IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Coursera Learner>
<July 31, 2023>

Outline

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

Executive Summary

- Summary of methodologies

- SpaceX Data Collection: SpaceX API & Web Scraping
- SpaceX Data Wrangling: One-hot Coding & Missing Values
- SpaceX Exploratory Analysis with SQL, Pandas, and Matplotlib
- SpaceX Interactive Visualization
- SpaceX Machine Learning Prediction

- Summary of all results

- EDA results
- Interactive Visual Analytics
- Predictive Analytics

Introduction

- Project background and context

The commercial space age is here, companies are making space travel affordable for everyone. Virgin Galactic is providing suborbital spaceflights. Rocket Lab is a small satellite provider.
Blue Origin manufactures sub-orbital and orbital reusable rockets. Perhaps the most successful is SpaceX. SpaceX's accomplishments include: Sending spacecraft to the International Space Station. 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; ther 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.

- Problems you want to find answers

Predict if the Falcon 9 first stage will land successfully using data from Falcon 9 rocket launches advertised on its websites.

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

- Describe how data sets were collected.

Data was collected by SpaceX API and web scraping.

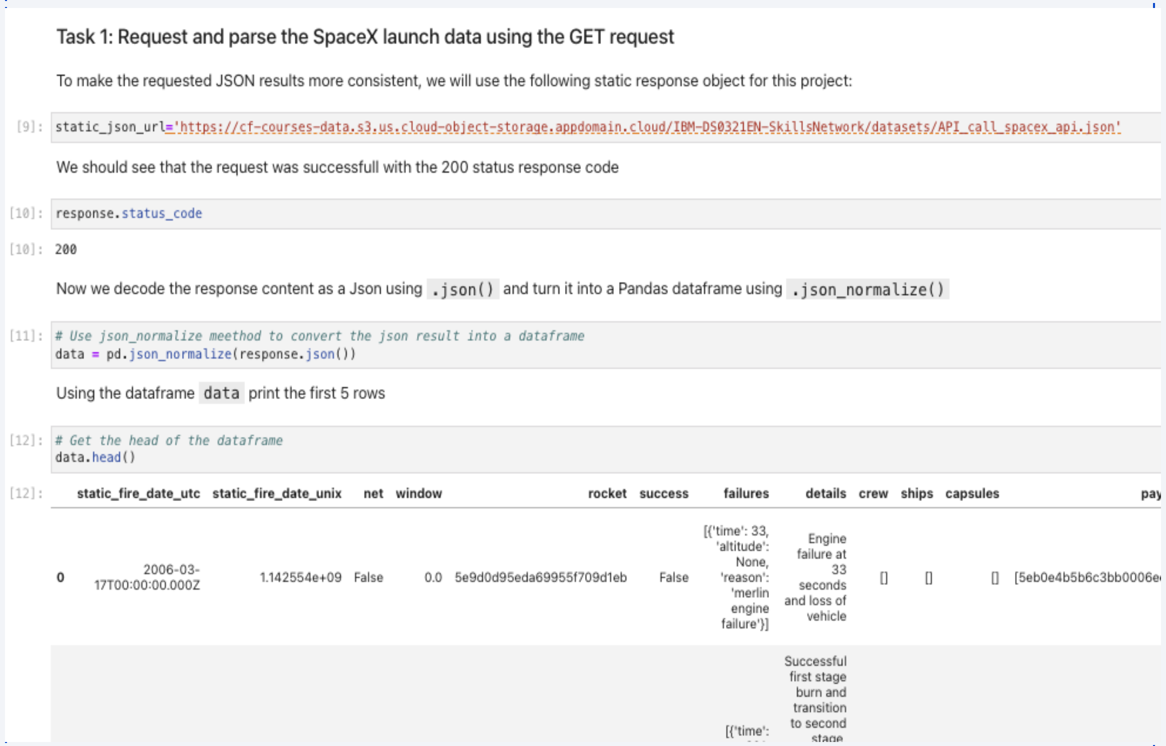
A get request to the SpaceX API was created. There were several helper functions defined to help with data extraction. The returned response content were then decoded as a Json result and converted to a Pandas data frame.

For web scraping, the information was collected from Wikipedia page title List of Falcon 9 and Falcon Heavy Launches. Web scraping helped to retrieve Falcon 9 historical records and stored them in a HTML. Later, BeautifulSoup was applied to parse the HTML result. Finally, I used HTML tag knowledge to select related information and converted it into a Pandas data frame.

Data Collection – SpaceX API

- Data collected using API by making a get request and decoding the result to a Json result, which was later converted to Pandas df.
- Below is the GitHub URL of the completed SpaceX API calls notebook as an external reference and peer-review purpose

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



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:

```
[9]: 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

```
[10]: response.status_code
[10]: 200
```

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

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

Using the dataframe `data` print the first 5 rows

```
[12]: # Get the head of the dataframe
data.head()
```

[12]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	pay
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bb0006e
							[{'time':	Successful first stage burn and transition to second stage				

Data Collection - Scraping

- Data was collected from a Wikipedia page with a get request and a BeautifulSoup object. The parsed content was converted to a Pandas data frame.
- Below is the GitHub URL of the completed web scraping notebook

https://github.com/DSSophia/Coursera/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.

[9]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
re = requests.get(static_url)
```

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

[11]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(re.text)
```

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

[12]:
```
# Use soup.title attribute
soup.title
```

[12]:
```
<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

[13]:
```
# 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.

[14]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">Version,<br/>Booster</a> <sup class="reference" id="cite_
</th>
<th scope="col">Launch site
```

Data Wrangling

After obtaining a Pandas data frame from previous data collection process, data was filtered by BoosterVersion column so that only Falcon 9 related records were kept in the data frame. There were some missing values in the resulted data frame and the missing values in PayloadMass column were replaced with mean value of that column. A new column 'class' was created to classify the launching outcome.

Below is the Github URL:

https://github.com/DSSophia/Coursera/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_1_L3_labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

`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 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 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.

[12]:
```
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:

[13]:
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

[13]:
```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

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 variable `landing_class`:

[19]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [0 if i in bad_outcomes else 1 for i in df['Outcome']]
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

[20]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

[20]:

	Class
0	0
1	0
2	0
3	0

EDA with Data Visualization

Data analysis and feature engineering was performed using Pandas and Matplotlib.

- Exploratory Data Analysis
- Feature Engineering

Used scatter plots to visualize the relationship between Flight Number and Launch Site, Payload and Launch Site, FlightNumber and Orbit type, Payload and Orbit type.

Used Bar chart to visualize the relationship between success rate of each orbit type.

Line plot to visualize the launch success yearly trend.

