IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Caroline Atienza
18/07/2024

Outline

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

Executive Summary

- 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.
  - Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- Summary of methodologies
  - The data science methodologies and Python have been used:
    - Data collection
    - Data wrangling
    - Exploratory data analysis
    - Visual analytics
    - Predictive analysis

- Summary of all results
  - With Exploratory data analysis, a lot of features have been analysed. For example, the launch success rate keeps increasing since 2013 to 2020
  - Interactive analytics demo with folium and dash. For example, the launch site are near coasts
  - Predictive analysis with the results in a confusion matrix and accuracy of more than 90%

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.
  - Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.
  - This project is using Data Science to find answers
- Problems you want to find answers
  - We want to determine the cost of a launch, by determining if the first stage will land

Section 1

Methodology

Methodology

Executive Summary

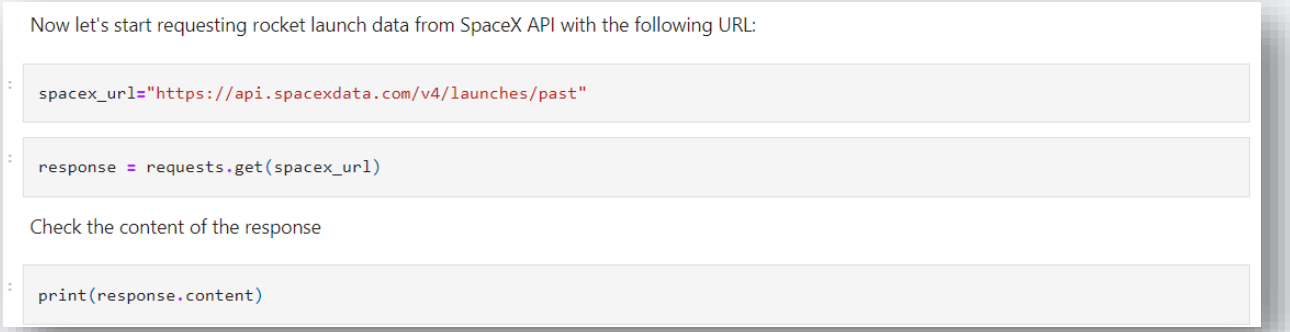
- Data collection methodology:
  - Data about rocket launch was collected from SpaceX API
  - Data about Falcon 9 historical launch records was collected from Wikipedia with Webscraping
- Perform data wrangling
  - Data was processed through data wrangling in Python, with Numpy and Pandas
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Classification models have been built, tuned, evaluated

Data Collection

- The first data collection has been made with SpaceX REST calls using Pandas, Numpy and Requests
- The second data collection, about Falcon 9 historical launch records, has been made from Wikipedia with Webscraping

Data Collection – SpaceX API

- The data collection has been made with SpaceX REST calls using Pandas, Numpy and Requests
- The GitHub URL of the completed SpaceX API calls notebook is
- https://github.com/caroat/Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

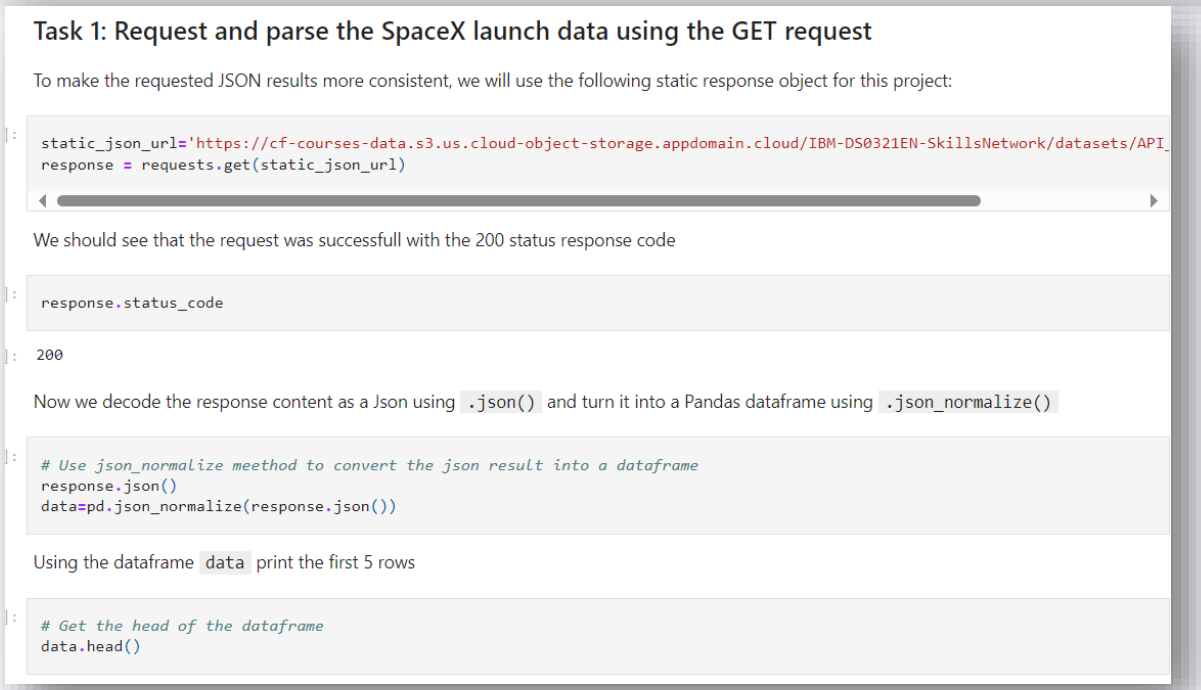

Now let's start requesting rocket launch data from SpaceX API with the following URL:

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

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

Check the content of the response

```
print(response.content)
```


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

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

```
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()`

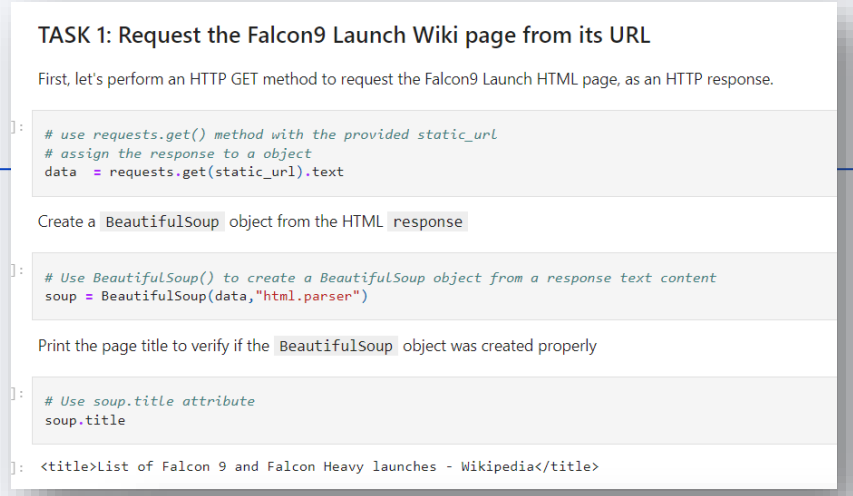
```
# Use json_normalize meethod to convert the json result into a dataframe
response.json()
data=pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

Data Collection - Scraping

- Data about Falcon 9 historical launch records was collected from Wikipedia with Webscraping and BeautifulSoup
  - Extract a Falcon 9 launch records HTML table from Wikipedia
  - Parse the table and convert it into a Pandas data frame
- The GitHub URL of the completed SpaceX API calls notebook is
- https://github.com/caroat/Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb



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
# assign the response to a object
data  = requests.get(static_url).text
```

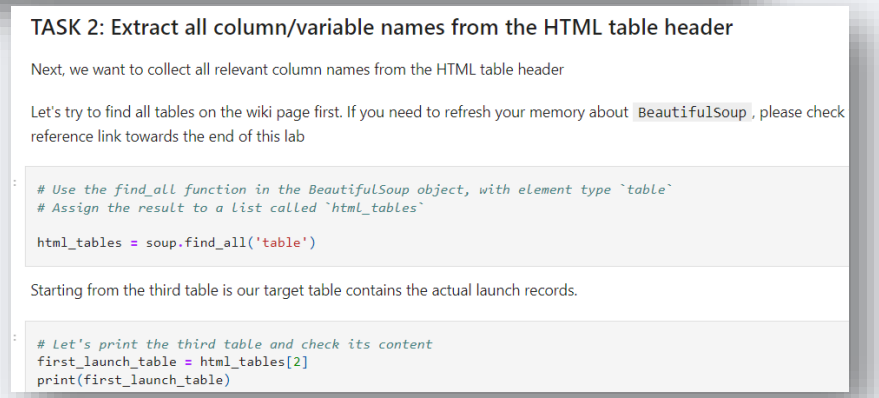
Create a `BeautifulSoup` object from the HTML `response`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data,"html.parser")
```

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

```
# Use soup.title attribute
soup.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

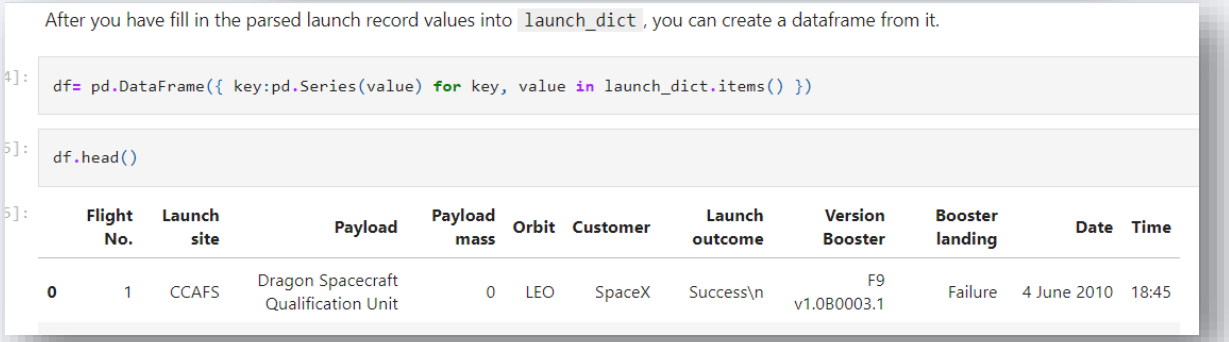
Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check reference link towards the end of this lab

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`

html_tables = soup.find_all('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)
```



