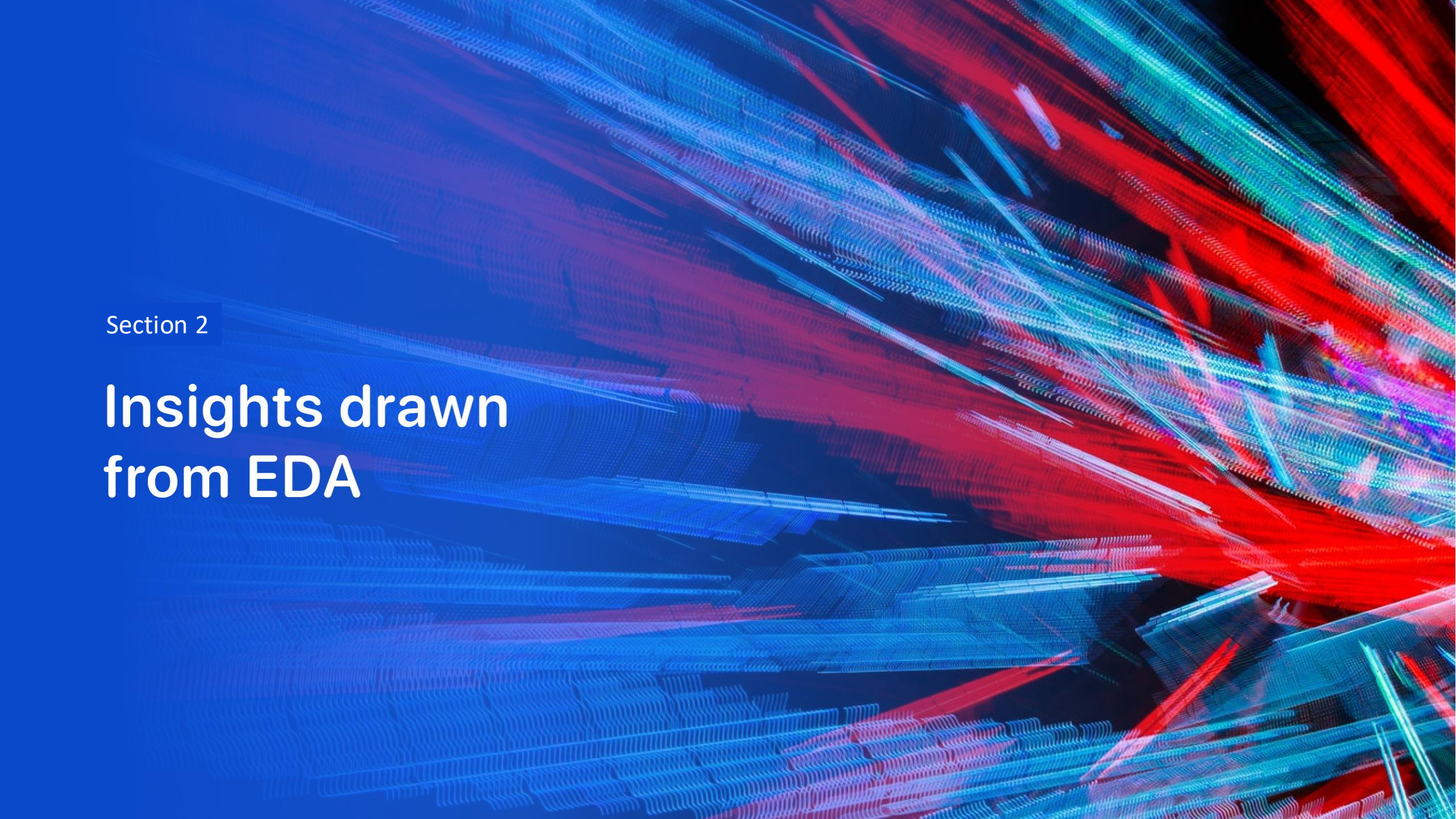
- Summarize what plots/graphs and interactions you have added to a dashboard
- I've added these objects to my dashboard:
  - a Launch Site Drop-down Input Component
  - a callback function to render success-pie-chart based on selected site dropdown
  - a Range Slider to Select Payload
  - a callback function to render the success-payload-scatter-chart scatter plot
- I added those plots and interactions to perform interactive visual analytics on SpaceX launch data in real-time
- The GitHub URL of your completed Plotly Dash lab: https://github.com/fsigal/DataScienceCapstoneIBM/blob/main/spacex-dash-app%20(1).py

Predictive Analysis (Classification)

- I created a machine learning pipeline to predict if the first stage will land given the data from the preceding labs.
- First, I create a column for the class, standardize the data and split into training data and test data.
- Then, I create a logistic regression object with a GridSearchCV object logreg_cv with cv = 10 and fit the object to find the best parameters to improve the results.
- Calculate the accuracy on the test data using the method score
- I did the same processes with other training algorithms, as SVM, Decision Tree, and KNN.
- The GitHub URL of your completed predictive analysis lab is not presented because it could not be downloaded from Jupyter Lab.

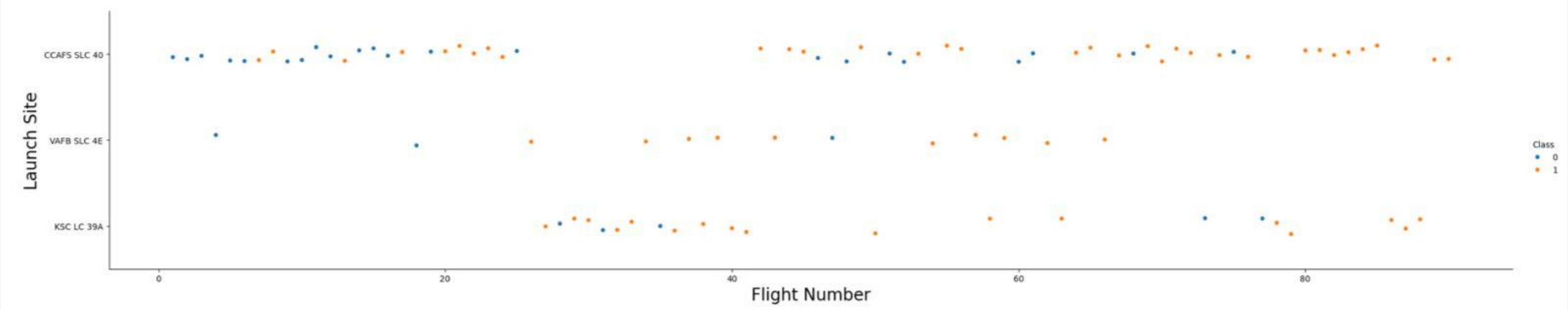
Results

- In the next chapter, I will show you:
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

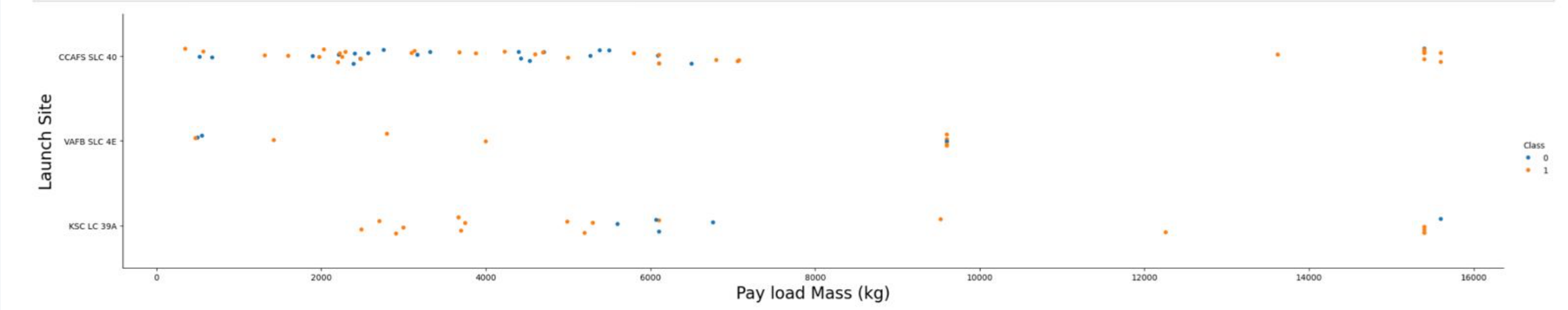
- Scatter plot of Flight Number vs. Launch Site



At the beginning, most of the flights were launched from CCAFS SLC 40 site. In the middle of the period, the other two sites were the most used launch sites, and in the end, again CCAFS SLC 40 was the most used site.