Below is the GitHub URL of your completed EDA with data visualization notebook, as an external reference and peer-review purpose

https://github.com/DSSophia/Coursera/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

Task 1

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

```
[10]: %sql select distinct Launch_Site from SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

[10]:

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'

```
[11]: %sql select * from SPACEXTABLE where Launch_Site like 'CCA%' LIMIT 5
 * sqlite:///my_data1.db
Done.
```

[11]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landin
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure
2010-08-12	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
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	

Task 3

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

```
[13]: %sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE WHERE Customer ='NASA (CRS)'
 * sqlite:///my_data1.db
Done.
```

[13]:

sum(PAYLOAD_MASS__KG_)
45596

Task 4

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

```
[15]: %sql select avg(PAYLOAD_MASS__KG_), Booster_Version from SPACEXTABLE WHERE Booster_Version=="F9 v1.1"
 * sqlite:///my_data1.db
Done.
```

```
[17]: %%sql
select min(Date) from SPACEXTBL where Landing_Outcome ='Success (ground pad)'
 * sqlite:///my_data1.db
Done.
```

[17]:

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

```
[20]: %%sql

select Booster_Version from SPACEXTABLE where Landing_Outcome="Success (drone ship)" and PAYLOAD_MASS__KG_>4000 AND PAYLOAD_MASS__KG_<6000
 * sqlite:///my_data1.db
Done.
```

[20]:

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

Task 7 ¶

List the total number of successful and failure mission outcomes

```
[21]: %%sql

select count(*), Mission_Outcome from SPACEXTABLE group by Mission_Outcome
 * sqlite:///my_data1.db
Done.
```

[21]:

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

Task 8

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

```
[23]: %%sql
select DISTINCT Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_=
(select max(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

- Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose

https://github.com/DSSophia/Coursera/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Created objects: markers, circles, lines. Created folium map to marked all the launch sites.
- Add the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose

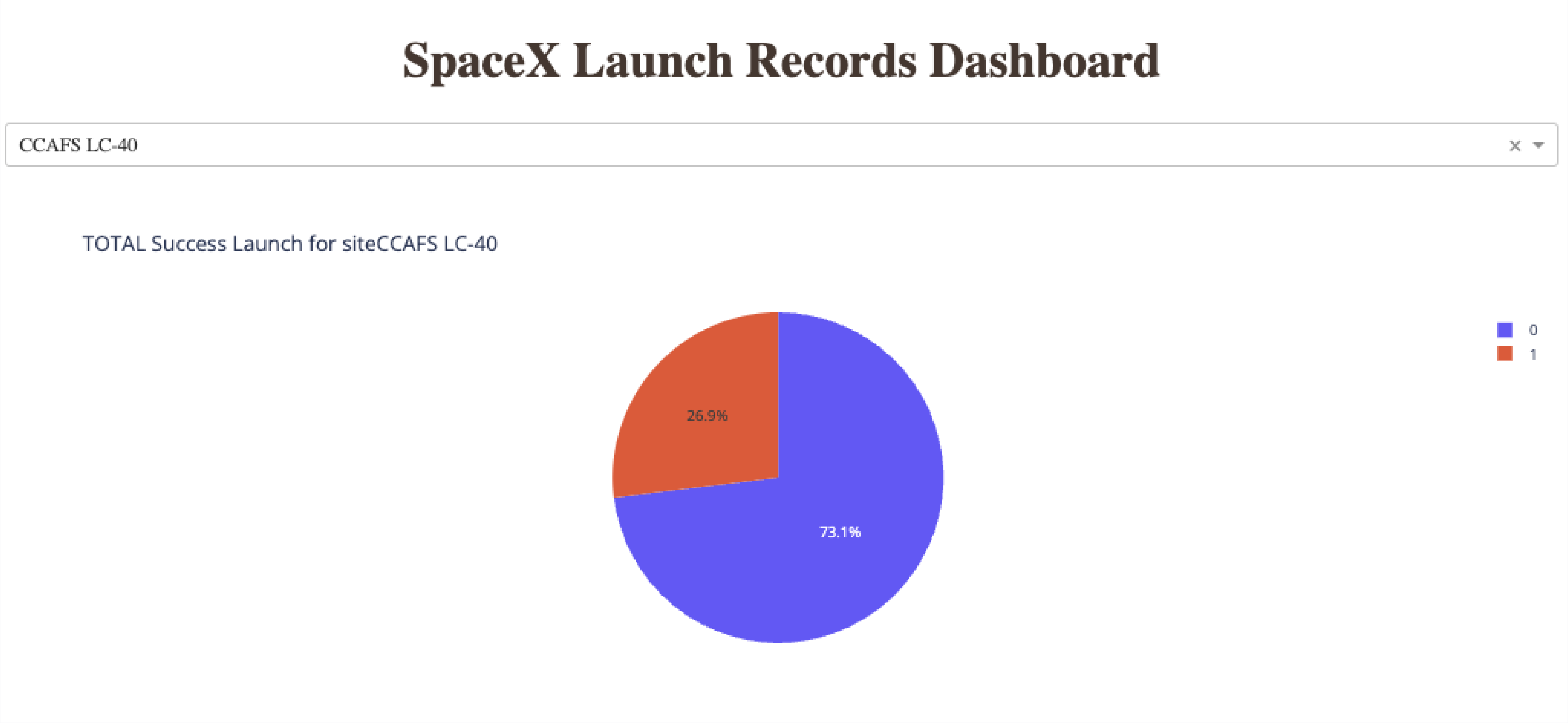
https://github.com/DSSophia/Coursera/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