After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

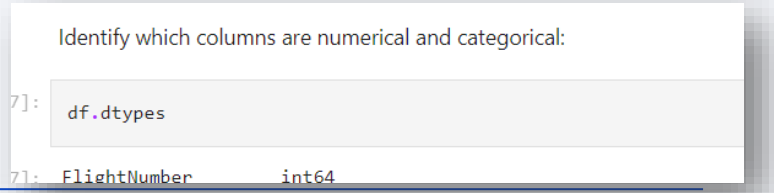
```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

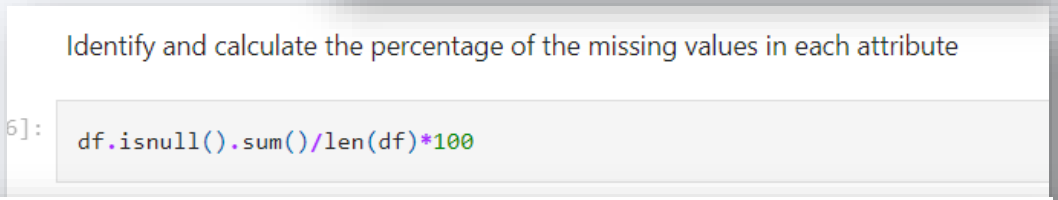
```
df.head()
```

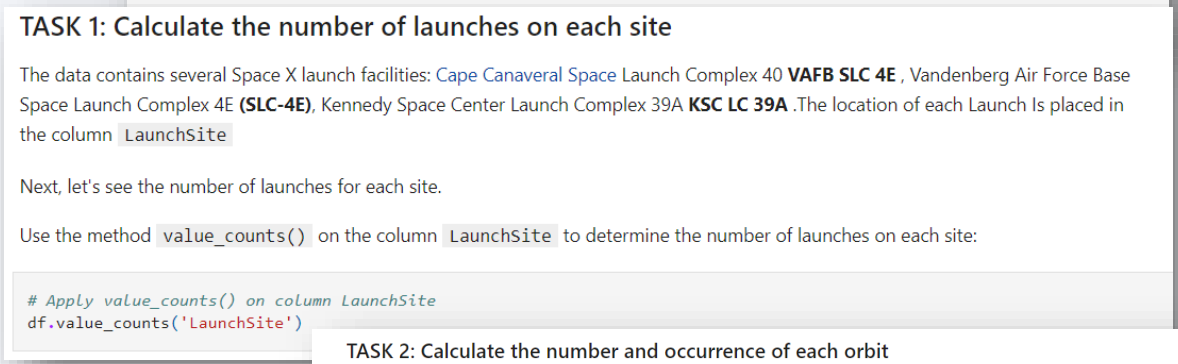
	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45

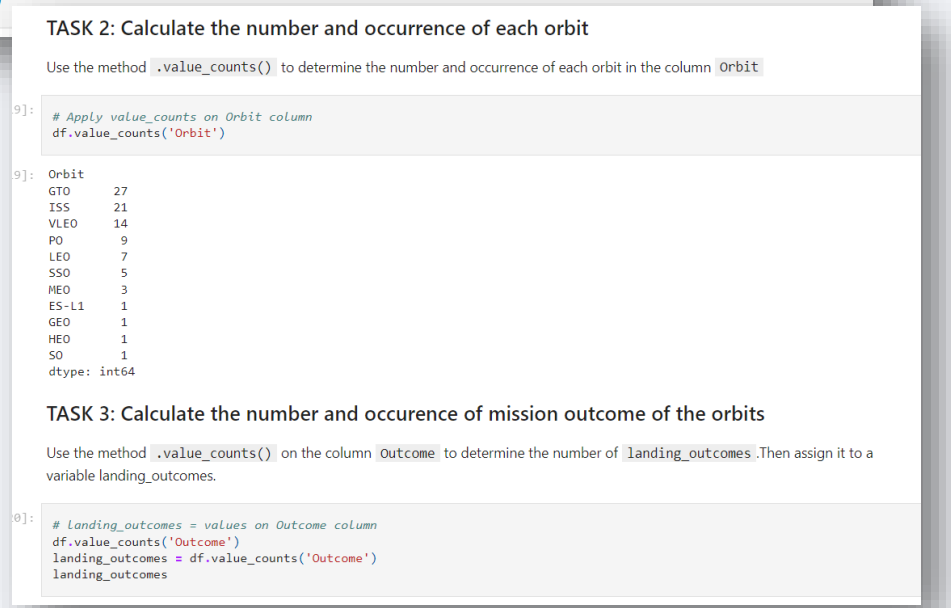
Data Wrangling

- With the Space X dataset obtained from last section, data has been analysed
  - Calculate the number of launches on each site
  - Calculate the number and occurrence of each orbit
  - Calculate the number and occurence of mission outcome of the orbits
  - Create a landing outcome label from Outcome column
- The GitHub URL of the completed data wrangling notebook is
- https://github.com/caroat/Applied-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb









EDA with Data Visualization

- Exploratory Data Analysis (EDA) and Feature Engineering has been done, using Pandas and Matplotlib

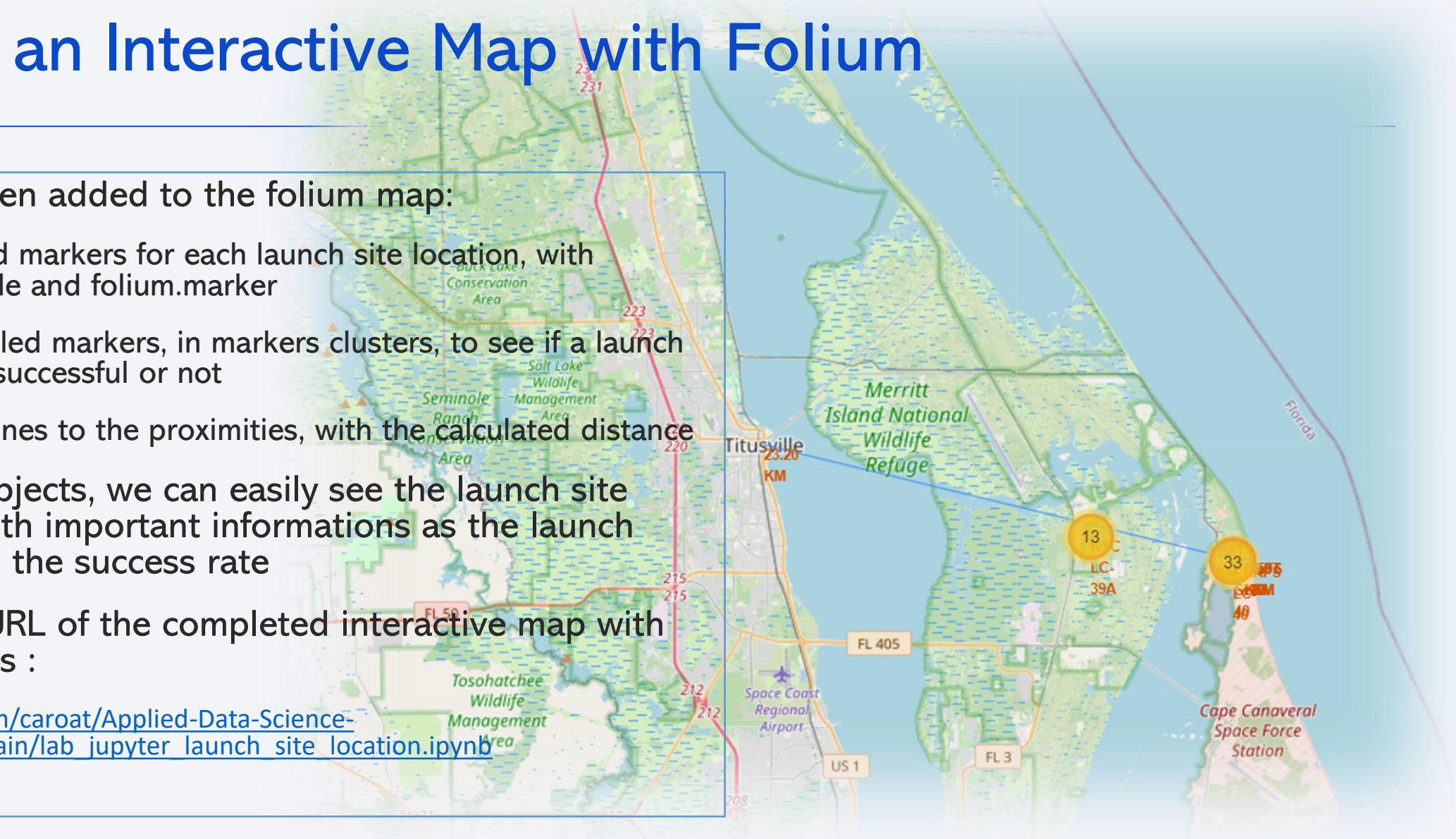
- The charts plotted are for visualizing the impacts of different features on the launch success rate:
  - Scatter charts for visualizing the impacts of : Payload mass, Flight number, launch site, orbit, and the impact of the relationship between these features
  - Bar charts, line charts for visualizing the impacts of : Orbit, Date
- The delailed charts are in section 2 of this report
- The GitHub URL of the completed data visualization notebook is
- https://github.com/caroat/Applied-Data-Science-Capstone/blob/main/edadataviz.ipynb

EDA with SQL

- The SQL queries performed are:
  - **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. Use a subquery**
  - **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.**
- The GitHub URL of the completed EDA with SQL notebook is :
- https://github.com/caroat/Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb

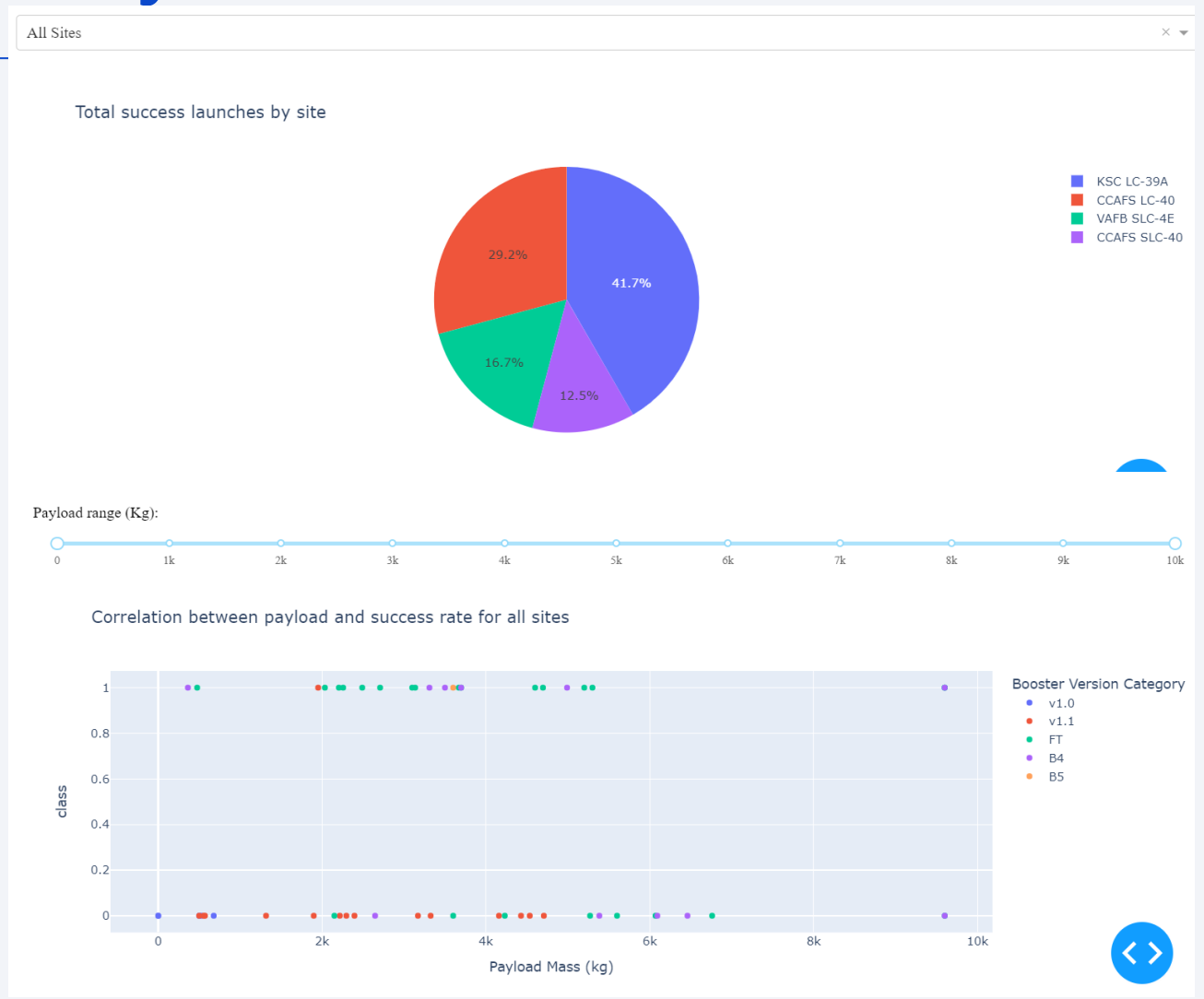
Build an Interactive Map with Folium

- There has been added to the folium map:
  - Circles and markers for each launch site location, with folium.circle and folium.marker
  - Color-labeled markers, in markers clusters, to see if a launch has been successful or not
  - Distance lines to the proximities, with the calculated distance
- With these objects, we can easily see the launch site locations , with important informations as the launch numbers and the success rate
- The GitHub URL of the completed interactive map with Folium map is :