Payload vs. Launch Site

- Scatter plot of Payload vs. Launch Site



CCAFS SLC 40 site was used to launch rockets of all sizes. VAFB SLC is often used to launch small crafts and KSC LC 39A is the site of choice for larger rockets.

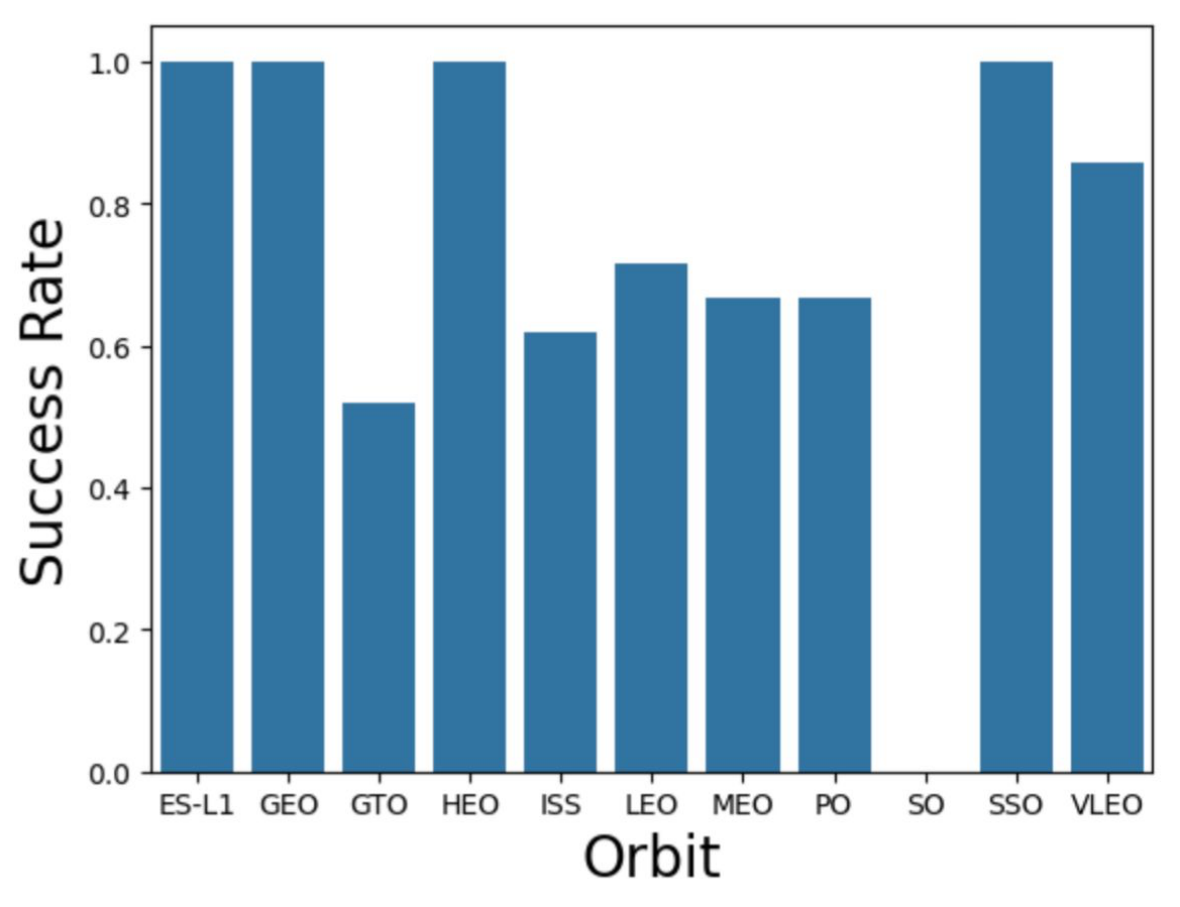
Success Rate vs. Orbit Type

- Bar chart for the success rate of each orbit type

There are four orbits with perfect success rate: ES-LI, GEO, HEO and SSO.

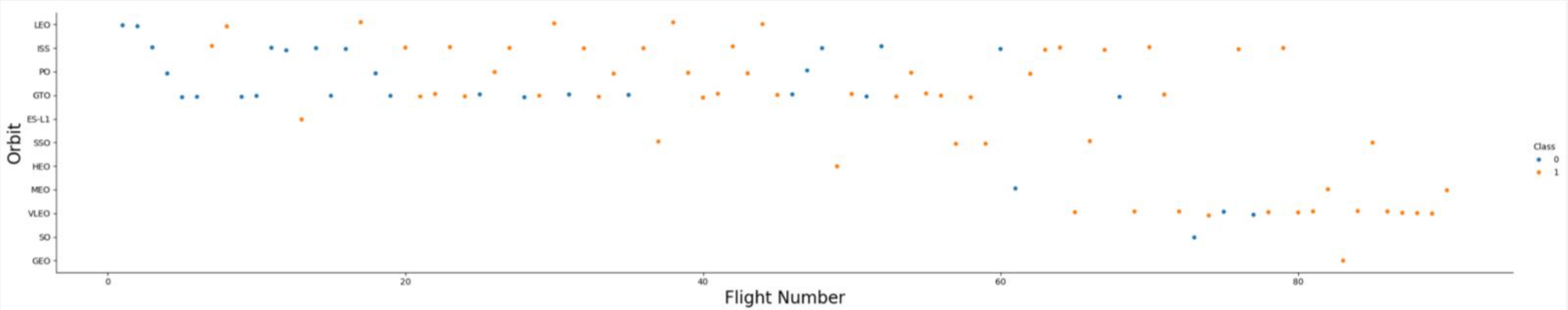
Only one orbit never participated on a successful landing: SO.

Most of the orbits has an intermediate success rate, between 0.5 and 0.7



Flight Number vs. Orbit Type

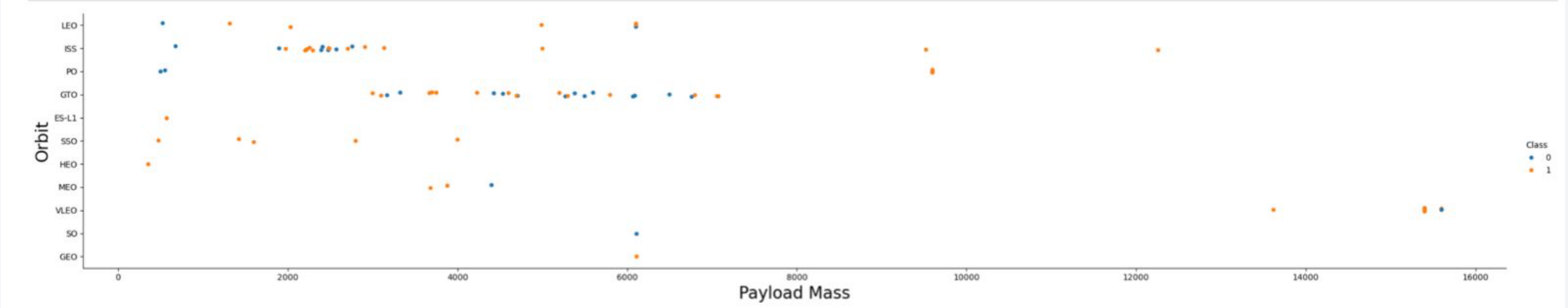
- Scatter plot of Flight number vs. Orbit type



First 60 flights went mostly to only four orbits: LEO, ISS, PO and GTO. After flight no. 60, other orbit types were included in flights planning.

