



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Abhishek Chatterjee
05-10-23



Outline

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

Executive Summary

➤ Summary of methodologies

- Data Collection through API
- Data Collection with Web Scraping
- Data Wrangling
- Exploratory Data Analysis with SQL
- Exploratory Data Analysis with Data Visualization
- Interactive Visual Analytics with Folium
- Machine Learning Prediction

➤ Summary of all results

- Exploratory Data Analysis result
- Interactive analytics in screenshots
- Predictive Analytics result

Introduction

- Project background and context

Space X 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 Space X 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 space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

- What factors determine if the rocket will land successfully?
- The interaction amongst various features that determine the success rate of a successful landing.
- What operating conditions needs to be in place to ensure a successful landing program.
- What is the previous record on successful landing? And location of the landing having highest success rate ?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
 - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
 - One-hot encoding was applied to categorical features
- 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

- The data was collected using various methods
 - Data collection was done using get request to the SpaceX API.
 - Next, we decoded the response content as a Json using `.json()` function call and turn it into a pandas dataframe using `.json_normalize()`.
 - We then cleaned the data, checked for missing values and fill in missing values where necessary.
 - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
 - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is [Github link \(click in github\)](#)

▼ 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:

```
[11]: 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 successful with the 200 status response code

```
[12]: response.status_code
```

```
[12]: 200
```

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

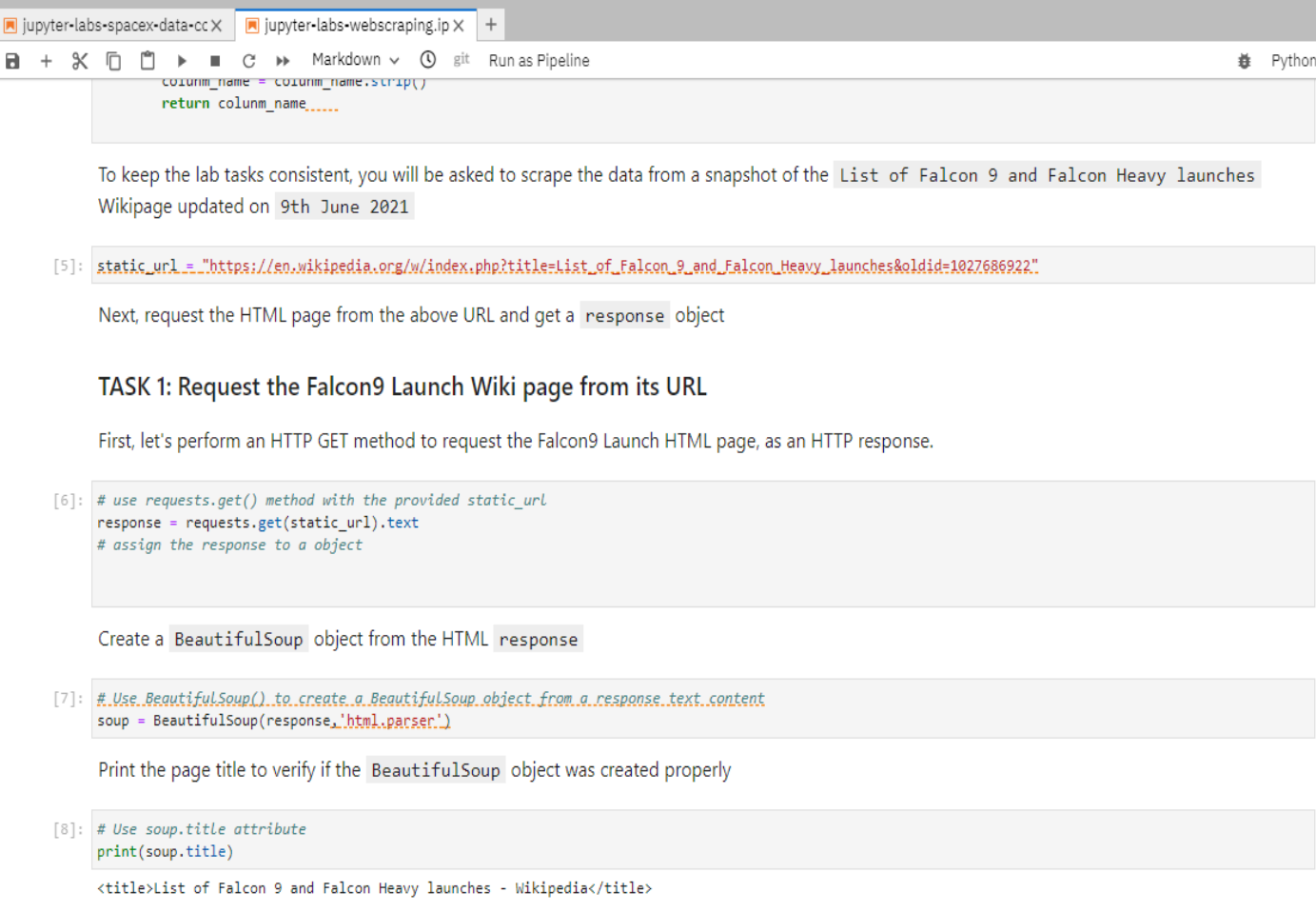
```
[13]: # Use json_normalize meethod to convert the json result into a dataframe
response = requests.get(static_json_url)
response.json()
data = pd.json_normalize(response.json())
data
```