- https://github.com/caroat/Applied-Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb



Build a Dashboard with Plotly Dash

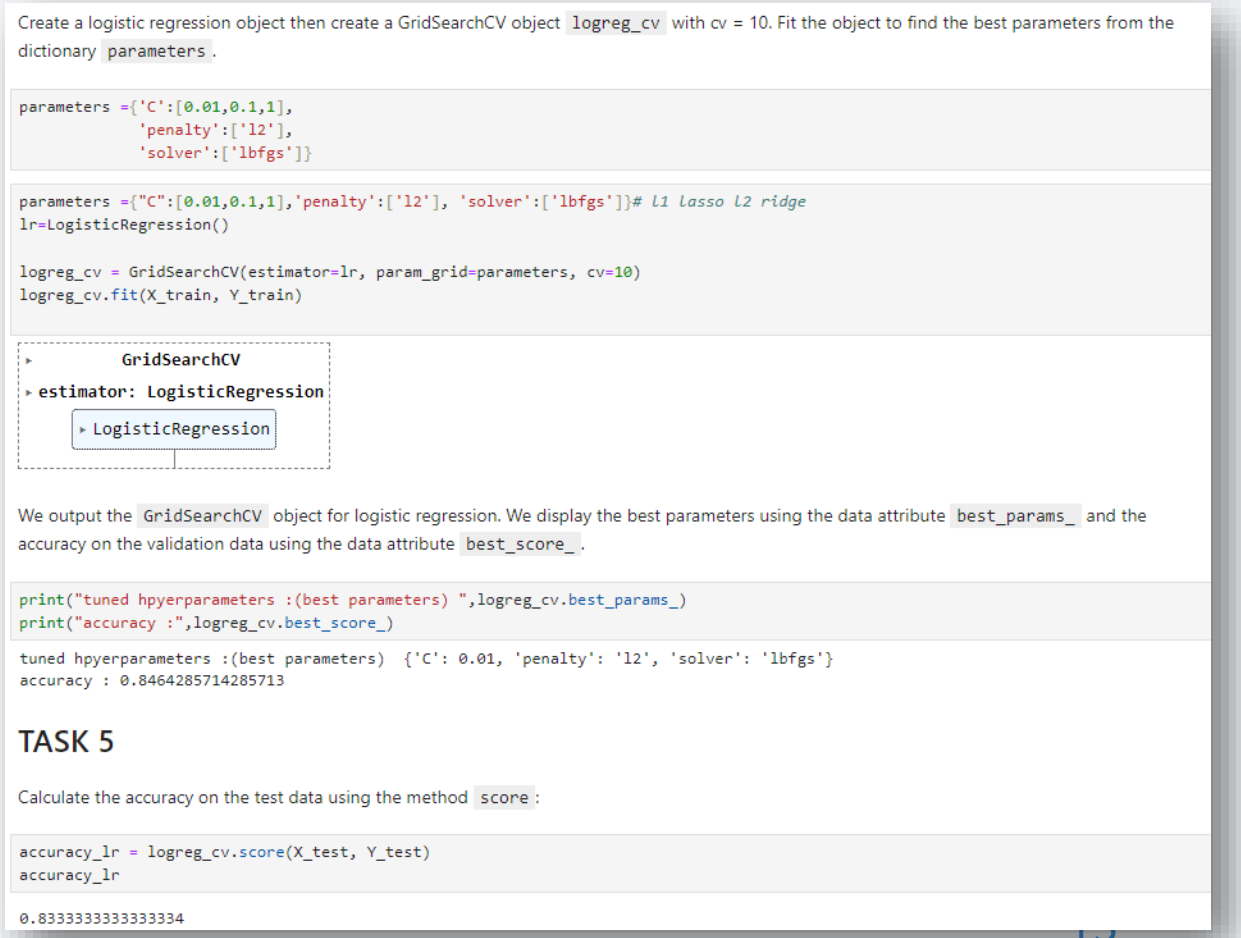
- With this dashboard, the users can perform interactive visual analytics on SpaceX launch data in real-time.
- It contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart.
- We can obtain some insights about the success rate , for example: Which site has the largest successful launches
- The GitHub URL of the completed Plotly Dash lab is:
- https://github.com/caroat/Applied-Data-Science-Capstone/blob/main/Dash%20capstone%20(1).ipynb



Predictive Analysis (Classification)

- A machine learning pipeline has been built, to predict if the first stage will land.
- The processus was : Perform exploratory Data Analysis and determine Training Labels
  - create a column for the class
  - Standardize the data
  - Split into training data and test data
  - Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
  - Find the method performs best using test data, with accuracy comparison
- The GitHub URL of the completed predictive analysis lab is:
- https://github.com/caroat/Applied-Data-Science-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Example with logistic regression



Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()

logreg_cv = GridSearchCV(estimator=lr, param_grid=parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```

GridSearchCV
estimator: LogisticRegression
LogisticRegression

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

TASK 5

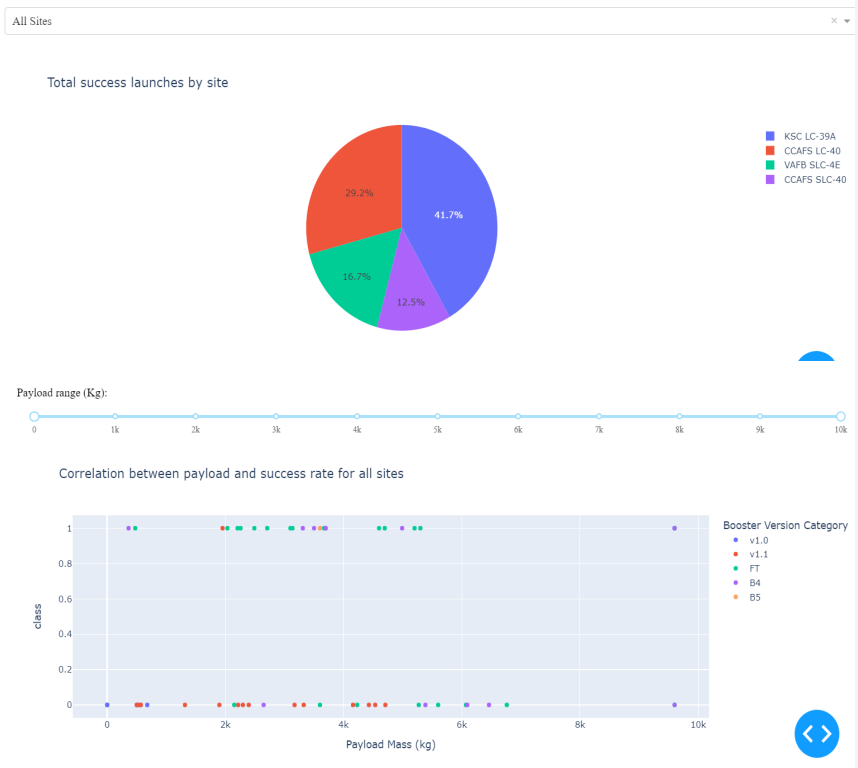
Calculate the accuracy on the test data using the method `score`:

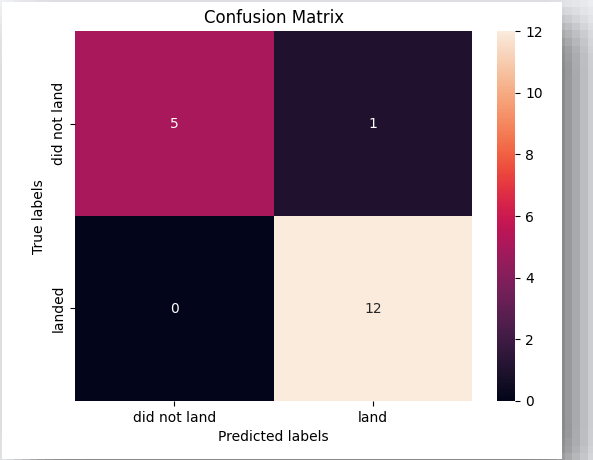
```
accuracy_lr = logreg_cv.score(X_test, Y_test)
accuracy_lr
```

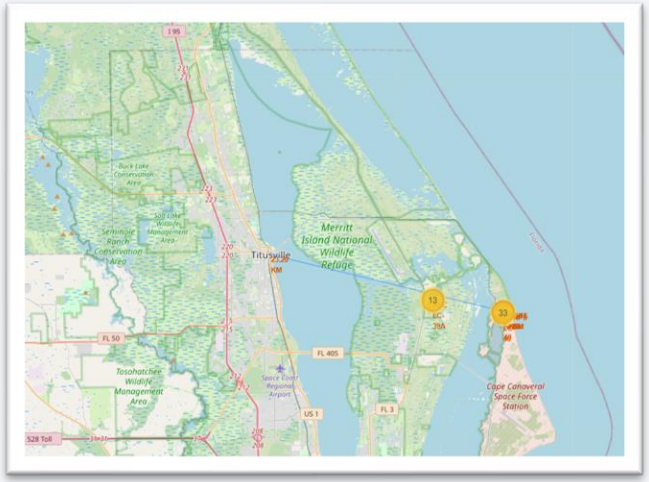
```
0.8333333333333334
```

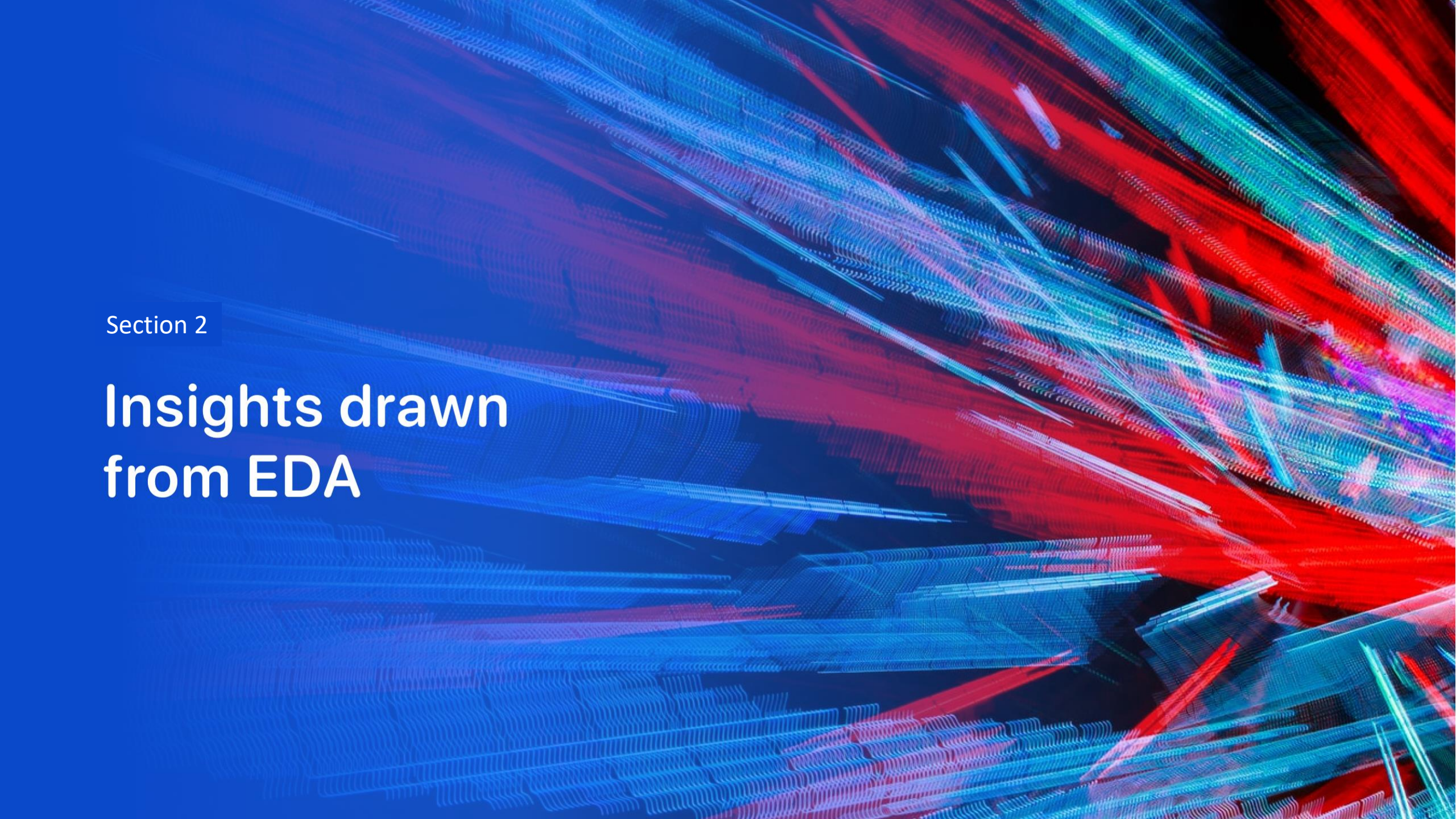
Results

- With Exploratory data analysis, a lot of features have been analysed. For example:
  - the total number of successful mission outcomes is 100, where the total number of failure mission outcomes is 1
  - The launch success rate keeps increasing since 2013 to 2020
- Interactive analytics demo with folium and dash
- Predictive analysis with the results in a confusion matrix



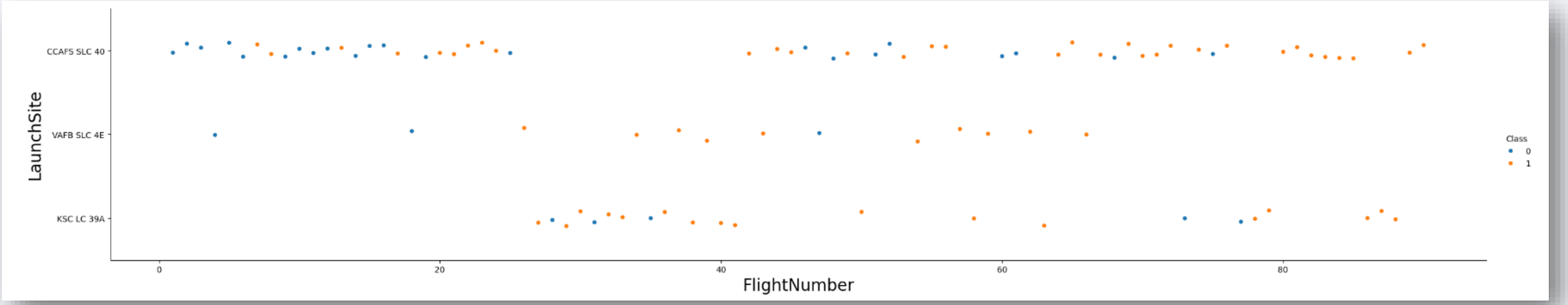





Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

Scatter plot of Flight Number vs. Launch Site

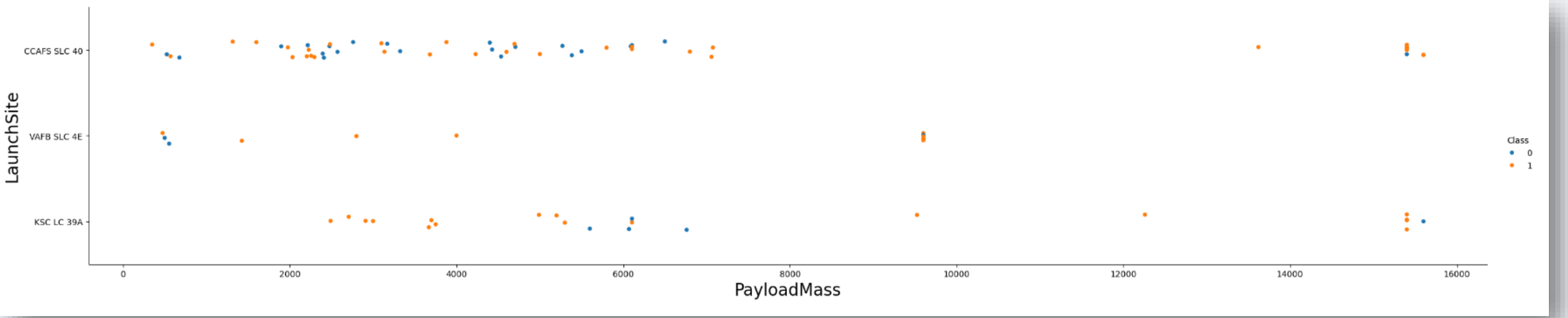


Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