Payload vs. Orbit Type

- Scatter plot of payload vs. orbit type

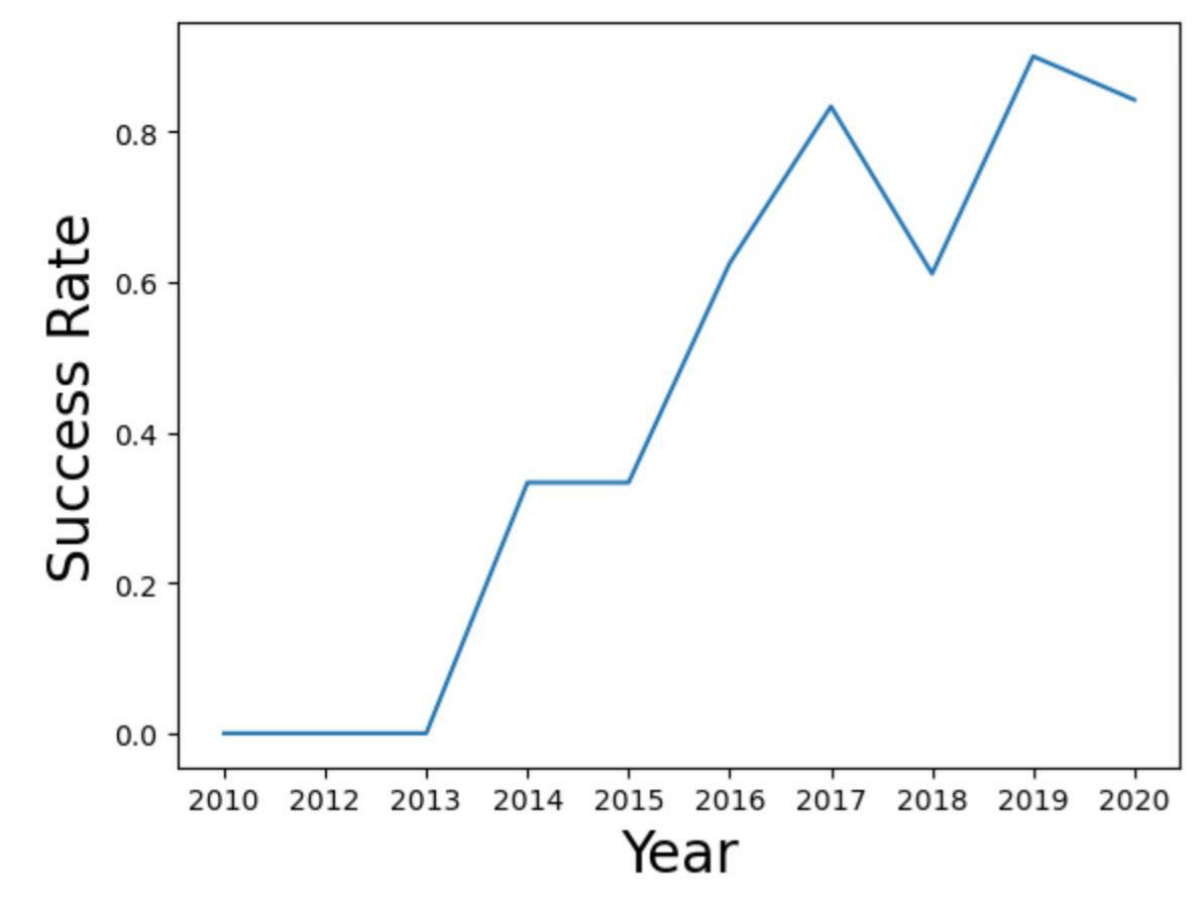


The largest aircrafts went to VLEO orbit.

Launch Success Yearly Trend

- Line chart of yearly average success rate

Until year 2013, there were no successful landings. After that year, success rate keeps growing every year, with an exception in year 2018, with success rate dropping to 0.6



All Launch Site Names

- There are four unique launch sites
- In this table we can see their names.

Launch Site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Launch Site Names Begin with 'CCA'

- Only two Launch Site Names Begin with CCA. Here we have a summary of the first 5 flights launched from those two sites:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

These five launches correspond to only one of those two sites: CCAFS LC-40

Total Payload Mass

- The total payload carried by boosters from NASA is 45,596 kgs. More than 45 tons carried from NASA to space!

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- When I filtered the booster version to match only F9 v1.1, the average payload mass carried was 2,534.67 kgs.

avg(PAYLOAD_MASS__KG_)
2534.6666666666665

First Successful Ground Landing Date

- The first successful landing outcome on ground pad is December 22nd, 2015. Before that day, there was no successful landing on ground pads.

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Here you can see the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000.

Payload
JCSAT-14
JCSAT-16
SES-10
SES-11 / EchoStar 105

Only four of them successfully landed on a drone ship with payload mass in that range

Total Number of Successful and Failure Mission Outcomes

- The total number of successful mission outcomes is 100 and only 1 failure.

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

Boosters Carried Maximum Payload

- These are the names of the boosters which have carried the maximum payload mass
- 12 boosters have carried the maximum payload mass. There is no booster that have carried more payload mass that every other booster.

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

- Only two launches failed landing in a drone ship in 2015. One of them in January and the other in April. Their booster versions, and launch site names are shown in the following table.

substr(Date,6,2)	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

- The following ranking sorts 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.
- On those dates, the most frequent outcome was no attempt. Among the other flights, the most frequent outcomes were Successful landing on a drone ship and Fail to land on a drone ship.

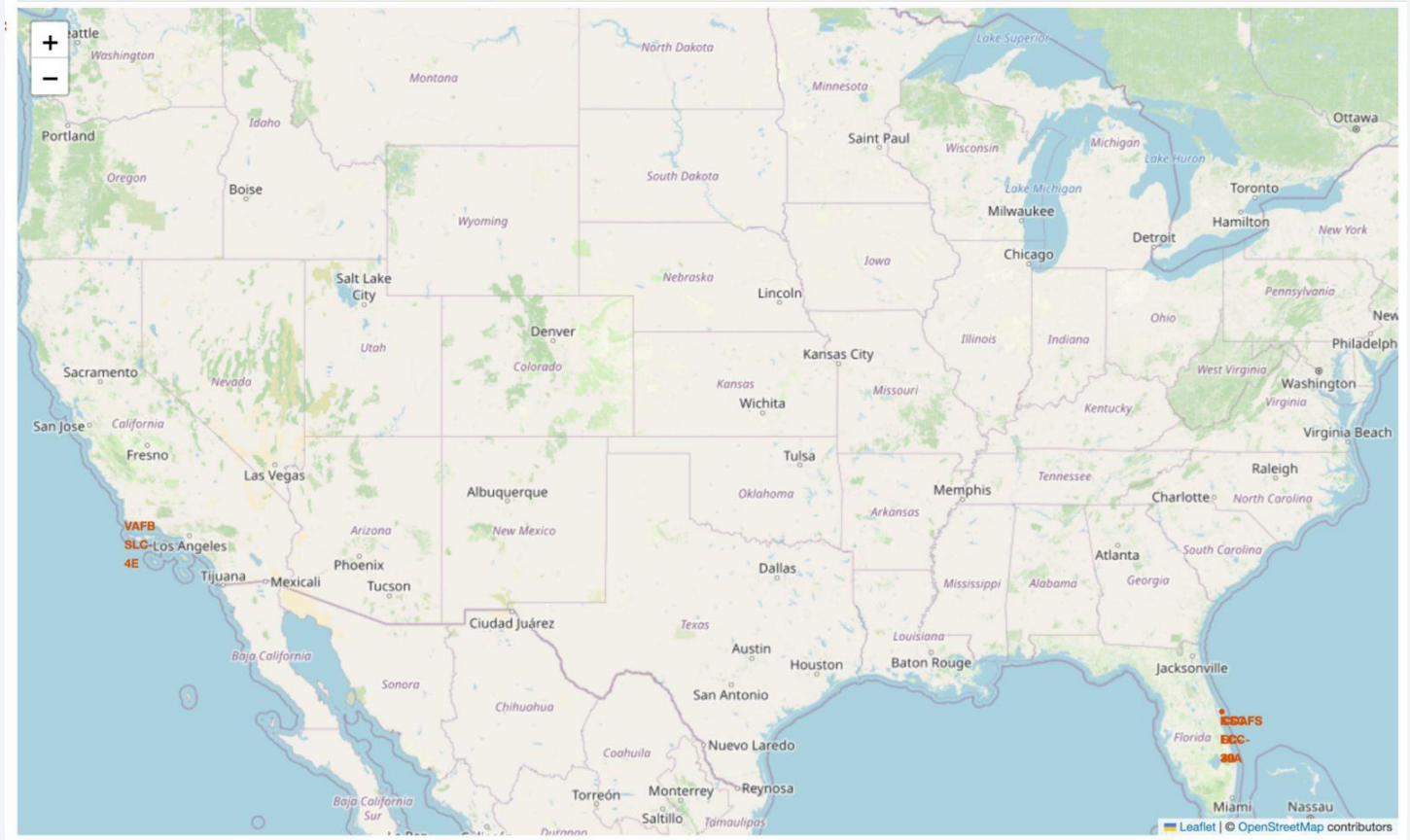
Landing_Outcome	Cuenta
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