```
[13]:
```

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of vehicle	[]

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is [Github link for webscraping](#)



The screenshot shows a Jupyter Notebook with two tabs: 'jupyter-labs-spacex-data-cc X' and 'jupyter-labs-webscraping.ip X'. The active tab contains the following content:

```
column_name = column_name.strip()
return column_name.....
```

To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the `List of Falcon 9 and Falcon Heavy launches` Wikipage updated on `9th June 2021`

[5]: `static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"`

Next, request the HTML page from the above URL and get a `response` object

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.

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

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

[7]: `# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response, 'html.parser')`

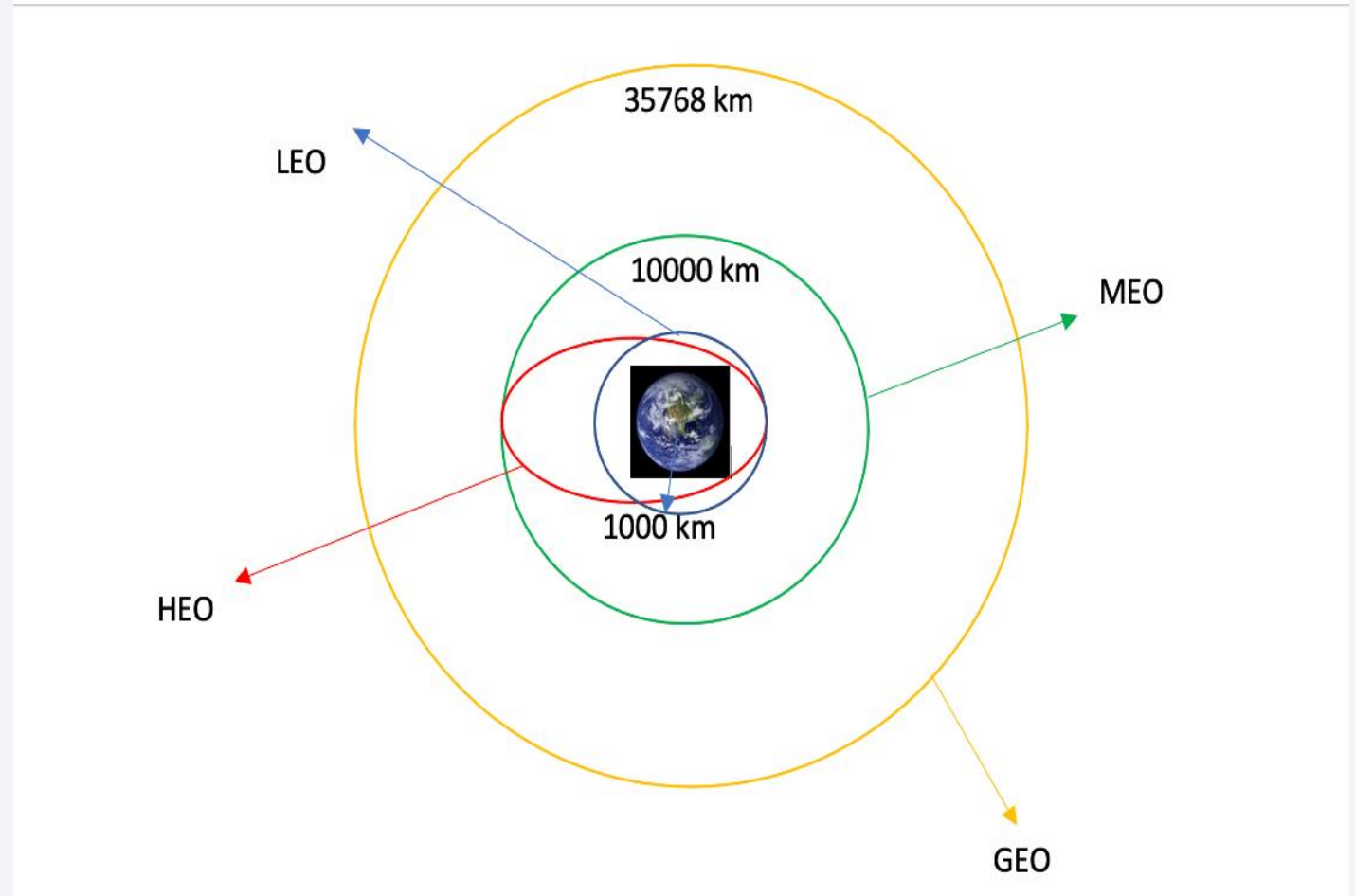
Print the page title to verify if the `BeautifulSoup` object was created properly

[8]: `# Use soup.title attribute
print(soup.title)`

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

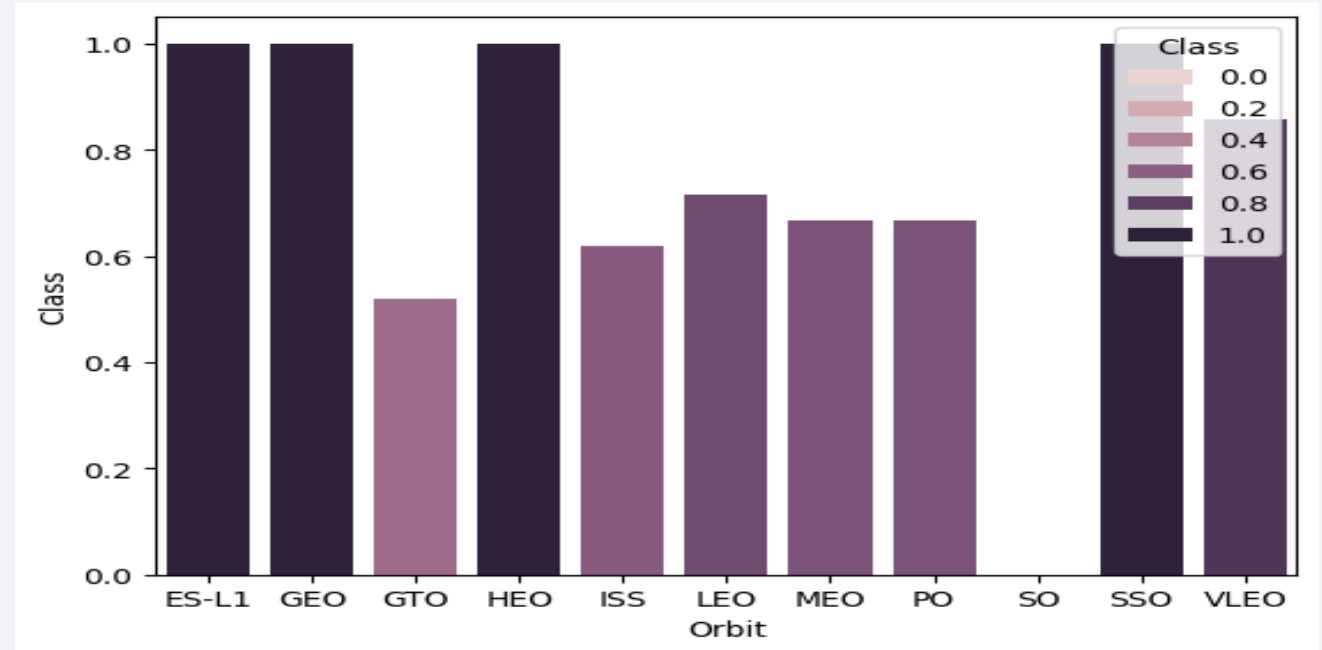
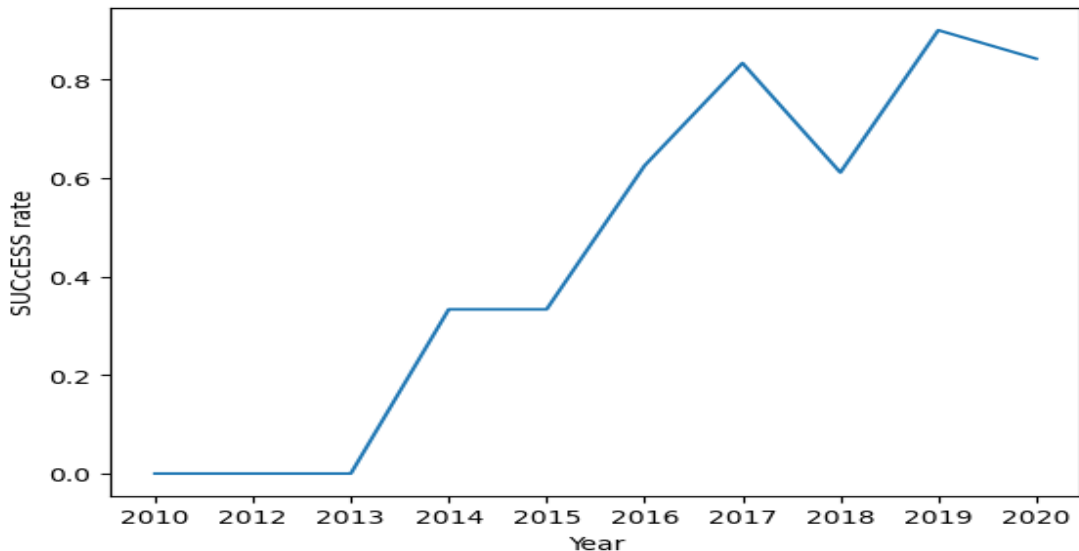
Data Wrangling

- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is [Github link for data wrangling](#)



EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.



- [Github Link for EDA datavisualization](#)

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
 - The names of unique launch sites in the space mission.
 - The total payload mass carried by boosters launched by NASA (CRS)
 - The average payload mass carried by booster version F9 v1.1
 - The total number of successful and failure mission outcomes
 - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is [Github link for the EDA_SQL](#)

Build an Interactive Map with Folium

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

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is [Github link for Dashboard](#)

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is [Github link for Predictive Analysis](#)

Results

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

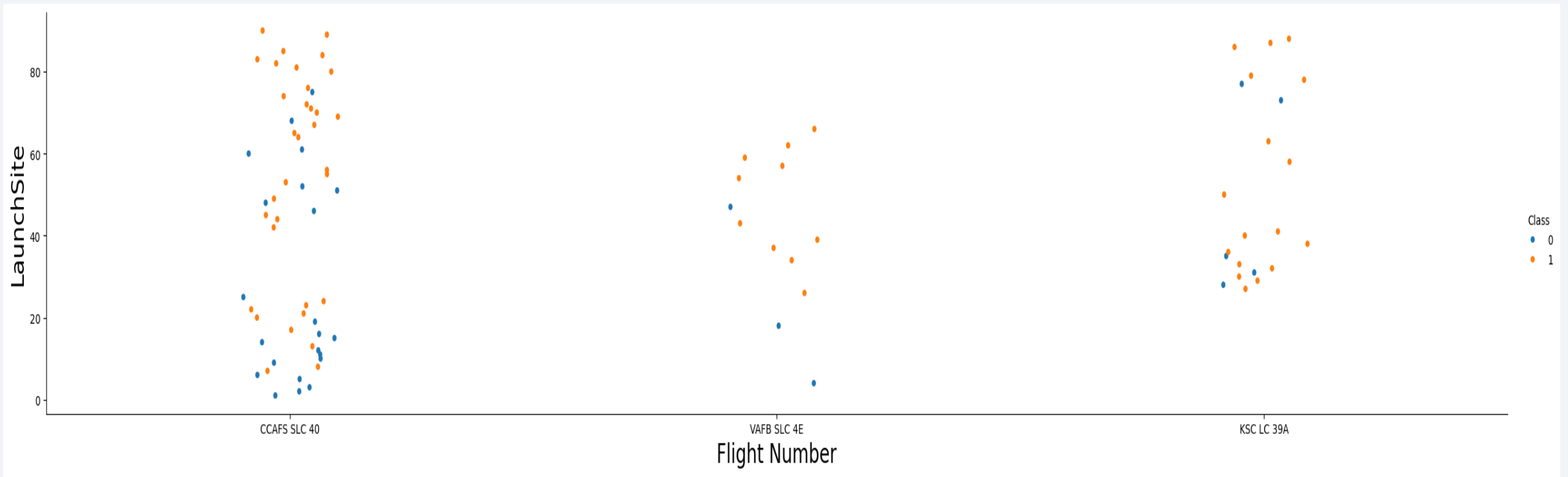
The background of the slide is an abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks in shades of blue and red, creating a sense of motion or data flow. A faint, light blue grid pattern is also visible, particularly in the lower-left quadrant. The overall effect is high-tech and digital.

Section 2

Insights drawn from EDA