```
# 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("FlightNumber",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```

Payload vs. Launch Site

Scatter plot of Payload vs. Launch Site



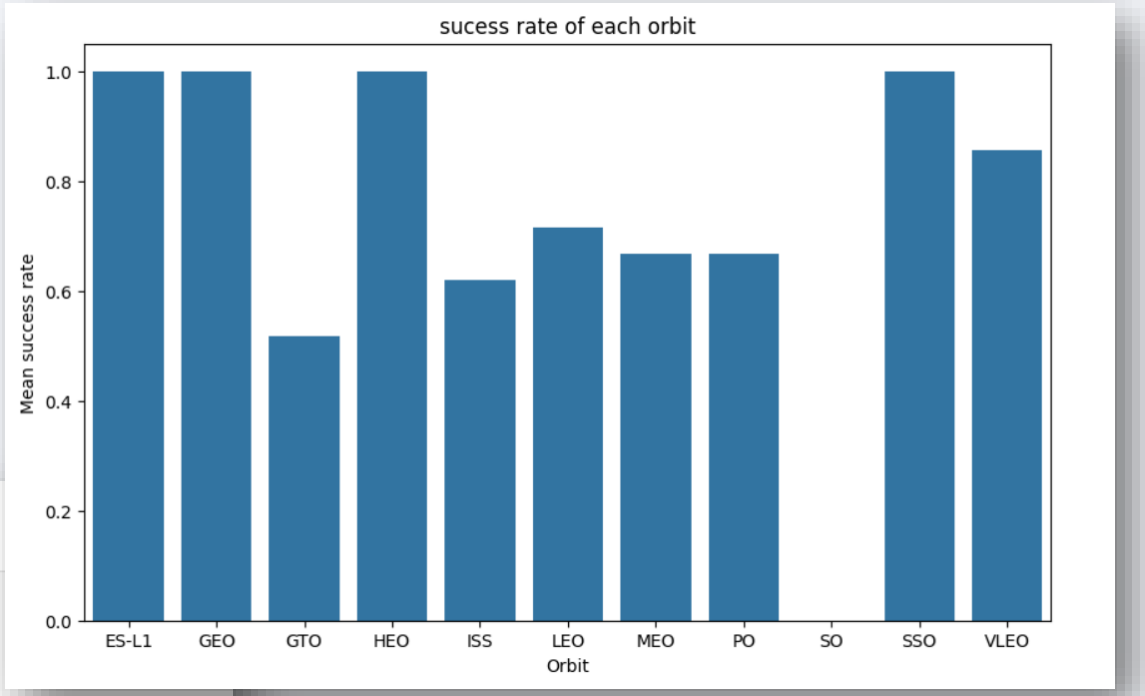
We also want to observe if there is any relationship between launch sites and their payload mass.

```
# 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("PayloadMass",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```

Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

Bar chart for the success rate of each orbit type



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

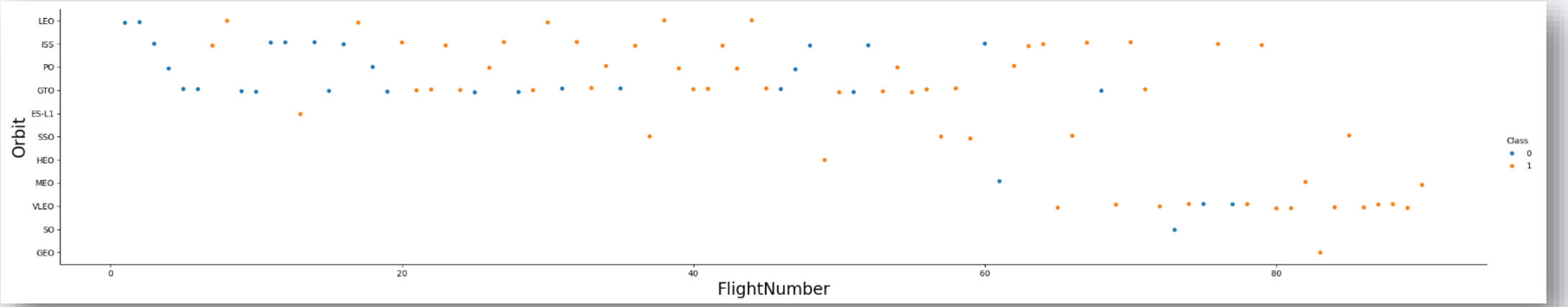
```
# HINT use groupby method on Orbit column and get the mean of Class column