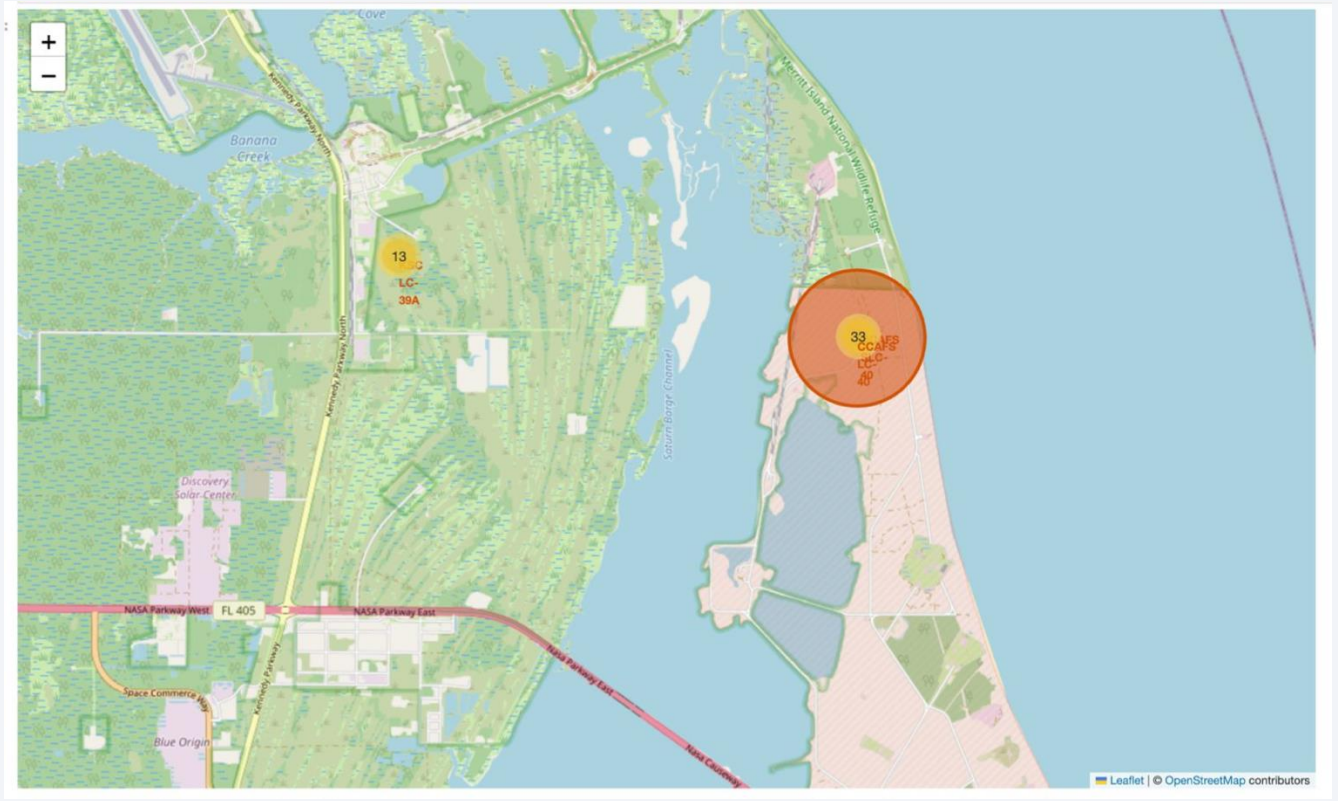
Geolocation of all launch sites

- Almost every launch site included on my analysis is located near the Atlantic coast in the state of Florida.
- The only one that is not located in the eastern coast is located in California.