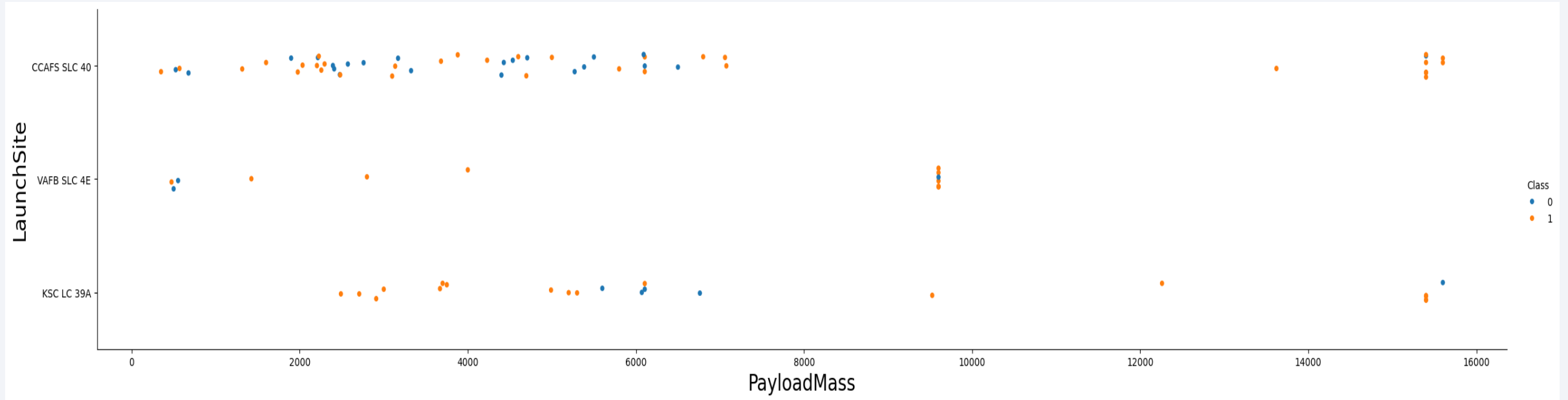
Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



Payload vs. Launch Site

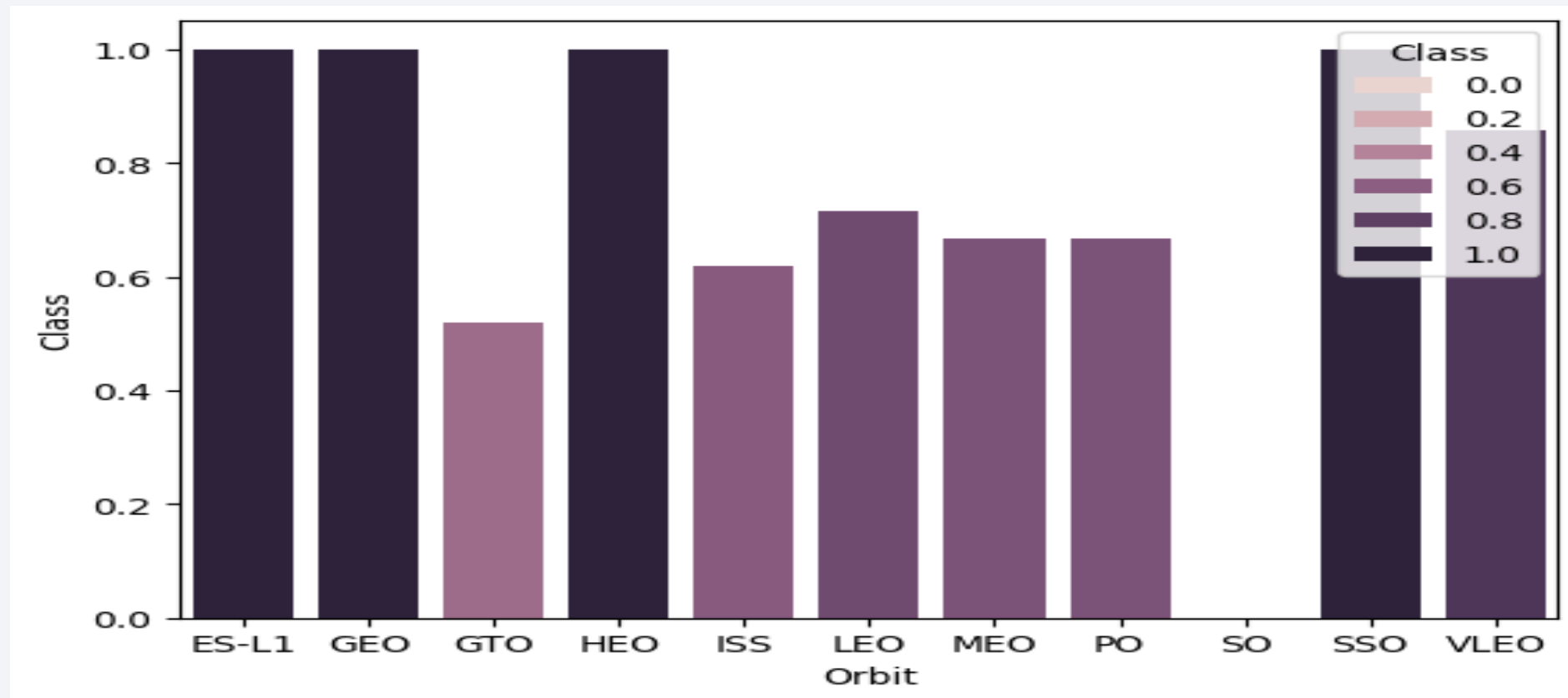
- Payload vs. Launch Site

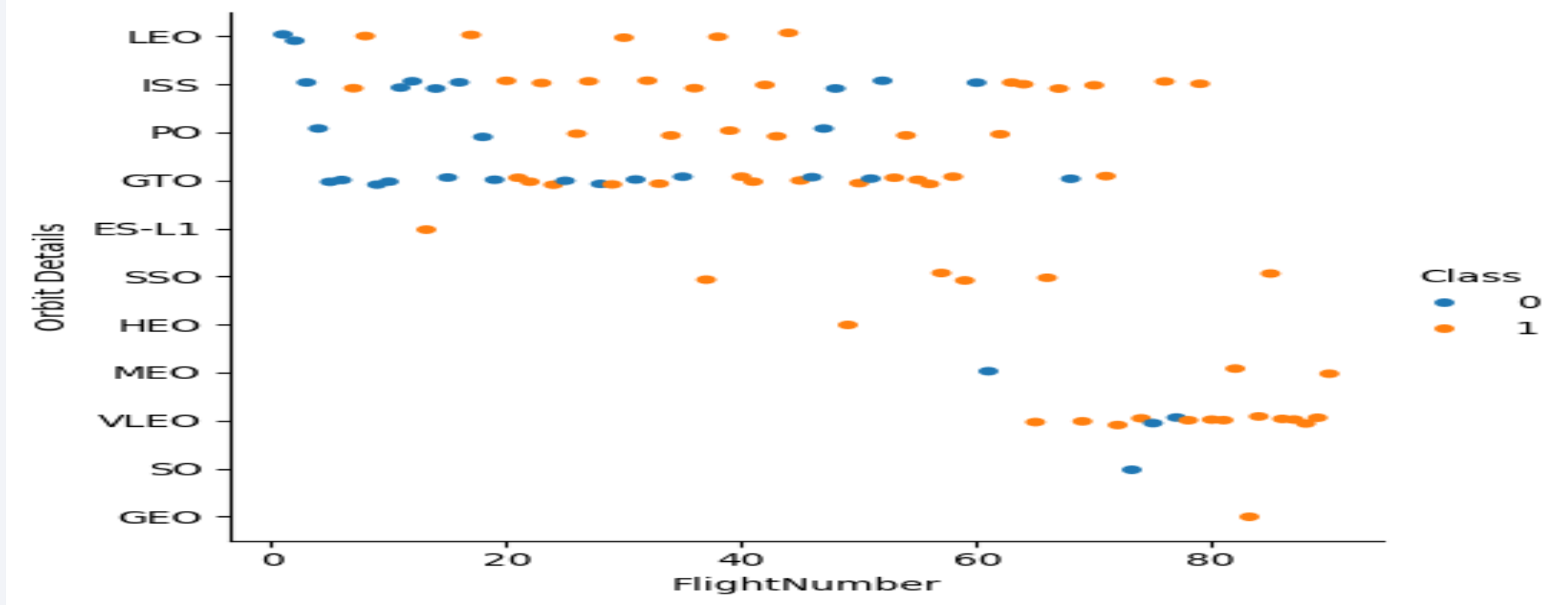


The greater the payload mass for launch site the higher the success rate for the rocket .

Success Rate vs. Orbit Type

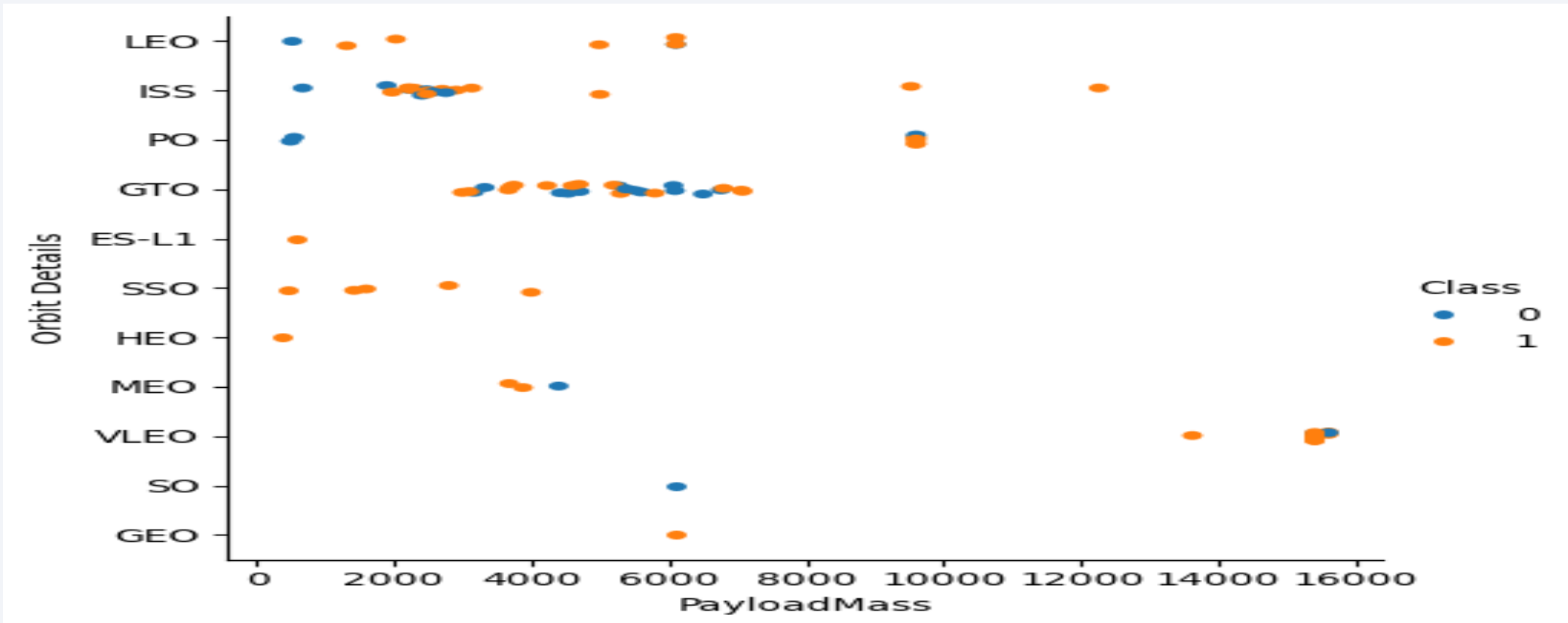
- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate .





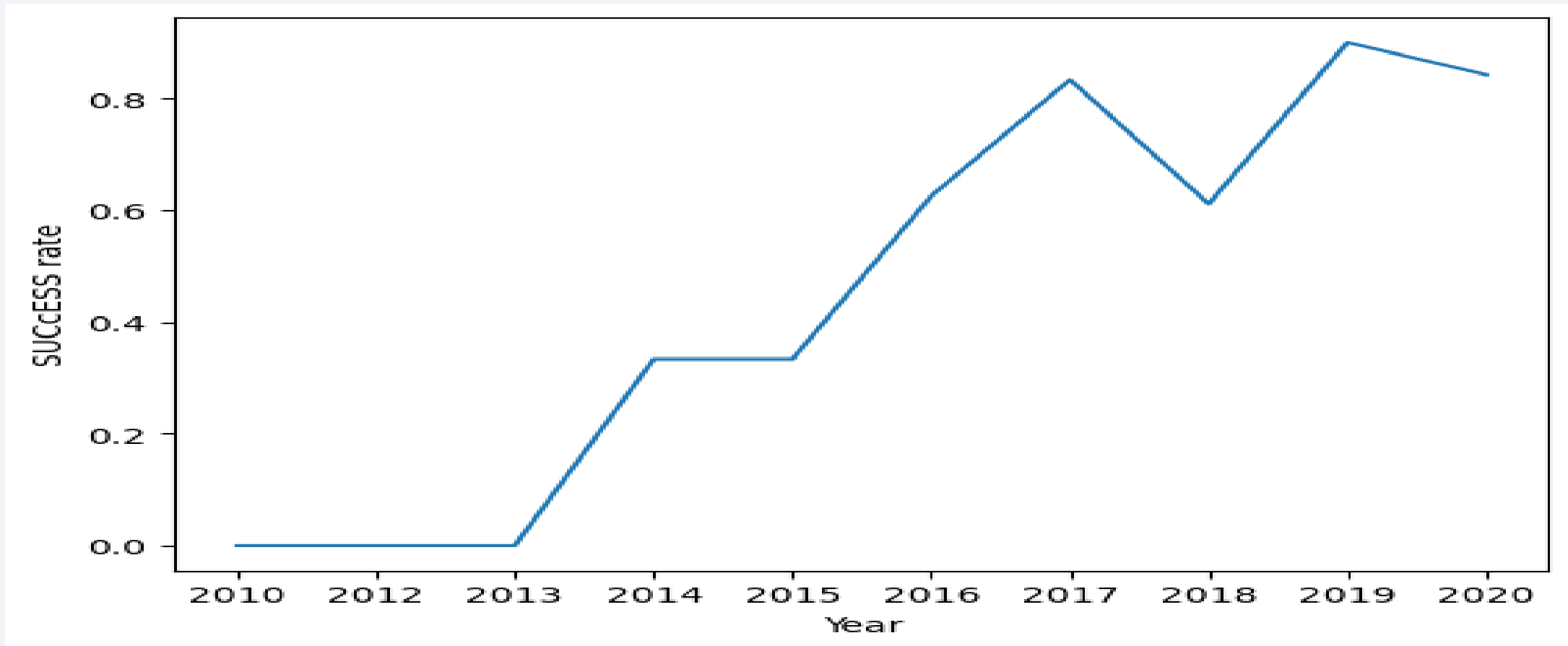
Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

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

Task 1

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