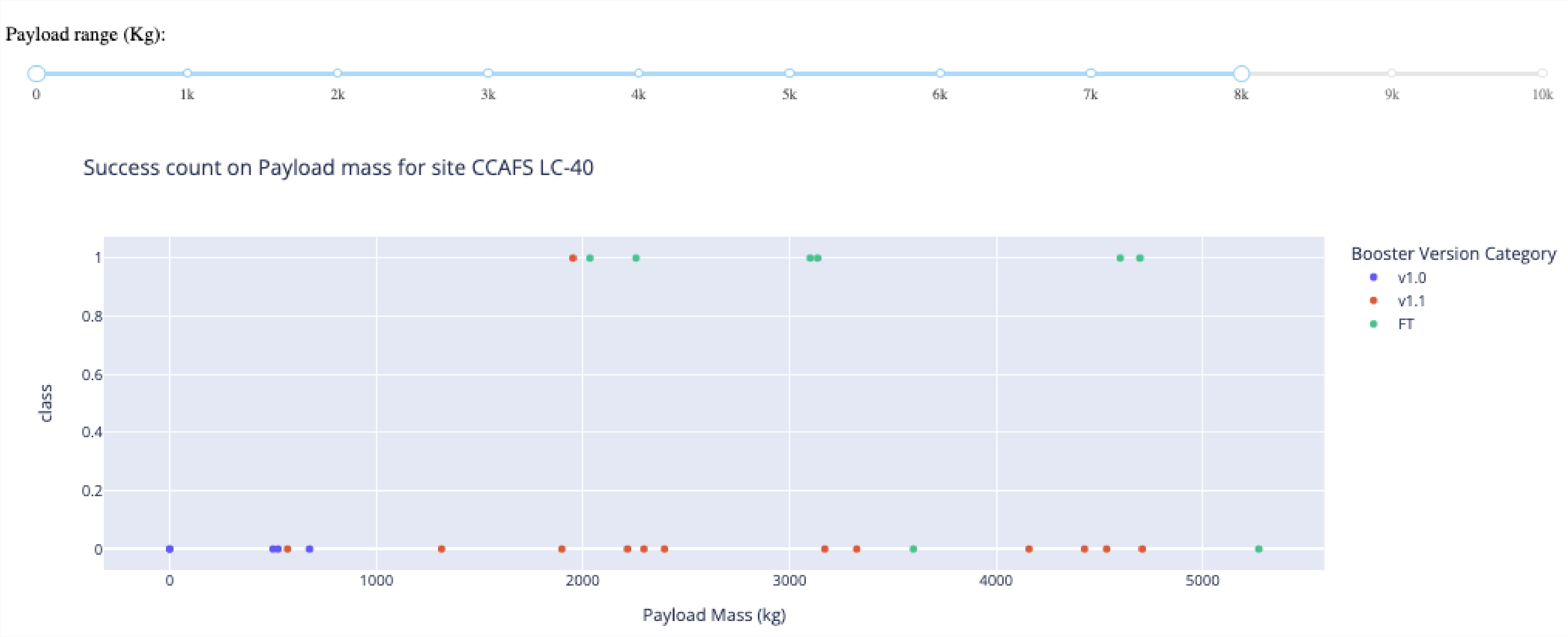
- The Plotly Dash was created by:
  - Adding a launch site drop-down input component
  - Adding a callback function to render success-pie-chart based on selected site dropdown
  - Adding a range slider to select payload
  - Adding a callback function to render the success-payload-scatter-chart scatter plot
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

https://github.com/DSSophia/Coursera/blob/main/spacex_dash_app.py

Plotly Dash


SpaceX Launch Records Dashboard
CCAFS LC-40
TOTAL Success Launch for siteCCAFS LC-40
0
1
26.9%
73.1%

Plotly Dash – Continued



Predictive Analysis (Classification)

- After obtaining a data frame containing related information. The data was splitted to training and testing set. Logistic regression, SVM, decision tree, and KNN was applied to build a predictive model. GridSearchCV was used to find the best parameters for the model.
- Accuracy and score were used to evaluate model performance.
- Below is the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

https://github.com/DSSophia/Coursera/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

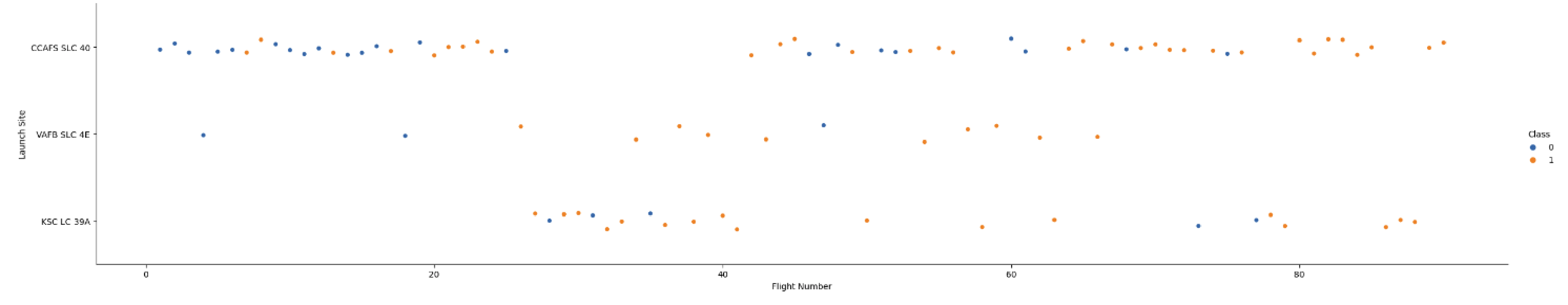
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