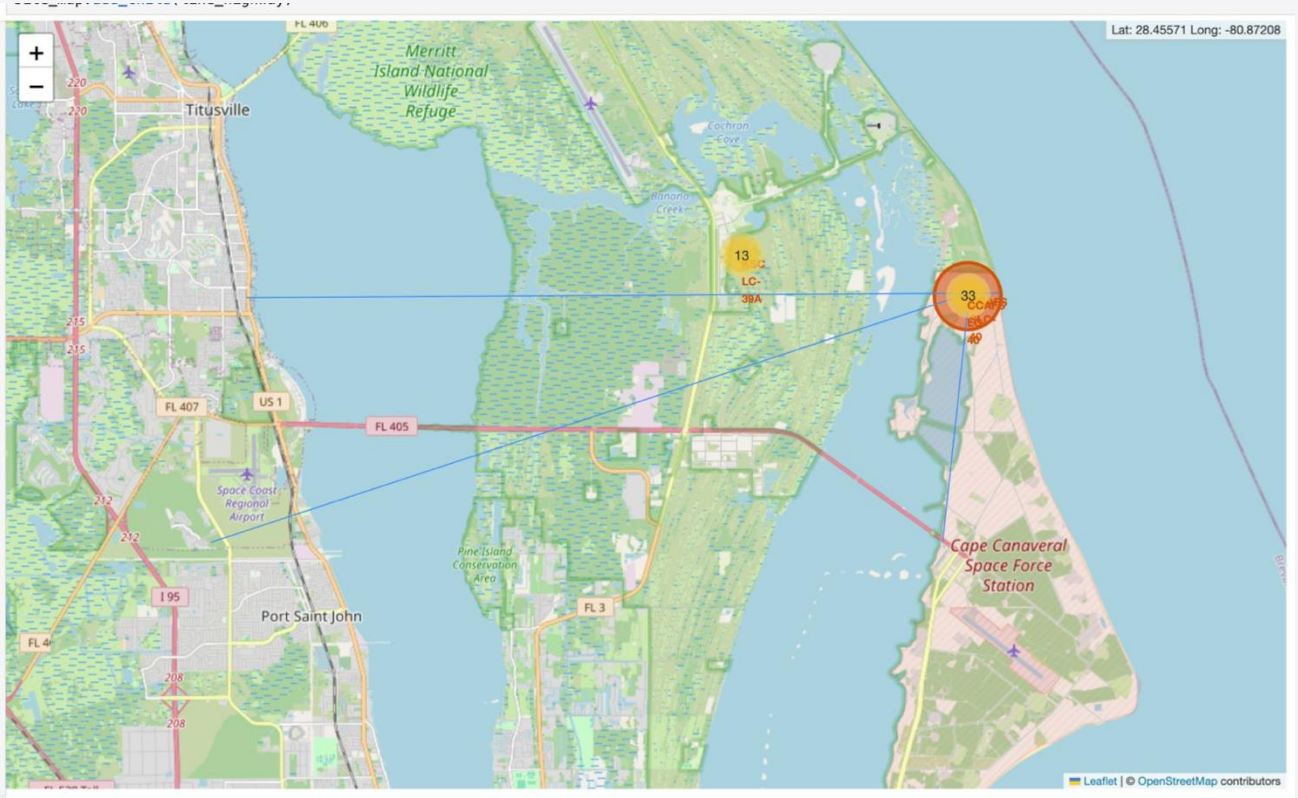
Geolocation of Florida state's launches

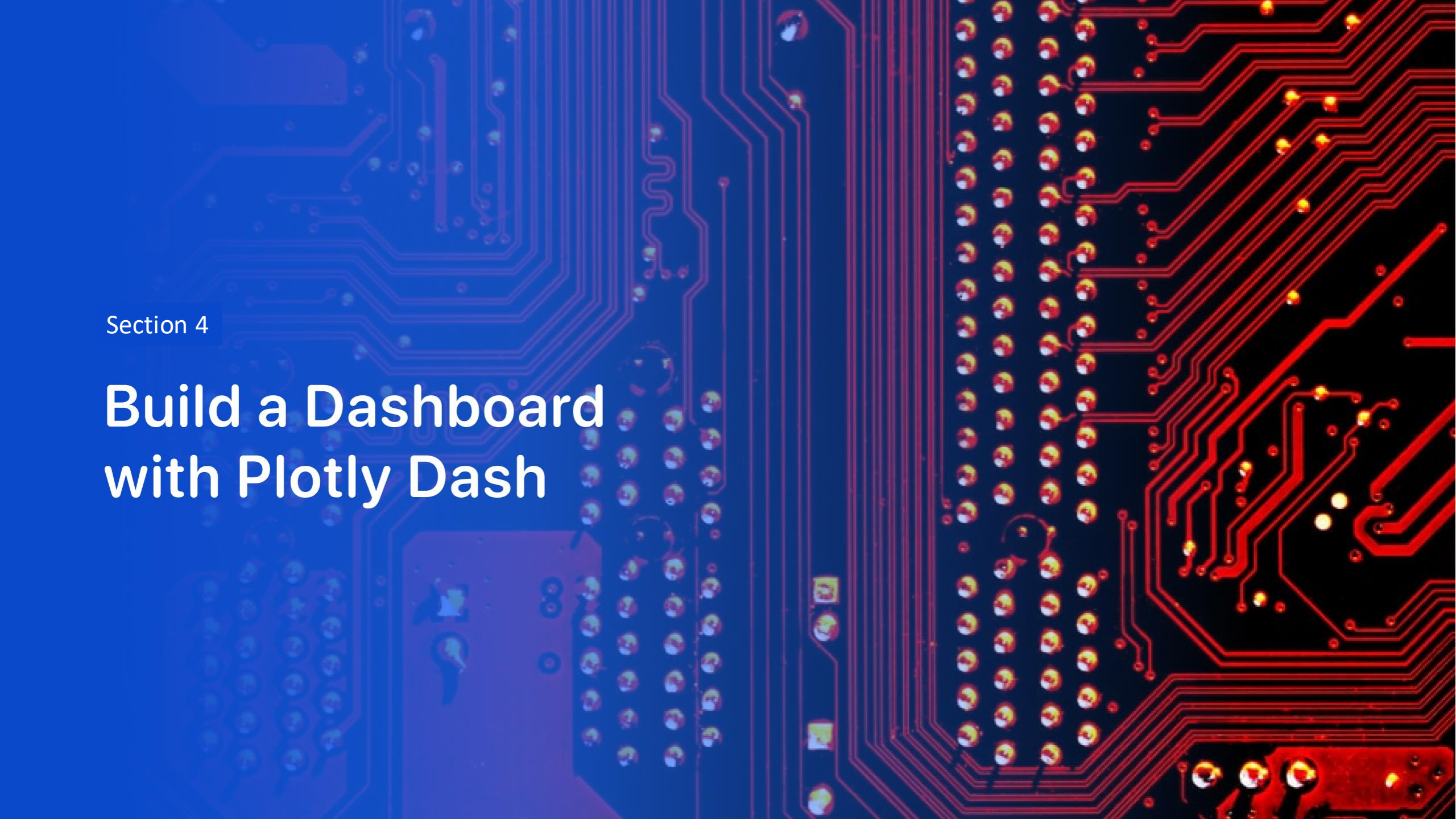
- 46 launches were made in Florida.
- 33 of them were performed in two sites that are so close to each other that they are shown in one cluster on the map.
- The other 13 launches were made in a site that is located some miles away from the other aforementioned launch sites.



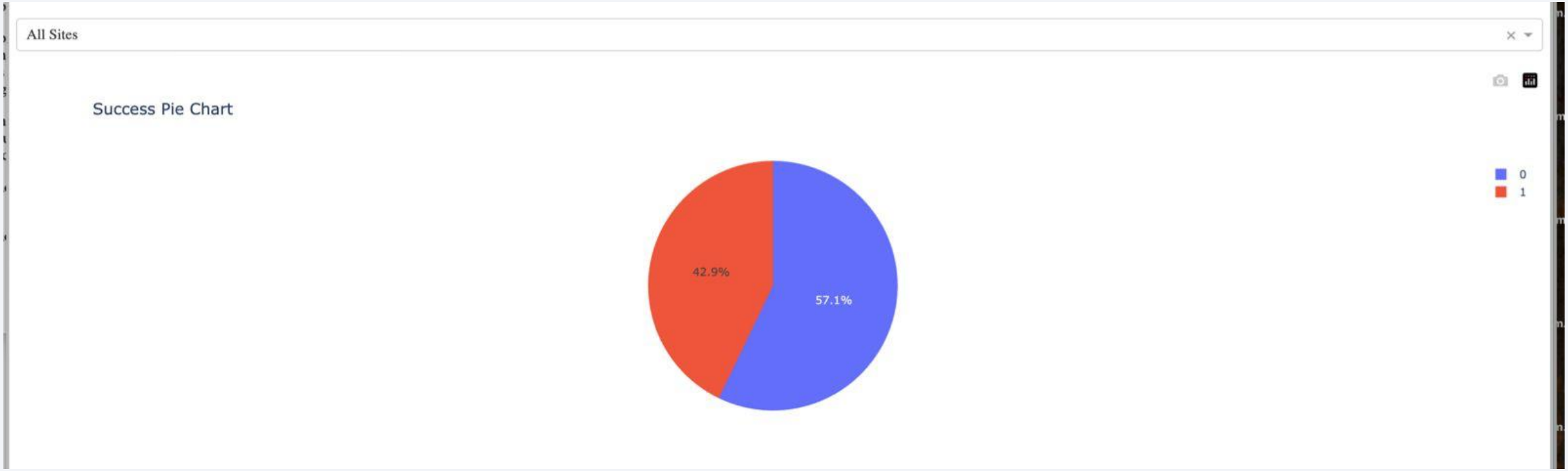
Launch site proximity to some points of interest

- Florida's launch sites are close to coastline, railways and highways.
- The map shows the distances between those spots and the main launch site.
- The nearest highway is closer to launch site than the nearest railway.
- The site is close to coastline and far from any important city.