```
[27]: %sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
```

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

```
Done.
```

```
[27]: Launch_Site
```

```
CCAFS LC-40
```

```
VAFB SLC-4E
```

```
KSC LC-39A
```

```
CCAFS SLC-40
```

Launch Site Names Begin with 'CCA'

- We used the query above to display 5 records where launch sites begin with `CCA`

Task 2

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

```
[28]: %sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

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

Done.

```
[28]: Launch_Site
```

```
CCAFS LC-40
```

```
CCAFS LC-40
```

```
CCAFS LC-40
```

```
CCAFS LC-40
```

```
CCAFS LC-40
```

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 619967 using the query below

▼ Task 3

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

```
[30]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS PAYLOADMASS FROM SPACEXTBL;
```

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

Done.

```
[30]: PAYLOADMASS
```

```
619967
```

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 6138.287128712871

Task 4

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

```
[31]: %sql SELECT AVG(PAYLOAD_MASS__KG_) AS PAYLOADMASS FROM SPACEXTBL;
```

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

Done.

```
[31]: PAYLOADMASS
```

```
6138.287128712871
```

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 06-04-2010

Task 5

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

Hint: Use min function

```
[32]: %sql SELECT MIN(DATE) FROM SPACEXTBL;
```

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

```
Done.
```

```
[32]: MIN(DATE)
```

```
2010-04-06
```


Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

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

```
[36]: %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE Landing_Outcome='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 ;
```

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

```
Done.
```

```
[36]: Booster_Version
```

```
F9 FT B1022
```

```
F9 FT B1026
```

```
F9 FT B1021.2
```

```
F9 FT B1031.2
```

Total Number of Successful and Failure Mission Outcomes

- We used wildcard like '%' to filter for **WHERE** MissionOutcome was a success or a failure.