df_orbit = df.groupby(df['Orbit'])['Class'].mean().reset_index()

plt.figure(figsize=(10,6))
sns.barplot(x='Orbit', y='Class',  data=df_orbit)
plt.xlabel('Orbit')
plt.ylabel('Mean success rate')
plt.title('sucess rate of each orbit')
plt.show()
```

Flight Number vs. Orbit Type

Scatter point of Flight number vs. Orbit type



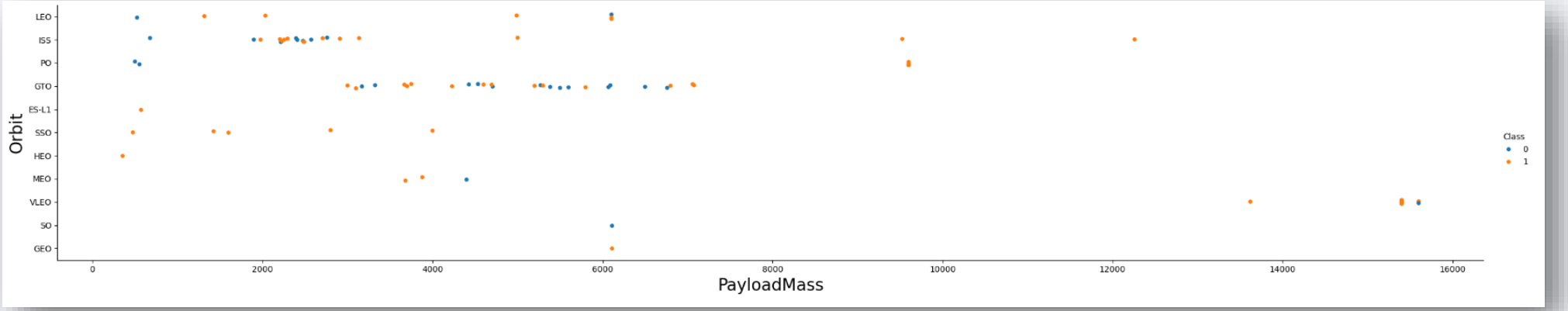
For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

```
# 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("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type

Scatter point of payload vs. orbit type



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

```
# Plot a scatter point chart with x axis to be Payload 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("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

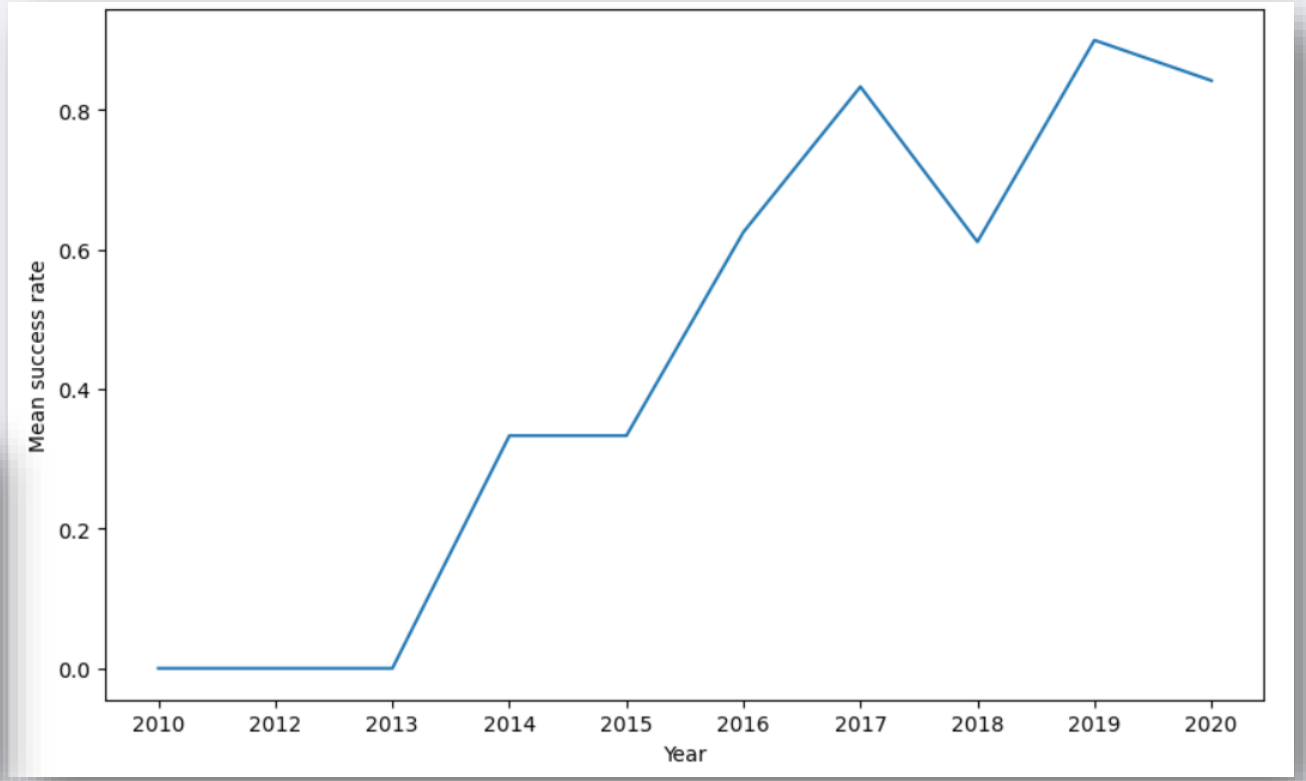
Launch Success Yearly Trend

Line chart of yearly average success rate

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate


df_orbit = df.groupby(df['Date'])['Class'].mean().reset_index()

plt.figure(figsize=(10,6))
sns.lineplot(x='Date', y='Class',  data=df_orbit)
plt.xlabel('Year')
plt.ylabel('Mean success rate')
plt.title('sucess rate of each year')
plt.show()
```



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

- The unique launch sites names are:
  - CCAFS LC-40
  - CCAFS SLC-40
  - KSC LC-39A
  - VAFB SLC-4E
- The query used a group by

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

```
%sql select Launch_Site from SPACEXTABLE \
    group by Launch_Site
```

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

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

Launch Site Names Begin with 'CCA'

- 5 records where launch sites begin with `CCA` are shown
- The query used: “like” and “limit”

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

Total Payload Mass

- The total payload carried by boosters from NASA (CRS) is 45596kg
- The query used “SUM” and “LIKE”

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

```
%sql select Customer, sum(PAYLOAD_MASS__KG_) from SPACEXTABLE \
    where Customer like "NASA (CRS)"
```

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

Customer	sum(PAYLOAD_MASS__KG_)
NASA (CRS)	45596

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 is 2928.4kg
- The query used “AVG” and “LIKE”

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

```
%sql select Booster_Version, avg(PAYLOAD_MASS__KG_) from SPACEXTABLE \
    where Booster_Version like "F9 v1.1"
```

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

Booster_Version	avg(PAYLOAD_MASS__KG_)
F9 v1.1	2928.4

First Successful Ground Landing Date

- The date of the first successful landing outcome on ground pad is 2015/12/22
- First, a query with group by on Landing_Outcome to find the exact name for Successful Ground Landing
- Then a query with the exact name and “MIN”

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

Hint:Use min function

```
%sql select Landing_Outcome from SPACEXTABLE \
    group by Landing_Outcome
```

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

Landing_Outcome
Controlled (ocean)
Failure
Failure (drone ship)
Failure (parachute)
No attempt
No attempt
Precluded (drone ship)
Success
Success (drone ship)
Success (ground pad)
Uncontrolled (ocean)

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

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

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 are:
  - F9 FT B1022
  - F9 FT B1026
  - F9 FT B1021.2
  - F9 FT B1031.2

- The query uses “WHERE” and “BETWEEN”

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 Booster_Version from SPACEXTABLE \
    where Landing_Outcome like "Success (drone ship)" \
    and PAYLOAD_MASS__KG_ between 4000 and 6000
```

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

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

Total Number of Successful and Failure Mission Outcomes

- The total number of successful mission outcomes is 100
- The total number of failure mission outcomes is 1
- The query uses “count(*)” and group by

List the total number of successful and failure mission outcomes

```
%sql select Mission_Outcome, count(*) from SPACEXTABLE \
    group by Mission_Outcome
```

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

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

Boosters Carried Maximum Payload

- The names of the booster which have carried the maximum payload mass are listed in this screenshot
- The query uses a subquery to find the maximum payload mass

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%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

2015 Launch Records

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

- The query uses two “WHERE”

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) as month, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE \
    where substr(Date,0,5)='2015' and Landing_Outcome = "Failure (drone ship)"
```