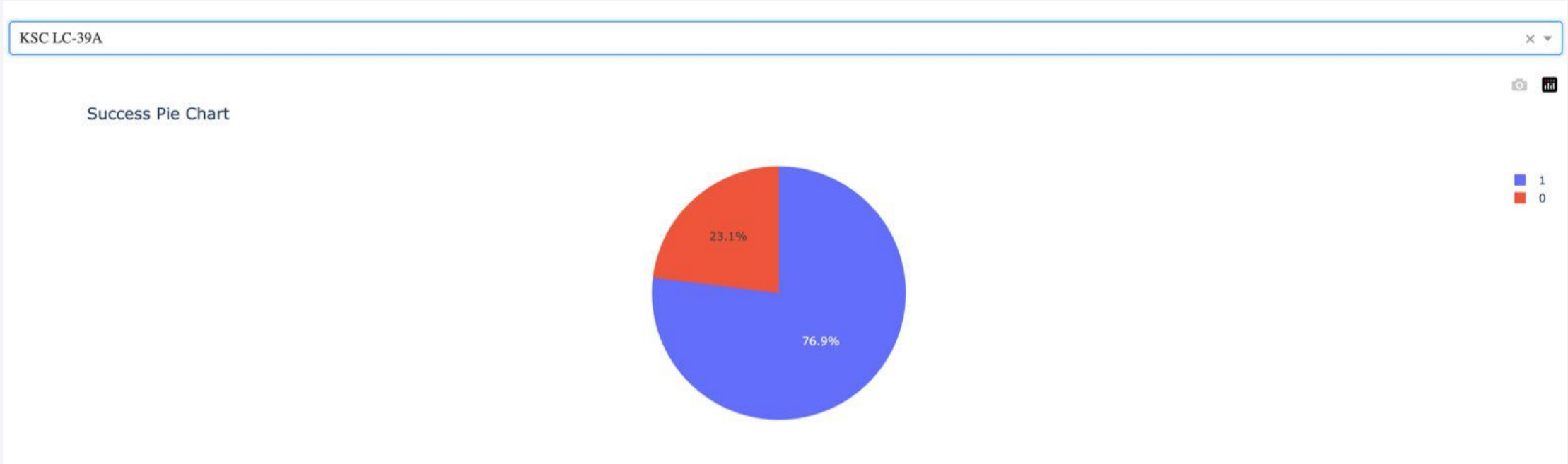

Section 4
Build a Dashboard with Plotly Dash

Success Frequency for all sites



In this pie chart we can see the successful (blue) and unsuccessful landings (red) for all sites. 57% of the launches ended with a successful landing.

<Dashboard Screenshot 2>

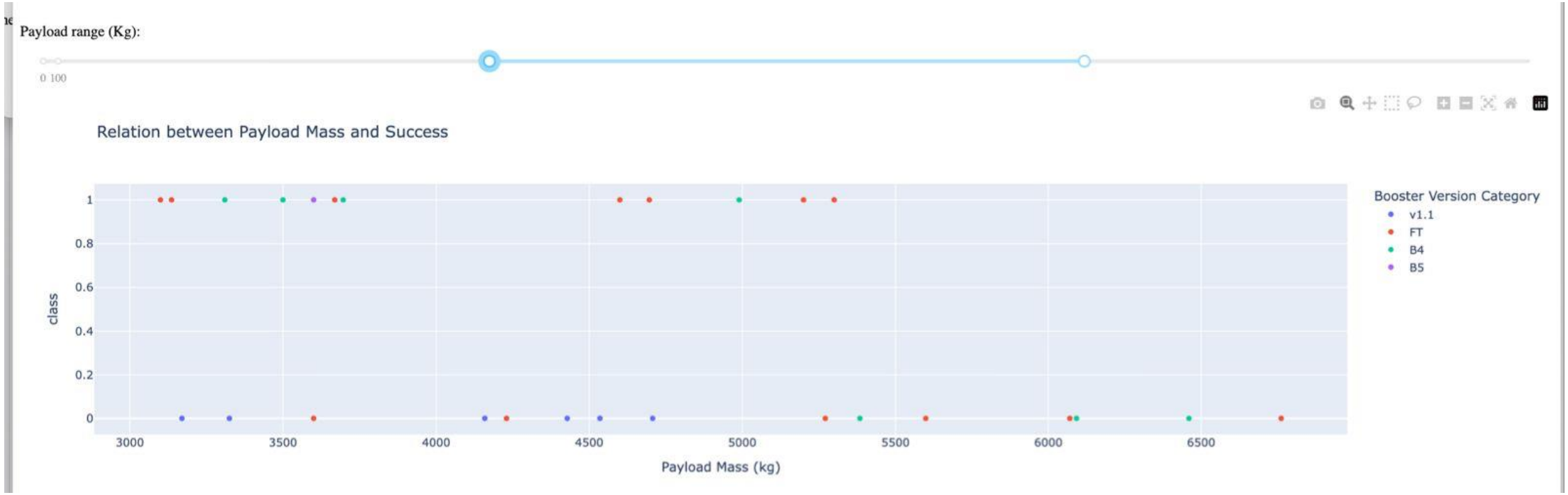


The site with the highest success rate is shown in this pie chart. 76.9% of launches from KSC LC-39A site ended successfully.

Success rate vs Payload Mass / Booster Version

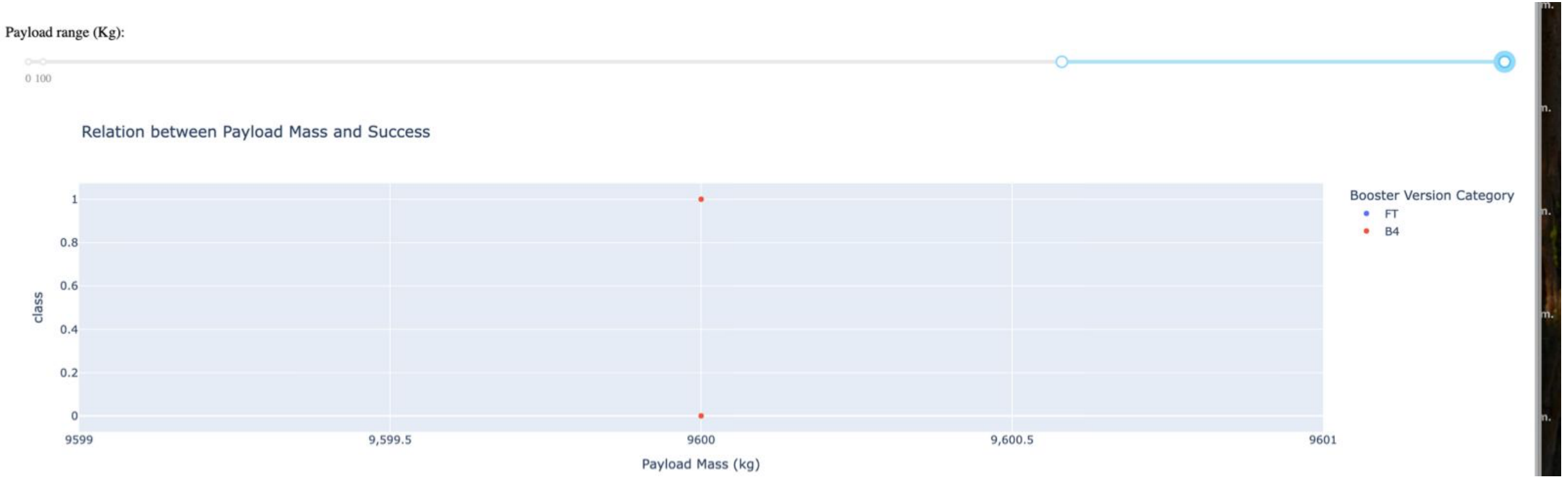
Now I analyze the relation between success and payload mass. For launches with payload mass under 3,000 kgs, the success/unsuccess ratio is near 50/50. The Booster Version Category with more successful landings in this range is FT.

Success rate vs Payload Mass / Booster Version



For launches with payload mass between 3,000 and 7,000 kgs, unsuccessful landings are more frequent than successful ones. The Booster Version Category with more successful landings in this range is also FT.