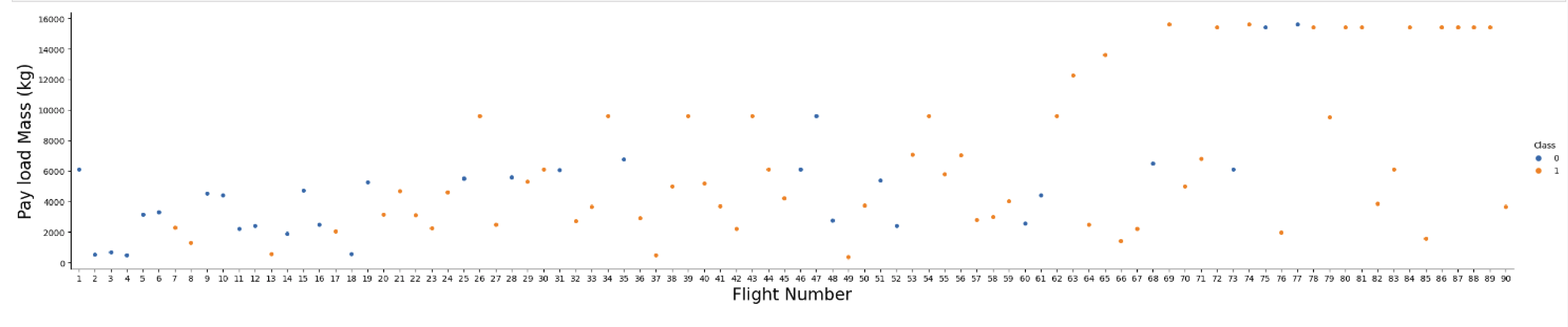
Flight Number vs. Launch Site

- Scatter plot of Flight Number vs. Launch Site



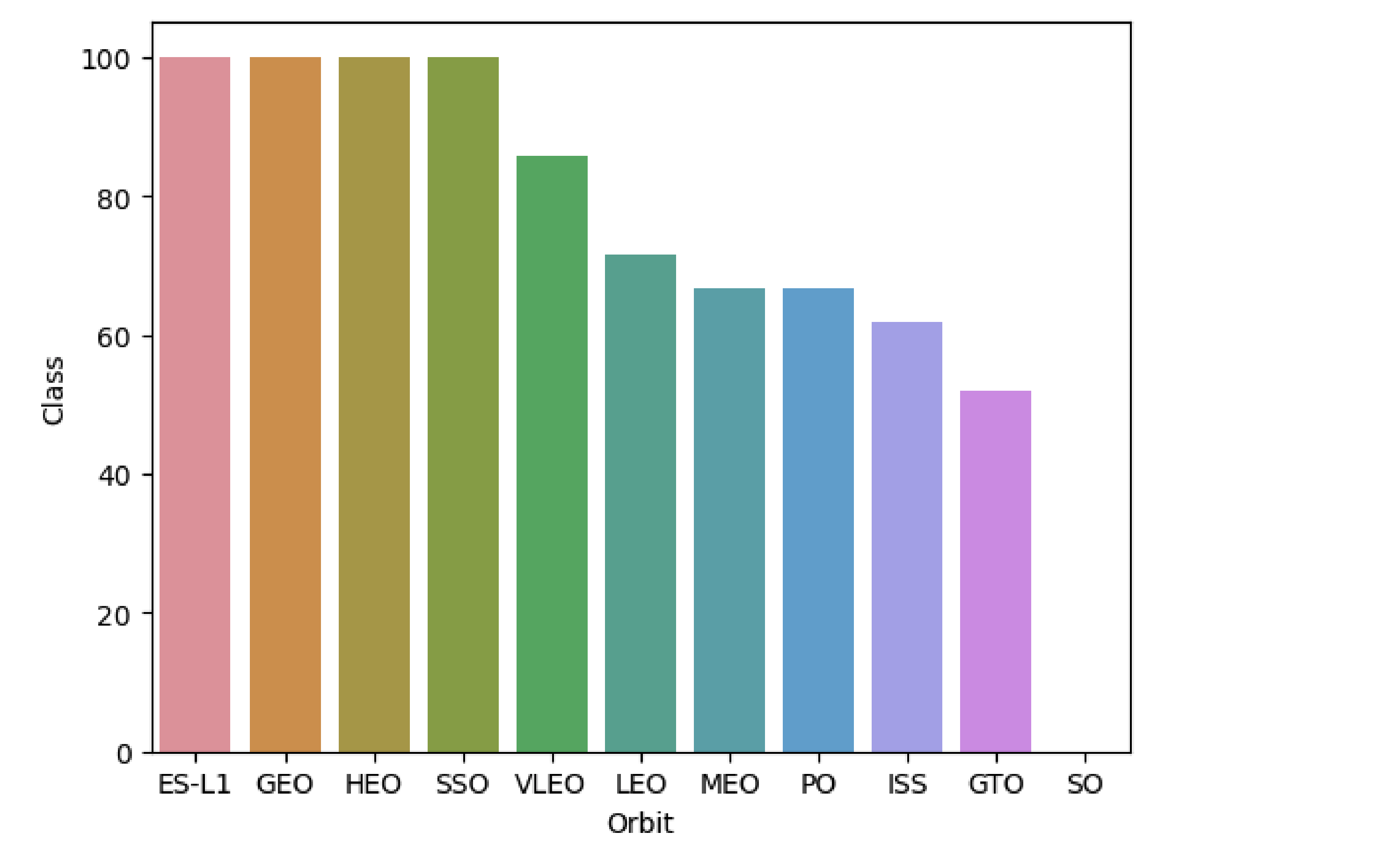
Payload vs. Launch Site

- Scatter plot of Payload vs. Launch Site



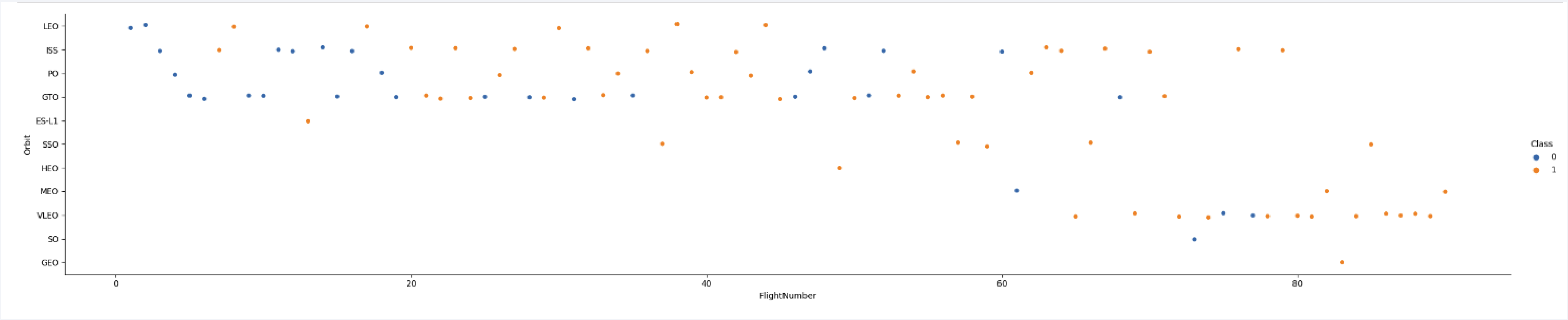
Success Rate vs. Orbit Type

- Bar chart for the success rate of each orbit type



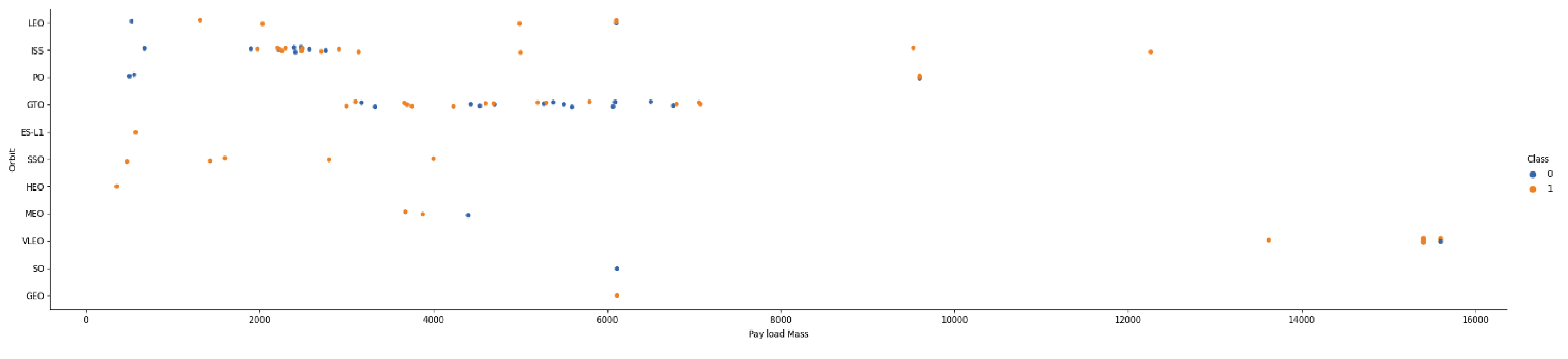
Flight Number vs. Orbit Type

- Scatter point of Flight number vs. Orbit type



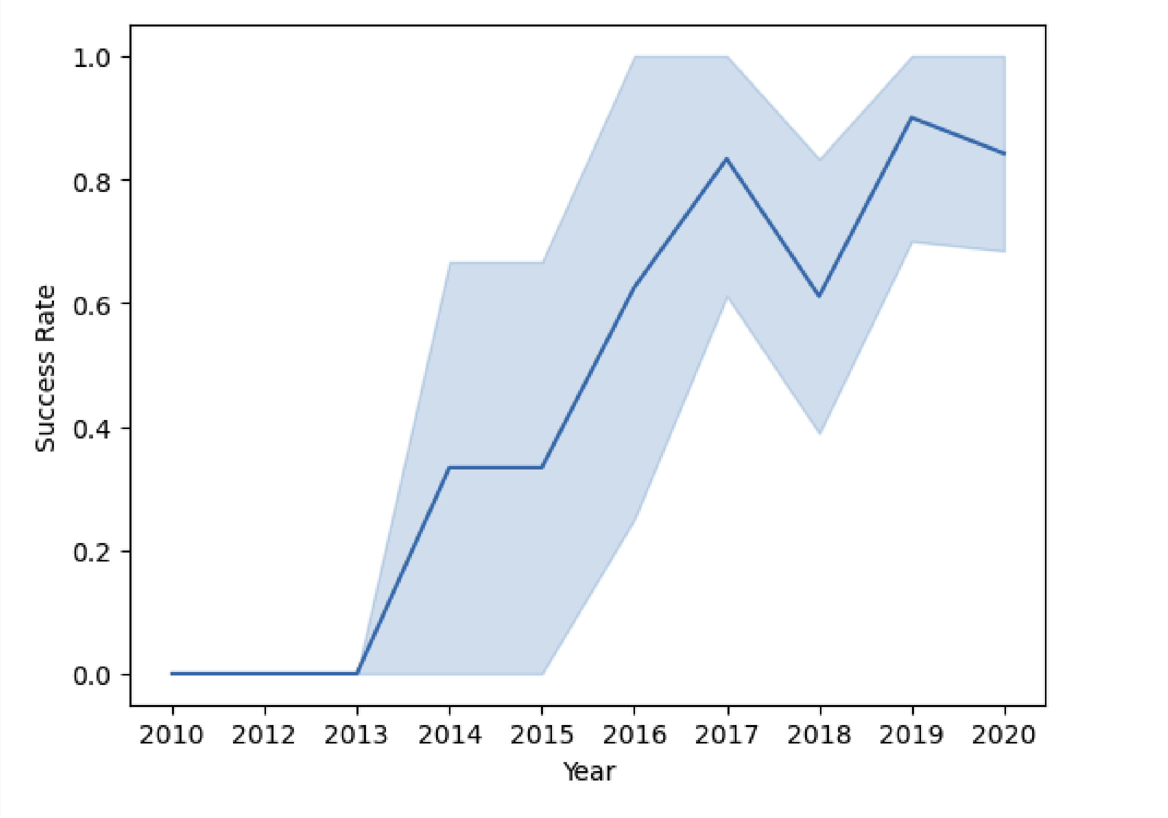
Payload vs. Orbit Type

- Scatter point of payload vs. orbit type



Launch Success Yearly Trend

- Line chart of yearly average success rate



All Launch Site Names

- Find the names of the unique launch sites

Task 1

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

[10]:
```
%sql select distinct Launch_Site from SPACEXTABLE
```

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

[10]:

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

Task 2

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

```
[11]: %sql select * from SPACEXTABLE where Launch_Site like 'CCA%' LIMIT 5
```

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

[11]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

Task 3

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

```
[13]: %sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE WHERE Customer ='NASA (CRS)'

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

[13]:

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

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

Task 4

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

```
[15]: %sql select avg(PAYLOAD_MASS__KG_), Booster_Version from SPACEXTABLE WHERE Booster_Version=="F9 v1.1"
```

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

[15]:

avg(PAYLOAD_MASS__KG_)	Booster_Version
2928.4	F9 v1.1

First Successful Ground Landing Date

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

Task 5

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

Hint:Use min function

[17]:
```
%%sql
select min(Date) from SPACEXTBL where Landing_Outcome ='Success (ground pad)'
```

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

[17]:

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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

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

[20]:
```
%%sql

select Booster_Version from SPACEXTABLE where Landing_Outcome="Success (drone ship)" and PAYLOAD_MASS__KG_>4000 AND PAYLOAD_MASS__KG_<6000
```

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

[20]:

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

Task 7

List the total number of successful and failure mission outcomes

[21]:
```
%%sql

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

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

[21]:

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

Boosters Carried Maximum Payload

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

Task 8

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

[23]:
```
%%sql
select DISTINCT Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_=
(select max(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

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

[23]:

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

2015 Launch Records

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

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, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

[25]:
```
%%sql
select substr(Date,4,2) as month, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE WHERE substr(Date,7,4)=="2015" and Landing_Outcome="Failure (drone ship)"
```

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

[25]:

month	Landing_Outcome	Booster_Version	Launch_Site

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

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.

[27]:
```
%%sql
select count(*) as total, Landing_Outcome from SPACEXTABLE group by Landing_Outcome having date>="2010-06-04" and date <="2017-03-20" order by total desc
```

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

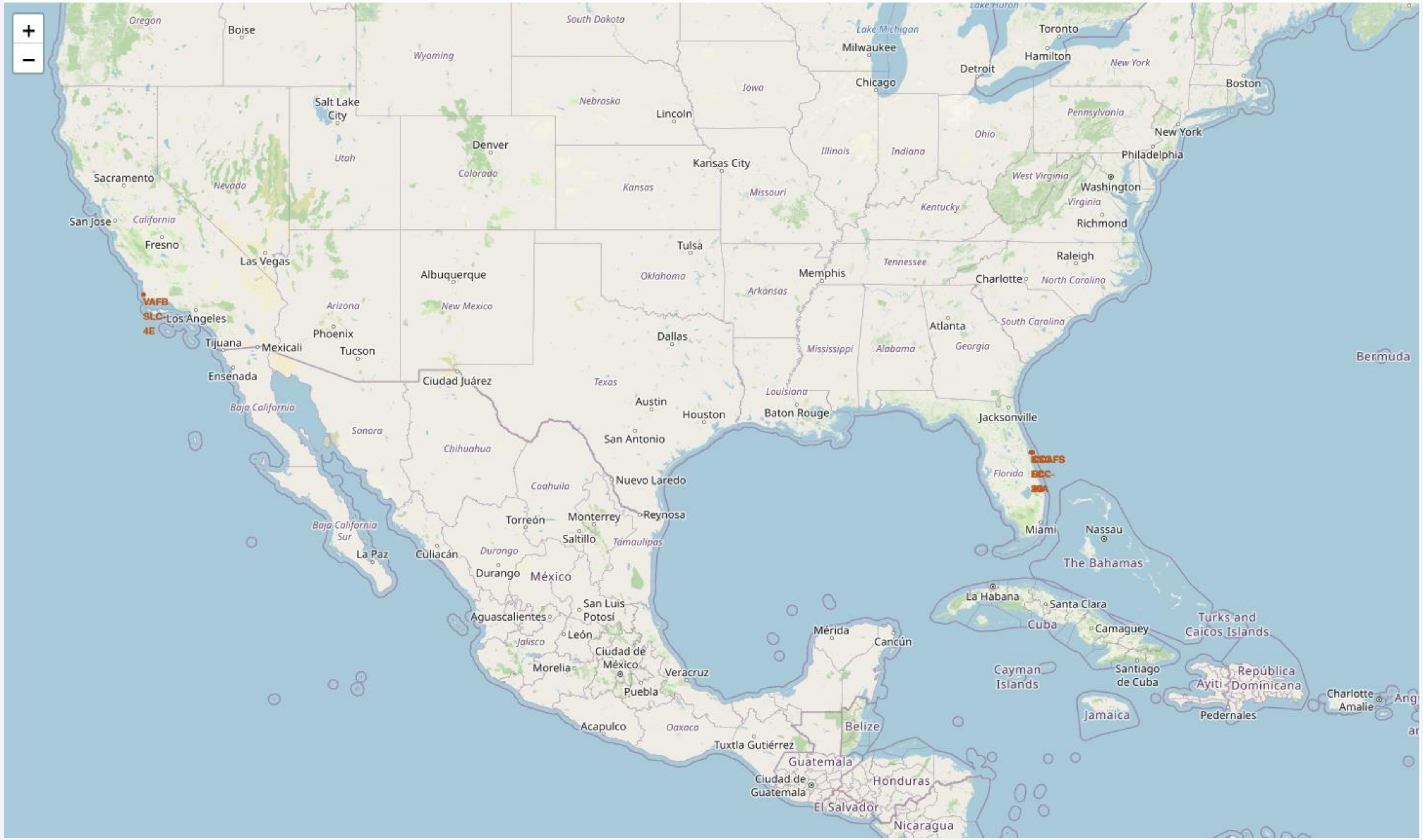
[27]:

total	Landing_Outcome
21	No attempt
14	Success (drone ship)
9	Success (ground pad)
5	Failure (drone ship)
5	Controlled (ocean)
2	Uncontrolled (ocean)
1	Precluded (drone ship)

Section 3

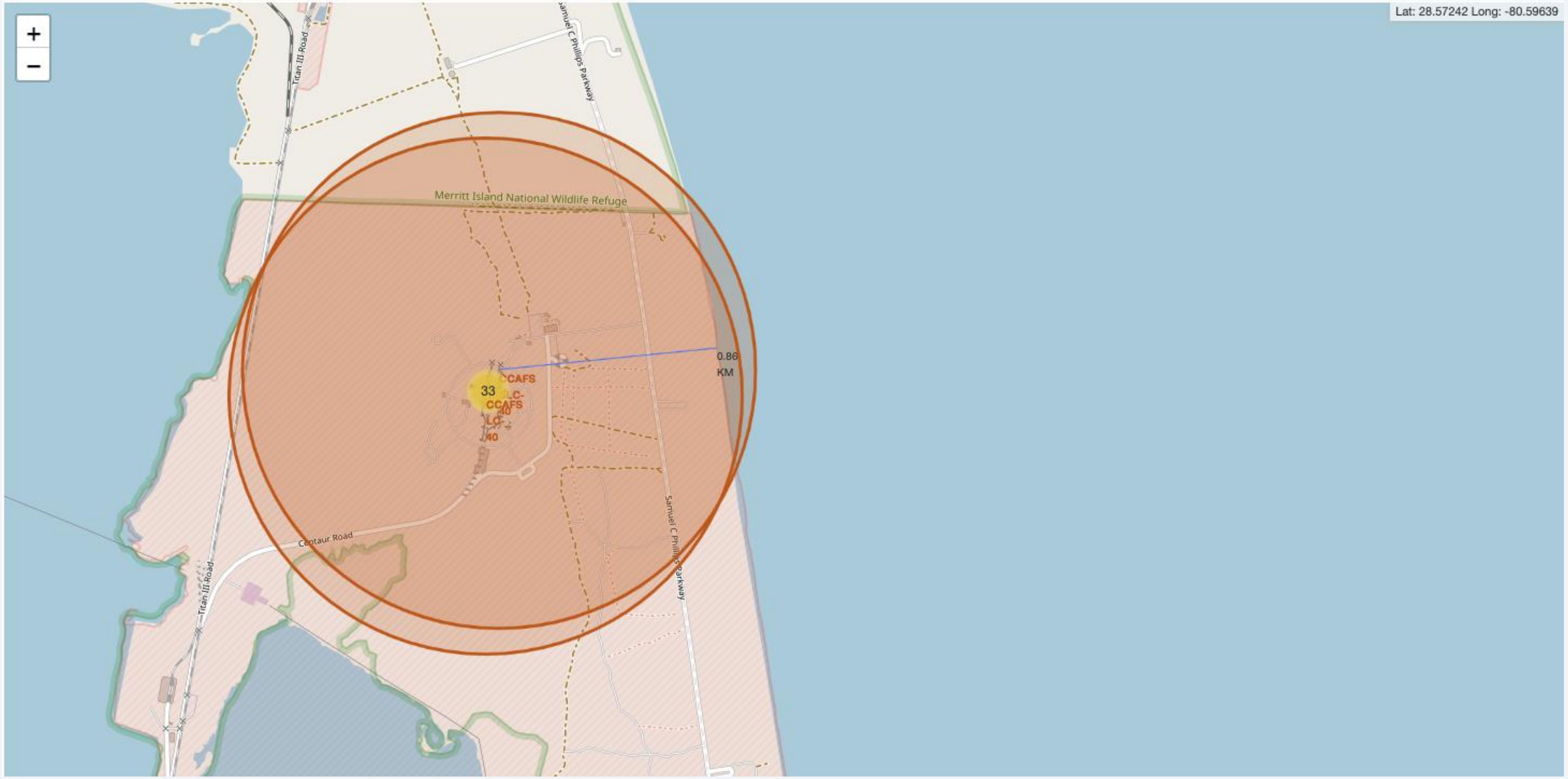
Launch Sites Proximities Analysis

Mark all launch sites on a map



Mark the success/failed launches for each site on the map

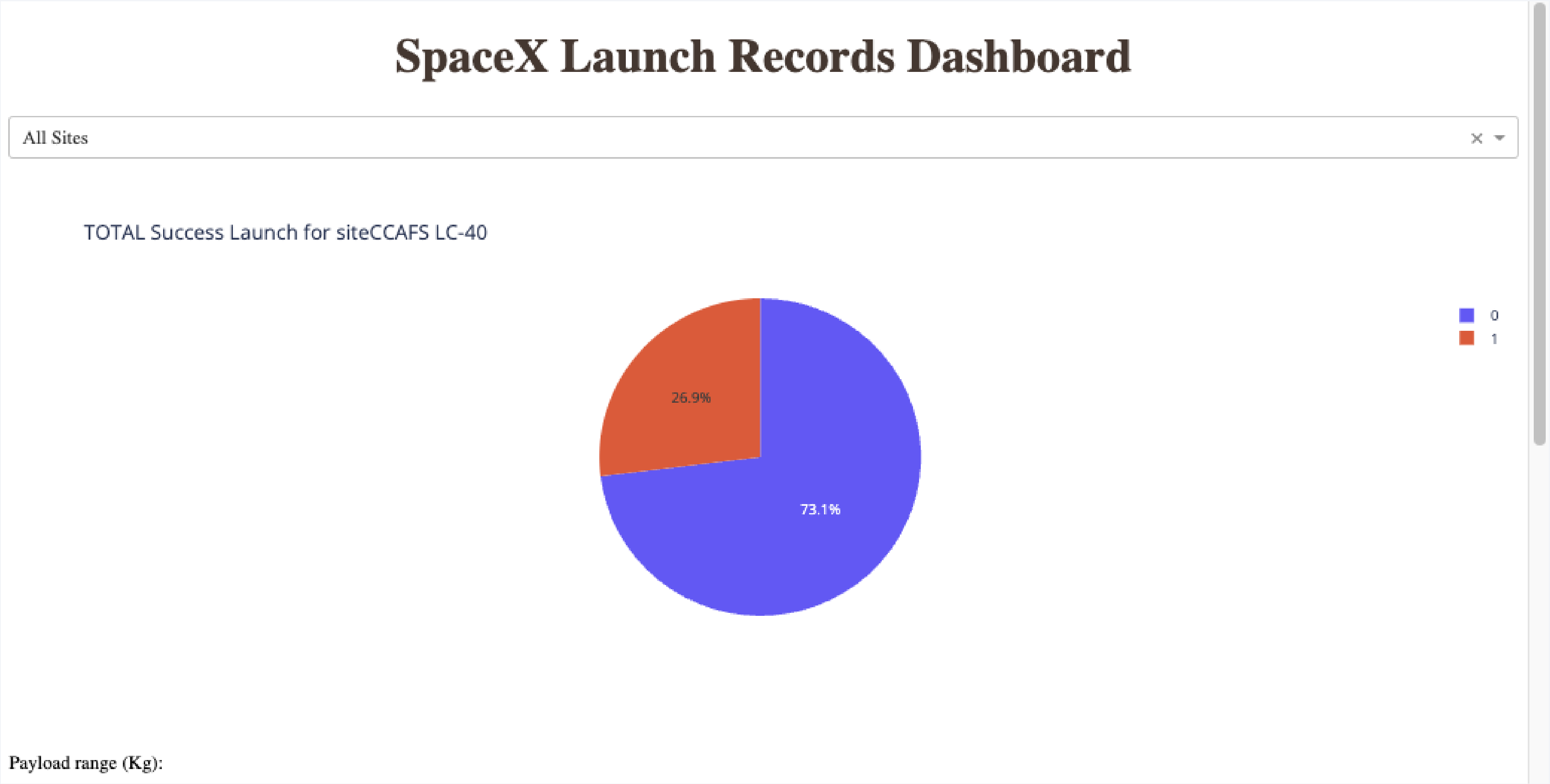
Calculate the distances between a launch site to its proximities