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

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

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

- The 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 is in the screenshot

- The query uses “ORDER BY” and “DESC”

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 count from SPACEXTABLE \
    where Date between "2010-06-04" and "2017-03-20" \
    group by Landing_Outcome\
    order by count desc
```

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

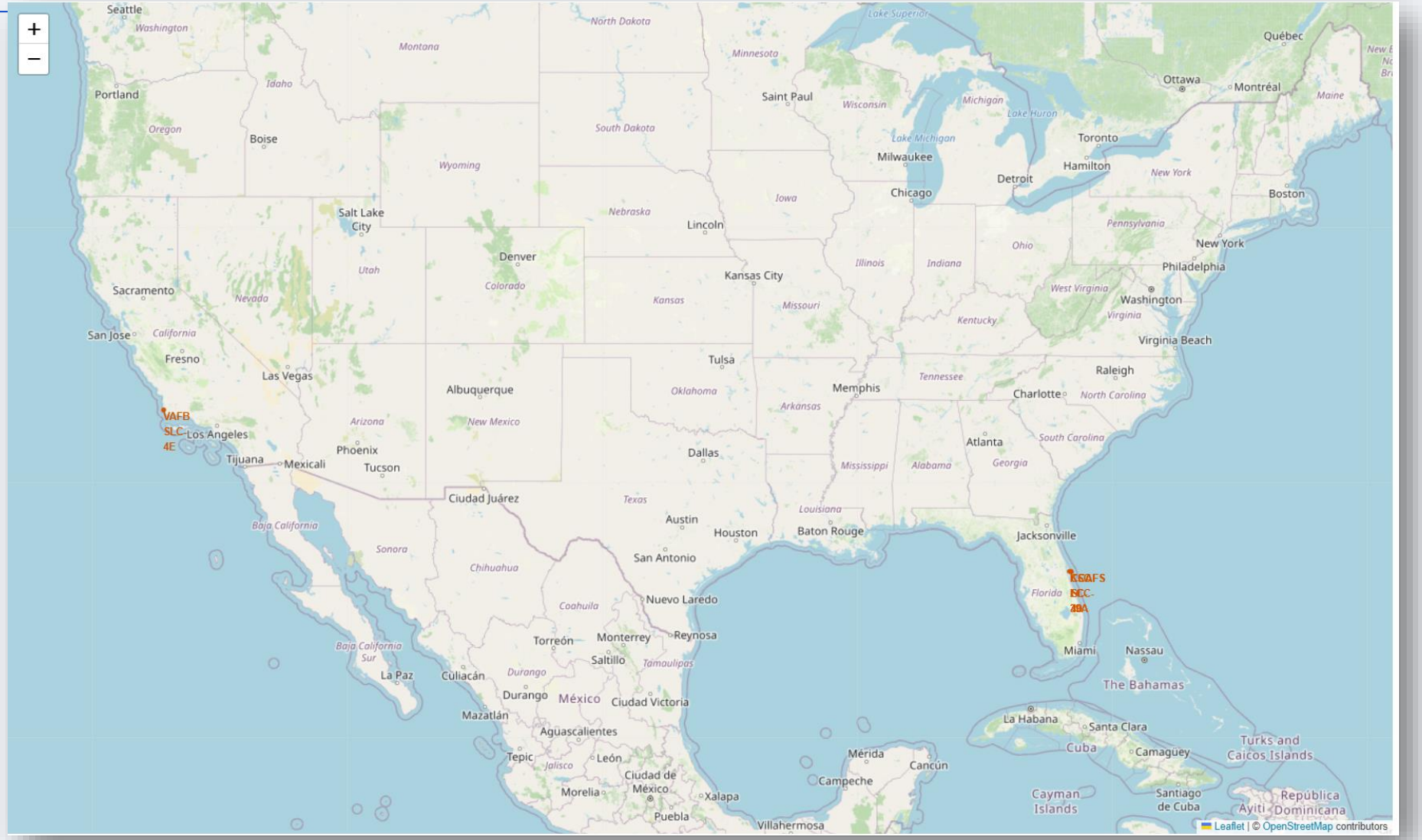
Landing_Outcome	count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

Folium Map : Launch sites' location

- This folium map shows the launch sites' location in US
- We see the locations, the names of the sites and the number of launches done if we zoom
- We can see that there is 2 major places and that they are near the coastline

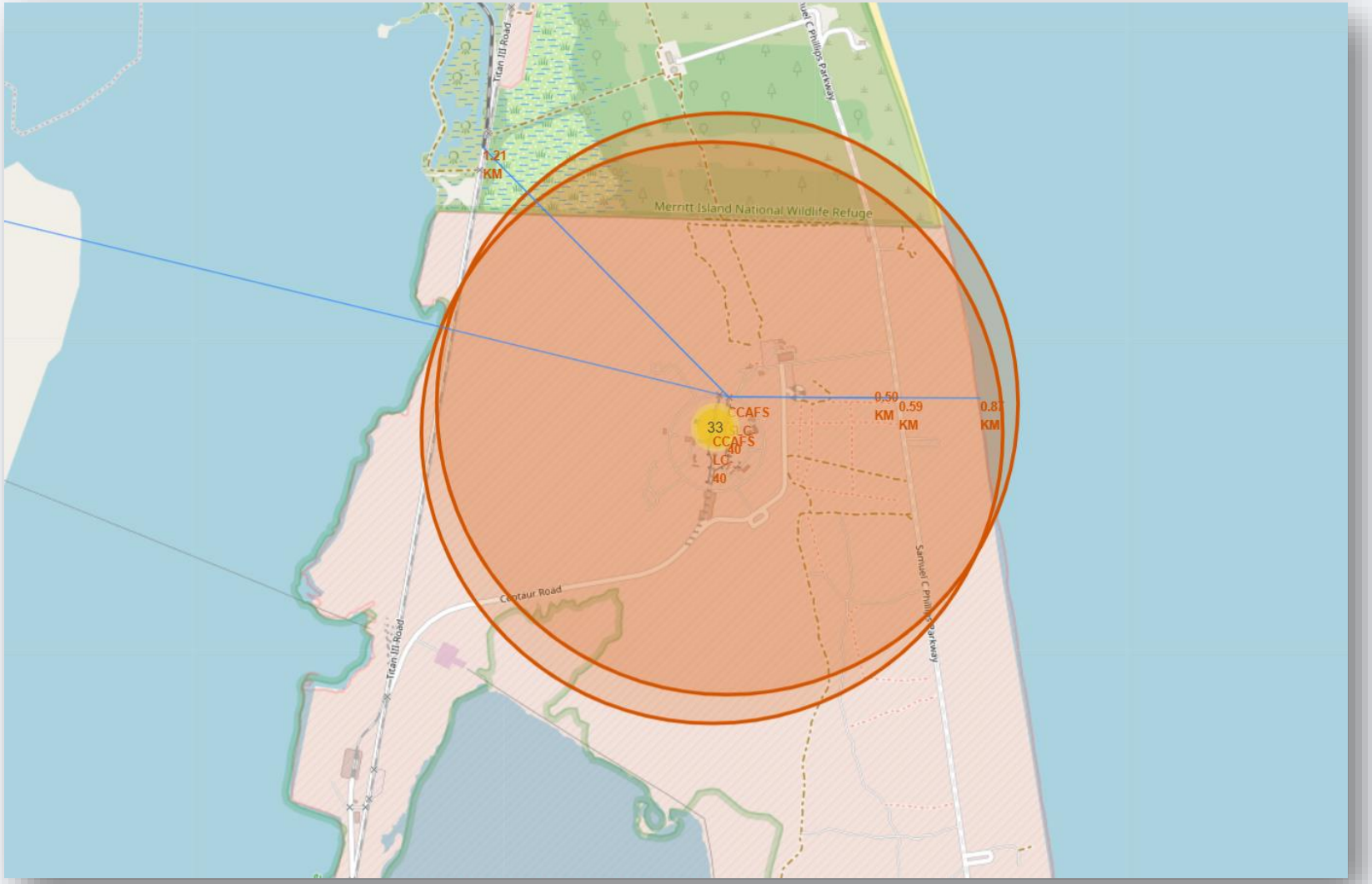


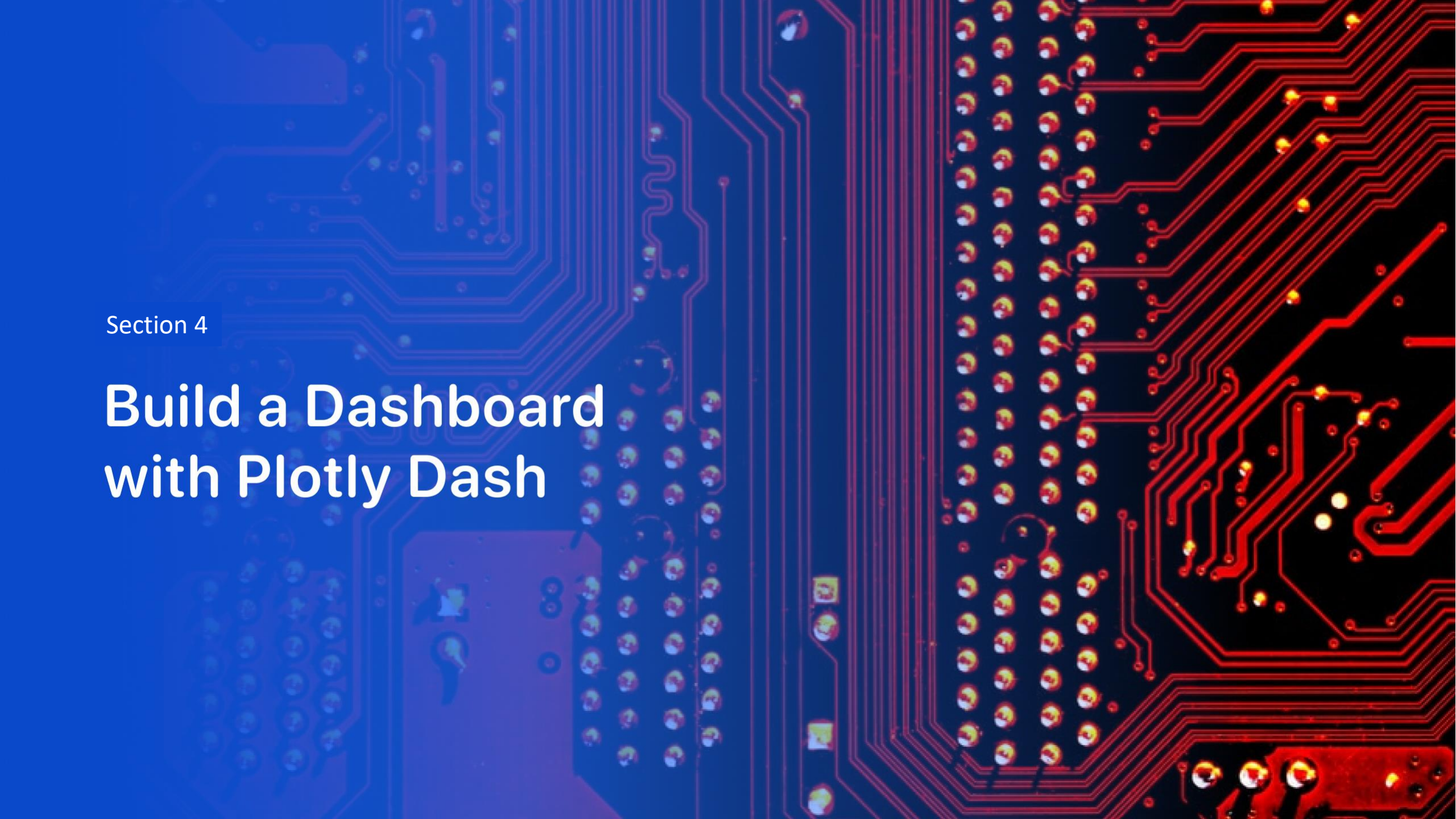
Folium Map : launch outcomes for a site

- This folium map shows the color-labeled launch outcomes of the launch site KSC LC-39A
- In green are successful launches, and in red the unsuccessful ones
- We can see that there is more successful launches

Folium Map : Launch site proximities

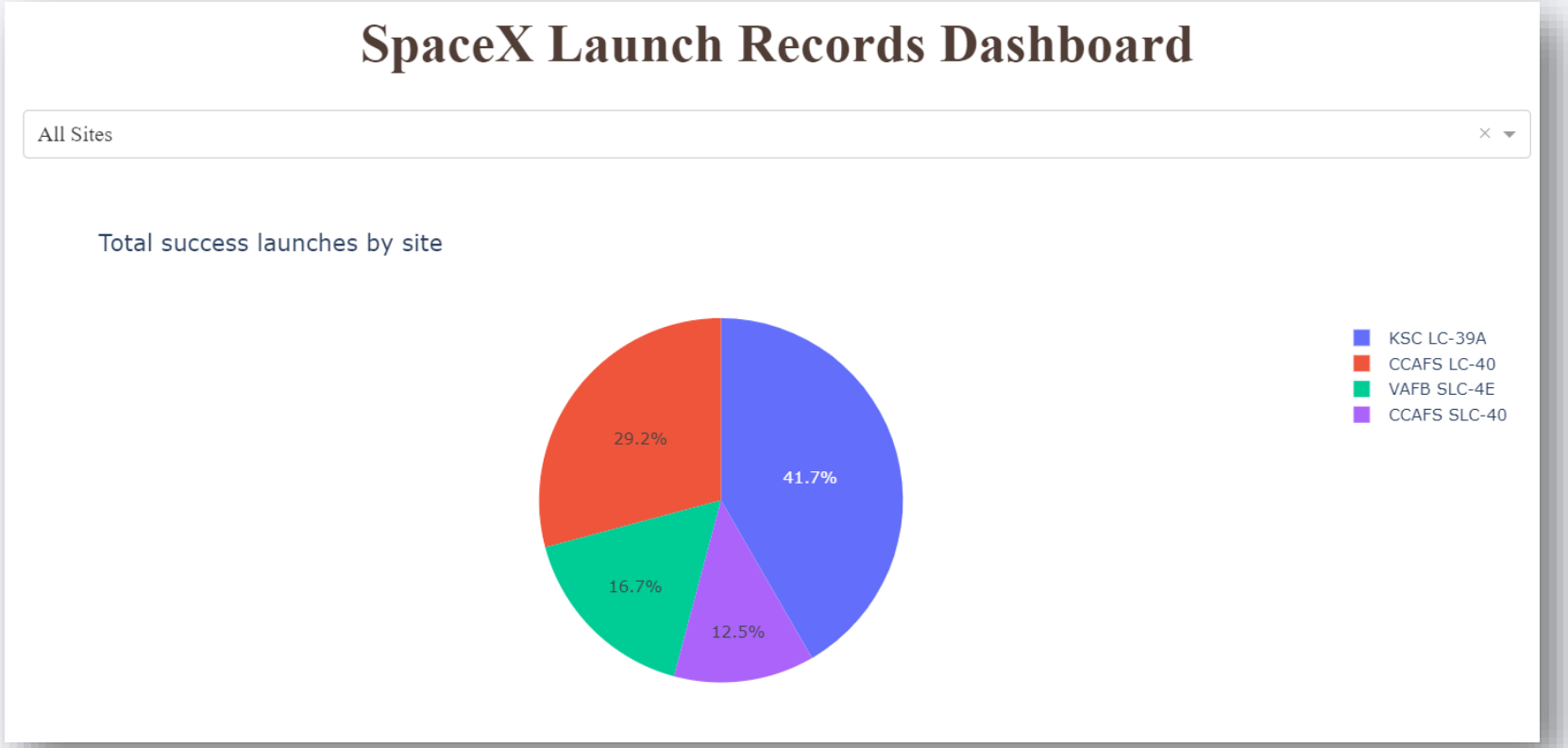
- This folium map shows the proximities of the launch site CCAFS SLC-40
- We can see the lines between the site and the proximities, and the distances are calculated and displayed