Success rate vs Payload Mass / Booster Version



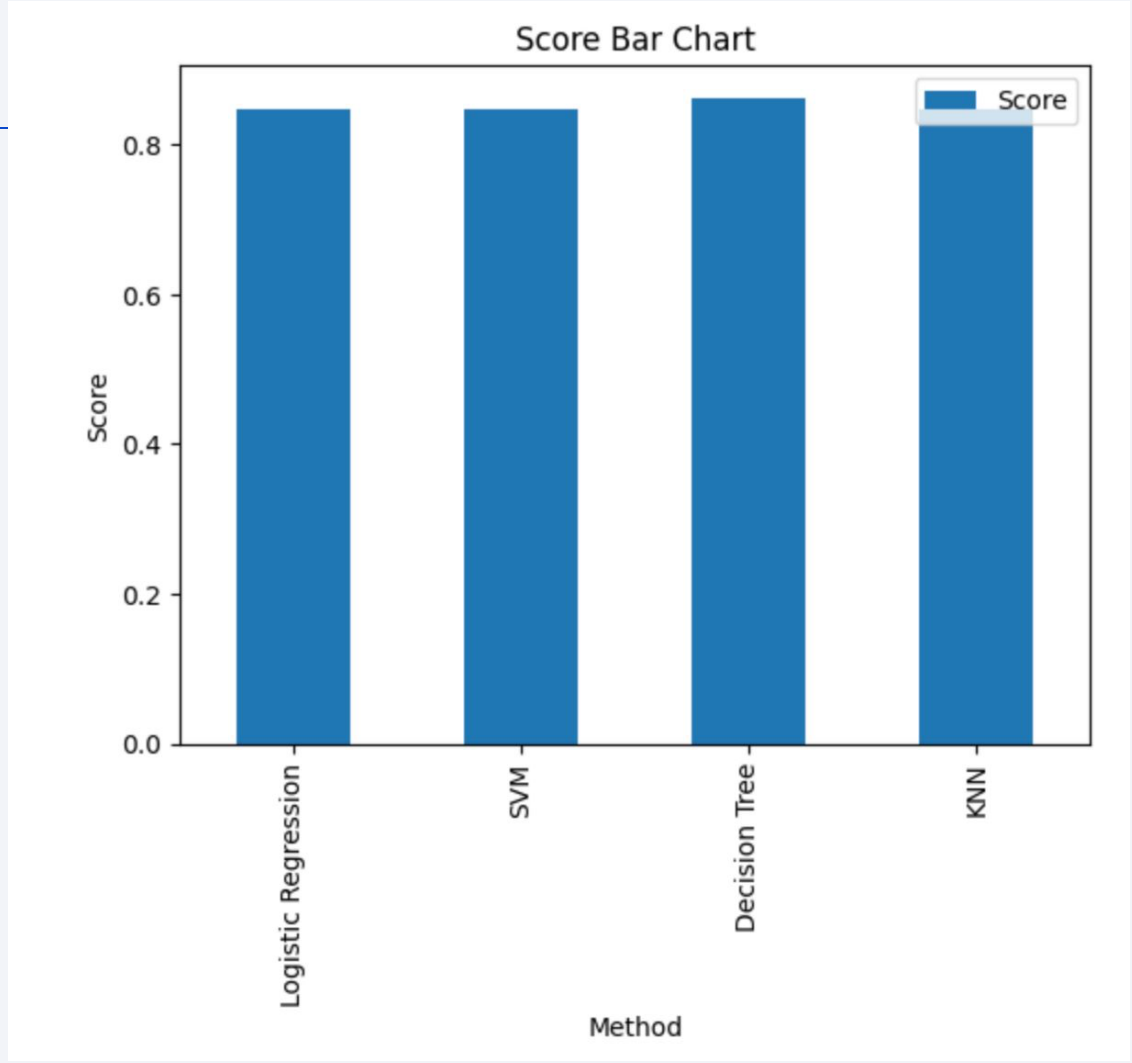
For launches with payload mass over 7,000 kgs, I found a few launches, and equally distributed between success and failure. The most frequent Booster Version Category in this range is B4.

Section 5

Predictive Analysis (Classification)

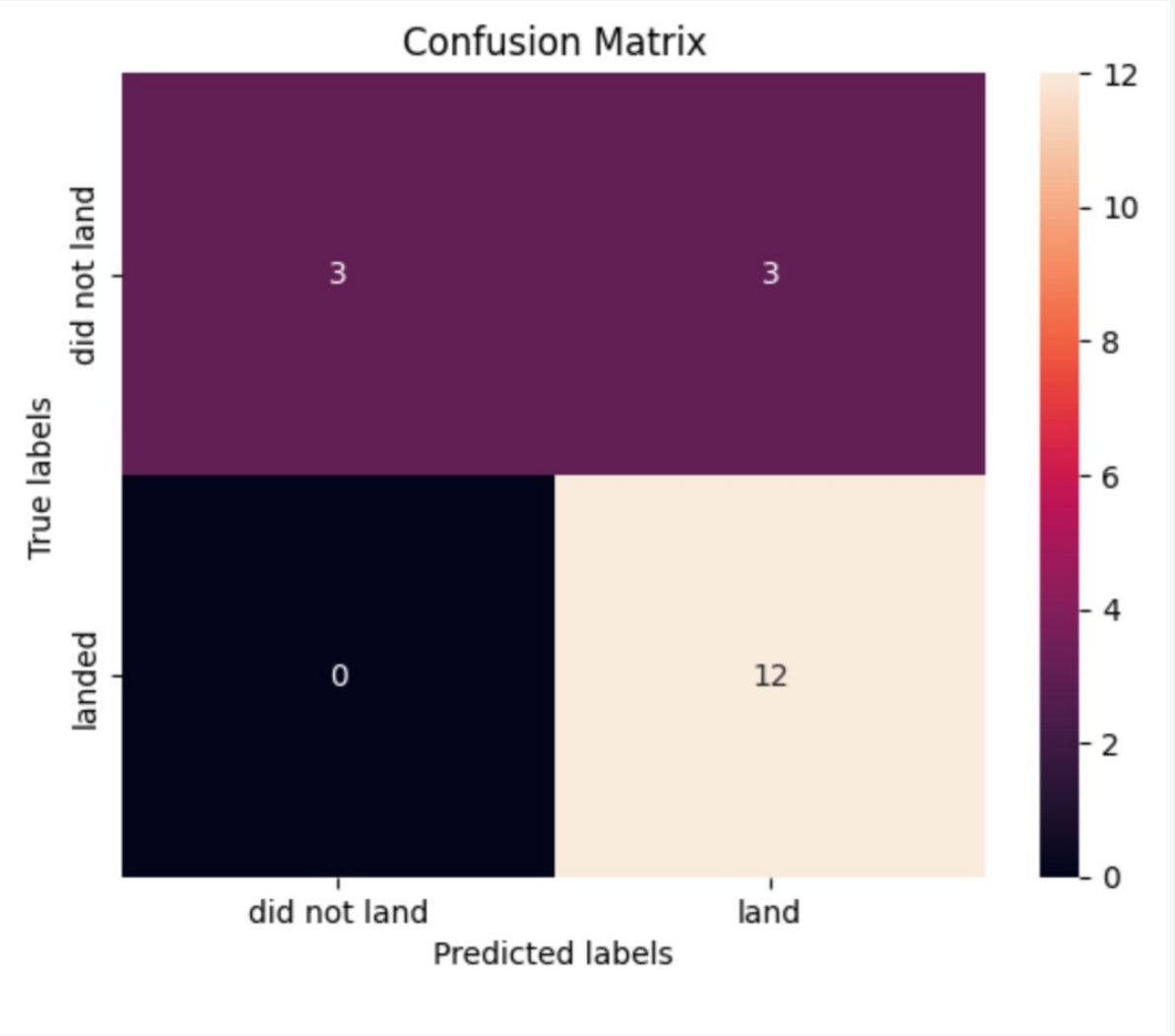
Classification Accuracy

- In this bar chart we can see the accuracy for every classification method.
- The best model according to accuracy is Decision Tree, with an accuracy = 0.86



Confusion Matrix

- Here we can see the confusion matrix for Decision Tree classification algorithm.
- There were 12 landed launches. All of them were classified as landed by the algorithm.
- There were 6 not landed launches. Half of them were classified as landed (false positives) and the other half as not landed.