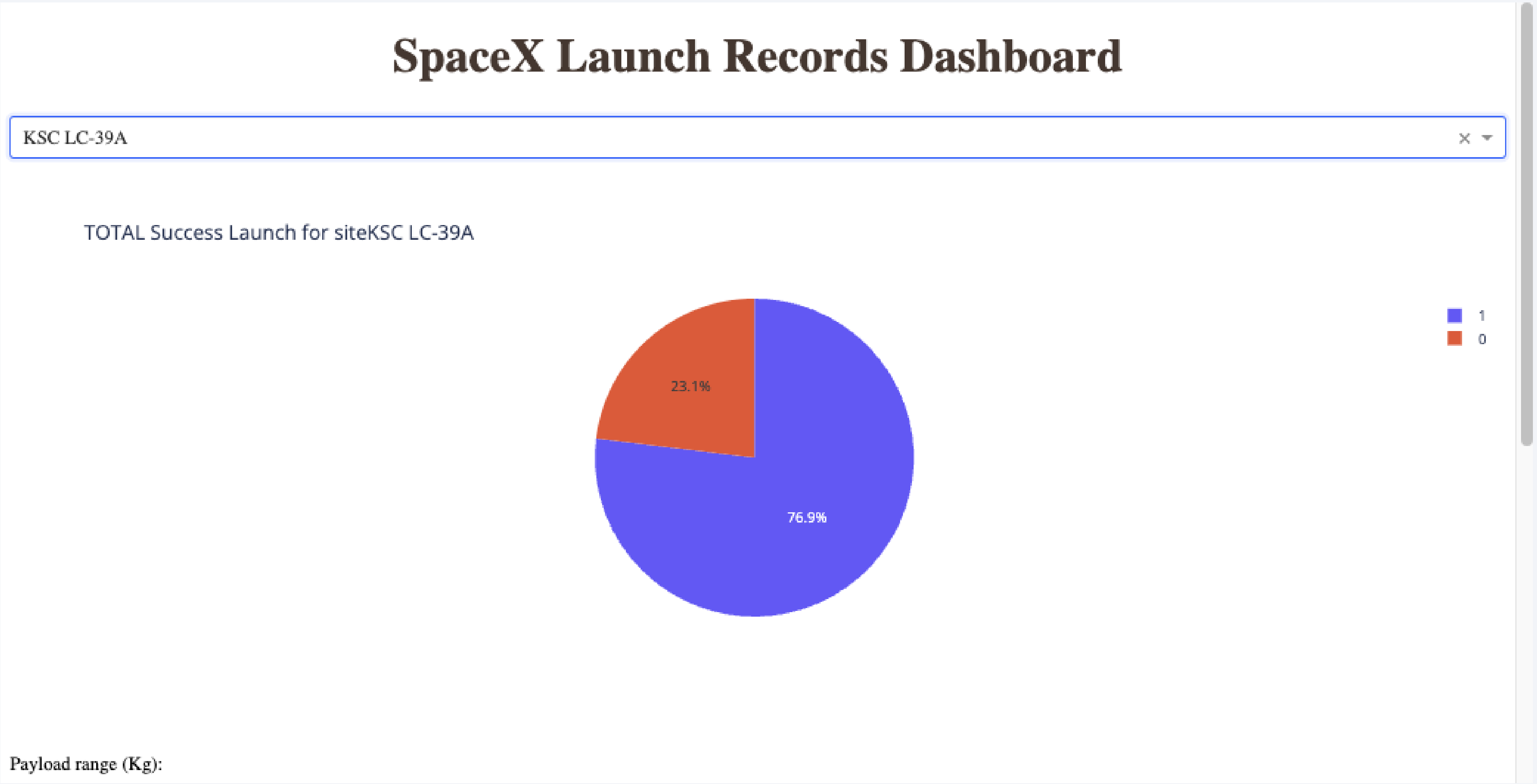
Section 4

Build a Dashboard with Plotly Dash

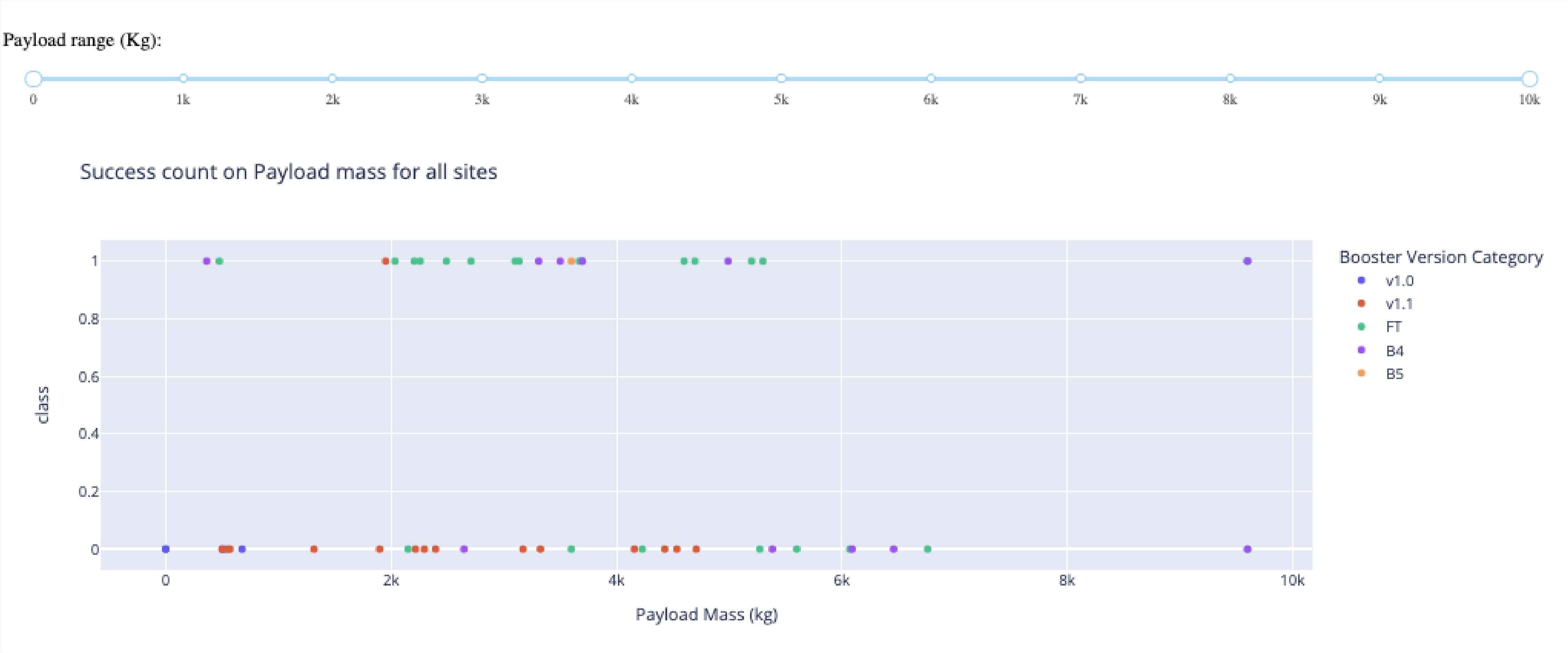
All Site – Pie chart



Highest launch success ratio – pie chart



Payload vs Launch Outcome scatter plot



Section 5

Predictive Analysis (Classification)

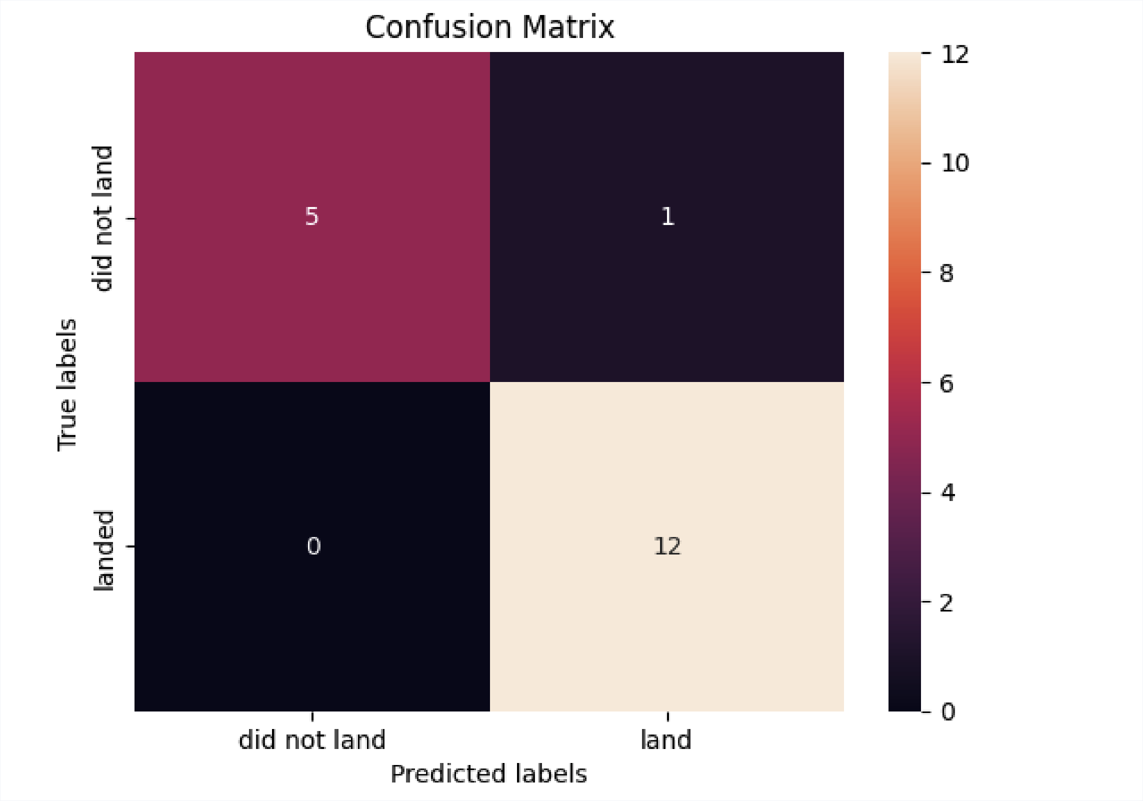
Classification Accuracy

- Visualize the built model accuracy for all built classification models

Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

Confusion Matrix

- Show the confusion matrix of the best performing model with an explanation



Conclusions

- KSC LC 39A launch site has the highest success rate compared to other launch sites
- The success rate increases with years
- The launching at GTO orbit has the lowest success rate.
- Decision tree model performs the best when predict the success rate compared to logistic regression, SVM, and KNN model

Thank you!