- This site is close to a railway, a highway and the coastline. It’s not close to a city




Section 4
Build a Dashboard with Plotly Dash

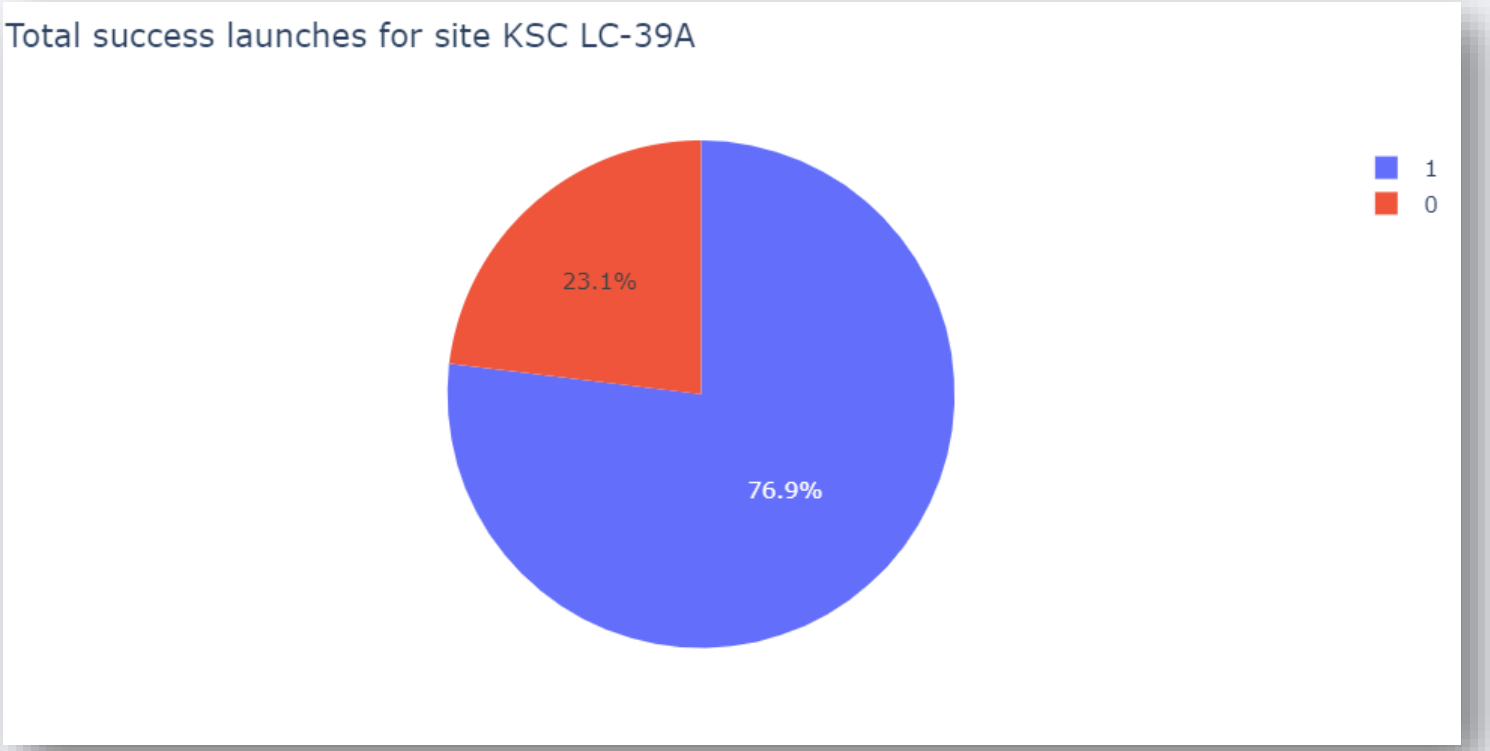
Dashboard : Launch success for all sites

- Pie chart of launch success count for all sites
- The site that has the largest successful launches is KSC LC-39A
- The one that has the worst is CCAFS SLC-40



Dashboard : launch success rate

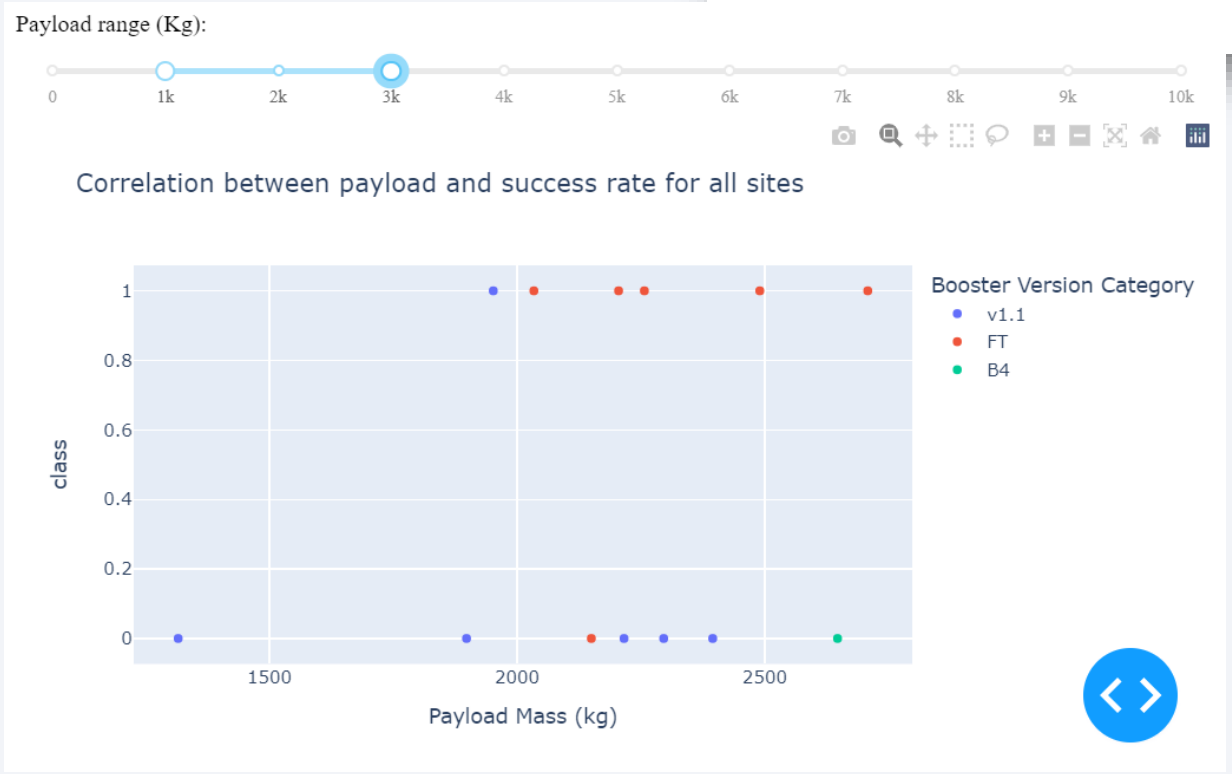
- Piechart for the launch site with highest launch success ratio, it is KSC LC-39A
- 76.9% of the launches of this site are successfull



Dashboard: Payload vs. Launch Outcome

- Scatter plots of Payload vs. Launch Outcome for all sites, with different payloads selected in the range slider

- The important findings are:
  - KSC LC-39A has the largest successful launches
  - KSC LC-39A has the highest launch success rate
  - The payloads between 1900kg and 3700kg have the highest launch success rate
  - The payloads over 5300kg have the lowest launch success rate
  - The F9 Booster version : FT, has the highest launch success rate

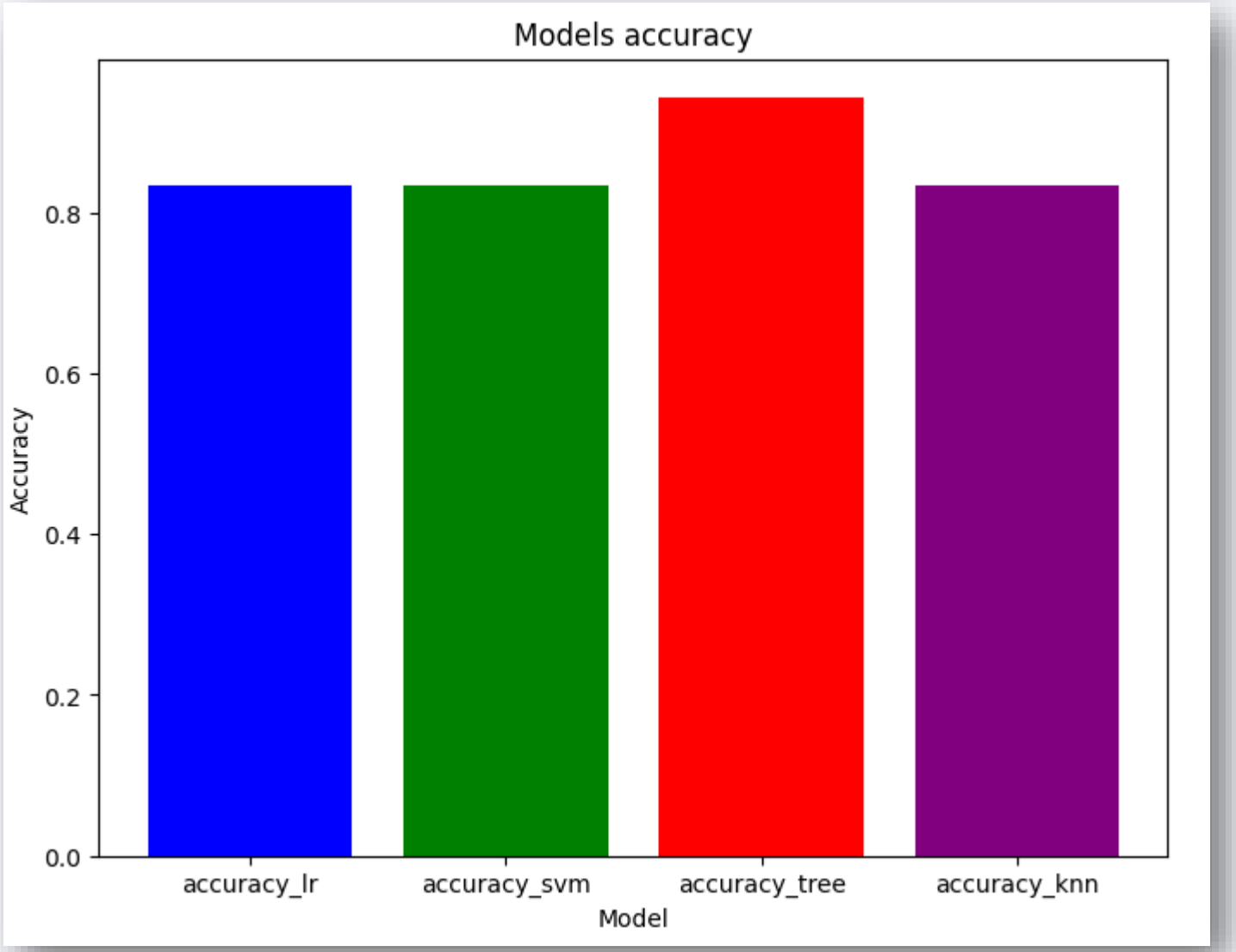


Section 5

Predictive Analysis (Classification)

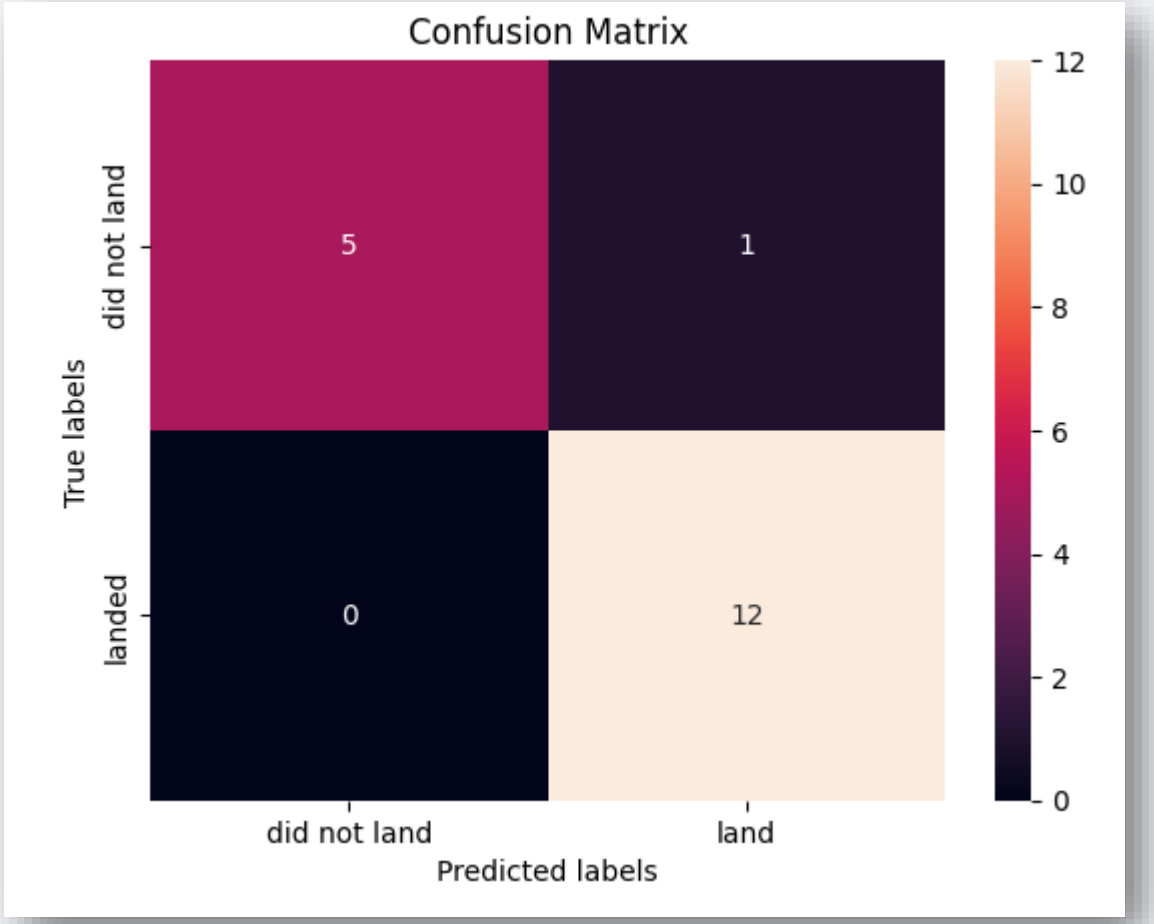
Classification Accuracy

- Bar chart of the built model accuracy for all built classification models
- The model that has the highest classification accuracy is the Tree Decision Model



Confusion Matrix

- Confusion matrix of the best performing model
- It's for the Tree Decision model
- 17 on 18 labels are correctly predicted. With the other models it was only 15 on 18.



Conclusions

- SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch
- Data science with Python has been used to analyse datas about rocket launches and build a model to determine if the first stage will land
- 1- Data collection and Data wrangling with datas about launches from SpaceX API and Wikipedia
- 2- Data analysis (EDA) using visualization and SQL, for analysing the impact of the main features as orbit, booster version, payload mass…
- 3- Interactive visual analytics using Folium and Plotly Dash, for analysing the impact of the launch sites
- 4- Predictive analysis using classification models, for predicting the success landing of the first stage

Appendix

- Here is a generated image with chat-GPT ;)

- Prompt :

- An image depicting a Space X Falcon 9 First Stage landing successfully upon a floating pad in the vast expanse of the ocean. The rocket appears steadied and stable, having landed upright on the designated pad. The vast open blue sea stretches around the landing platform, set against a backdrop of a clear sky. The notable features of the Falcon 9 like the grid fins and landing legs are visible and appear undamaged. Landing flames, smoke, and the residual heat haze from the descent create an atmosphere of accomplishment.

Thank you!