Conclusions

- At the beginning, most of the flights were launched from CCAFS SLC 40 site. In the middle of the period, the other two sites were the most used launch sites, and in the end, again CCAFS SLC 40 was the most used site.
- CCAFS SLC 40 site was used to launch rockets of all sizes. VAFB SLC is often used to launch small crafts and KSC LC 39A is the site of choice for larger rockets.
- There are four orbits with perfect success rate: ES-LI, GEO, HEO and SSO.Only one orbit never participated on a successful landing: SO. Most of the orbits has an intermediate success rate, between 0.5 and 0.7
- First 60 flights went mostly to only four orbits: LEO, ISS, PO and GTO. After flight no. 60, other orbit types were included in flights planning
- The largest aircrafts went to VLEO orbit.
- Until year 2013, there were no successful landings. After that year, success rate keeps growing every year, with an exception in year 2018, with success rate dropping to 0.6
- Almost every launch site included on my analysis is located near the Atlantic coast in the state of Florida. The only one that is not located in the eastern coast is located in California.

Conclusions

- 46 launches were made in Florida. 33 of them were performed in two sites that are close to each other and the other 13 launches were made in a site that is located some miles away from the other aforementioned launch sites.
- Florida's launch sites are close to coastline, railways and highways. The nearest highway is closer to launch site than the nearest railway. The site is close to coastline and far from any important city.
- 57% of the launches ended with a successful landing. The site with the highest success rate is KSC LC-39A (76.9%)
- For launches with payload mass under 3,000 kgs or over 7,000 kgs, the success/unsuccess ratio is near 50/50. or launches with payload mass between 3,000 and 7,000 kgs, unsuccessful landings are more frequent than successful ones. The Booster Version Category with more successful landings is FT for payload mass under 7,000 kgs and B4 for payload mass over 7,000 kgs.
- The best model to predict landing success, according to accuracy is Decision Tree, with an accuracy = 0.86. There were 12 landed launches. All of them were classified as landed by the algorithm. There were 6 not landed launches. Half of them were classified as landed (false positives) and the other half as not landed.

Appendix 1: Data Collection API scripts

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

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

```
response.status_code
```

```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

```
# 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 si
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)]
```

```
# Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

```
BoosterVersion[0:5]
```

```
['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

we can apply the rest of the functions here:

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_df[launch_df['BoosterVersion']!='Falcon 1']
```

Now that we have removed some values we should reset the FlgihtNumber column

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Appendix 2: Data Collection with Web Scraping

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url and headers
# assign the response to a object
r = requests.get(static_url,headers = headers)
```

Create a `BeautifulSoup` object from the HTML `response`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
bs = BeautifulSoup(r.text,'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute
bs.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches – Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = bs.find_all(name = 'table')
```

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
extracted_row = 0
for table_number,table in enumerate(bs.find_all('table',"wikitable plainrowheaders collapsible")):
    for rows in table.find_all("tr"):
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        row=rows.find_all('td')
        if flag:
            extracted_row += 1
            launch_dict['Flight No.'].append(flight_number)
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            launch_dict['Version Booster'].append(bv)
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site)
            payload = row[3].a.string
            launch_dict['Payload'].append(payload)
            payload_mass = get_mass(row[4])
            launch_dict['Payload mass'].append(payload_mass)
            orbit = row[5].a.string
            launch_dict['Orbit'].append(orbit)
            customer = row[6].a.string
            launch_dict['Customer'].append(customer)
            launch_outcome = list(row[7].strings)[0]
            launch_dict['Launch outcome'].append(launch_outcome)
            booster_landing = landing_status(row[8])
            launch_dict['Booster landing'].append(booster_landing)
```

Appendix 3: Data Wrangling scripts

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

```
LaunchSite
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: count, dtype: int64
```

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

Note: Do not count GTO, as it is a transfer orbit and not itself geostationary.

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

```
Orbit
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
HEO       1
ES-L1     1
SO        1
GEO       1
Name: count, dtype: int64
```

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome` , create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome` ; otherwise, it's one. Then assign it to the varia `landing_class` :

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for i in df['Outcome']:
    if i in bad_outcomes:
        landing_class.append(0)
    else: landing_class.append(1)
landing_class
```

TASK 3: Calculate the number and occurence of mission outcome of the orbits

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

```
Outcome
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: count, dtype: int64
```

`True Ocean` means the mission outcome was successfully landed to a specific region of the ocean while `False Ocean` means the mission outcome was unsuccessfully landed to a specific region of the ocean. `True RTLS` means the mission outcome was successfully landed to a ground pad `False RTLS` means the mission outcome was unsuccessfully landed to a ground pad. `True ASDS` means the mission outcome was successfully landed to a drone ship `False ASDS` means the mission outcome was unsuccessfully landed to a drone ship. `None ASDS` and `None None` these represent a failure to land.

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

We create a set of outcomes where the second stage did not land successfully:

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

Appendix 4: EDA with SQL

Task 1

Display the names of the unique launch sites in the space mission

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

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

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

Task 2

Display 5 records where launch sites begin with the string 'CCA'

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

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

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

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

sum(PAYLOAD_MASS__KG_)
45596

Task 4

Display average payload mass carried by booster version F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where "Booster_Version" like "F9 v1.
```

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

avg(PAYLOAD_MASS__KG_)
2534.6666666666665

Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

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

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

min(Date)
2015-12-22

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
%sql select "Payload" from SPACEXTABLE where "Landing_Outcome" = "Success (drone ship)" and PAYLOAD_MASS__KG_ between 4000 and 6000
```

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

Payload
JCSAT-14
JCSAT-16
SES-10
SES-11 / EchoStar 105

Task 7

List the total number of successful and failure mission outcomes

```
%sql select "Mission_Outcome", count(Payload) from SPACEXTABLE group by "Mission_Outcome"
```

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

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

Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

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

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

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

Task 9

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%sql select substr(Date,6,2),"Landing_Outcome","Booster_Version","Launch_Site" from SPACEXTABLE where substr(Date,0,5)="2015" and "Landing_Outcome"
```

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

substr(Date,6,2)	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

Task 10

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(Landing_Outcome) as Cuenta from (select * from SPACEXTABLE where Date between "2010-06-04" and "2017-03-20") gr
```

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

Landing_Outcome	Cuenta
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

Appendix 5: EDA with Visualizations (scripts)

TASK 1: Visualize the relationship between Flight Number and Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

TASK 2: Visualize the relationship between Payload Mass and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

TASK 3: Visualize the relationship between success rate of each orbit type

Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the sucess rate of each orbit

```
# HINT use groupby method on Orbit column and get the mean of Class column
orbit_means = []
orbit_means = pd.DataFrame(df.groupby('Orbit')["Class"].mean())
sns.barplot(y="Class", x="Orbit", data=orbit_means)
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```

TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

TASK 5: Visualize the relationship between Payload Mass and Orbit type

Similarly, we can plot the Payload Mass vs. Orbit scatter point charts to reveal the relationship between Payload Mass and Orbit type

```
# Plot a scatter point chart with x axis to be Payload Mass and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload Mass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

TASK 6: Visualize the launch success yearly trend

You can plot a line chart with x axis to be `Year` and y axis to be average success rate, to get the average launch success trend.

The function will help you get the year from the date:

```
# A function to Extract years from the date
year=[]
def Extract_year():
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
Extract_year()
df['Date'] = year
df.head()
```

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
year_means = []
year_means = pd.DataFrame(df.groupby('Date')["Class"].mean())

sns.lineplot(y="Class", x="Date", data=year_means)
plt.xlabel("Year",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```

TASK 7: Create dummy variables to categorical columns

Use the function `get_dummies` and `features` dataframe to apply OneHotEncoder to the column `Orbits`, `LaunchSite`, `LandingPad`, and `Serial`. Assign the value to the variable `features_one_hot`, display the results using the method head. Your result dataframe must include all features including the encoded ones.

```
# HINT: Use get_dummies() function on the categorical columns
features_one_hot = pd.get_dummies(features[["Orbit","LaunchSite","LandingPad","Serial"]], dtype=int)
features_one_hot = pd.concat([features_one_hot,features], axis = 1)
features_one_hot = features_one_hot.drop(["Orbit","LaunchSite","LandingPad","Serial"], axis = 1)
```

TASK 8: Cast all numeric columns to `float64`

Now that our `features_one_hot` dataframe only contains numbers, cast the entire dataframe to variable type `float64`

```
# HINT: use astype function
features_one_hot.astype("float64")
```

Thank you!