Task 7

List the total number of successful and failure mission outcomes

```
[38]: %sql SELECT COUNT(MISSION_OUTCOME) AS MISSIONOUTCOMES FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

Done.

```
[38]: MISSIONOUTCOMES
```

MISSIONOUTCOMES
1
98
1
1

Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

Task 8

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

```
[46]: %sql SELECT BOOSTER_VERSION AS BOOSTERVERSIONS FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

```
[46]: BOOSTERVERSIONS
```

```
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

- We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

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

In [18]:

```
task_9 = '''
    SELECT BoosterVersion, LaunchSite, LandingOutcome
    FROM SpaceX
    WHERE LandingOutcome LIKE 'Failure (drone ship)'
    AND Date BETWEEN '2015-01-01' AND '2015-12-31'
    ...
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2017-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome 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.

```
[54]: %sql SELECT Landing_Outcome FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' ORDER BY DATE DESC;
```

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

```
Done.
```

```
[54]: Landing_Outcome
```

No attempt
Success (ground pad)
Success (ground pad)
Success (drone ship)
Success (ground pad)
Success (drone ship)
Success (drone ship)
Success (ground pad)
Failure (drone ship)
Success (drone ship)
Success (drone ship)
Failure (drone ship)
Failure (drone ship)
Success (ground pad)
Controlled (ocean)
Failure (drone ship)

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

Section 3

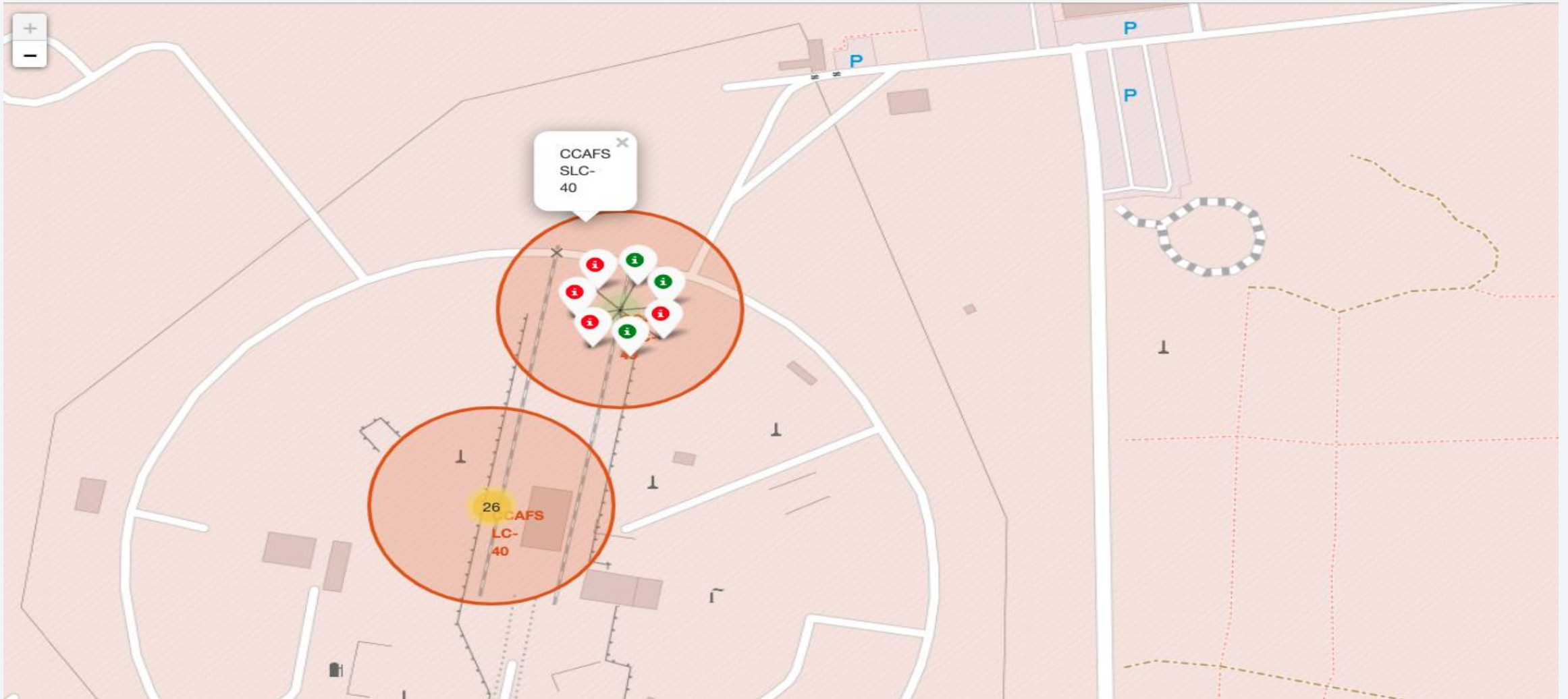
Launch Sites Proximities Analysis

All launch sites global map markers

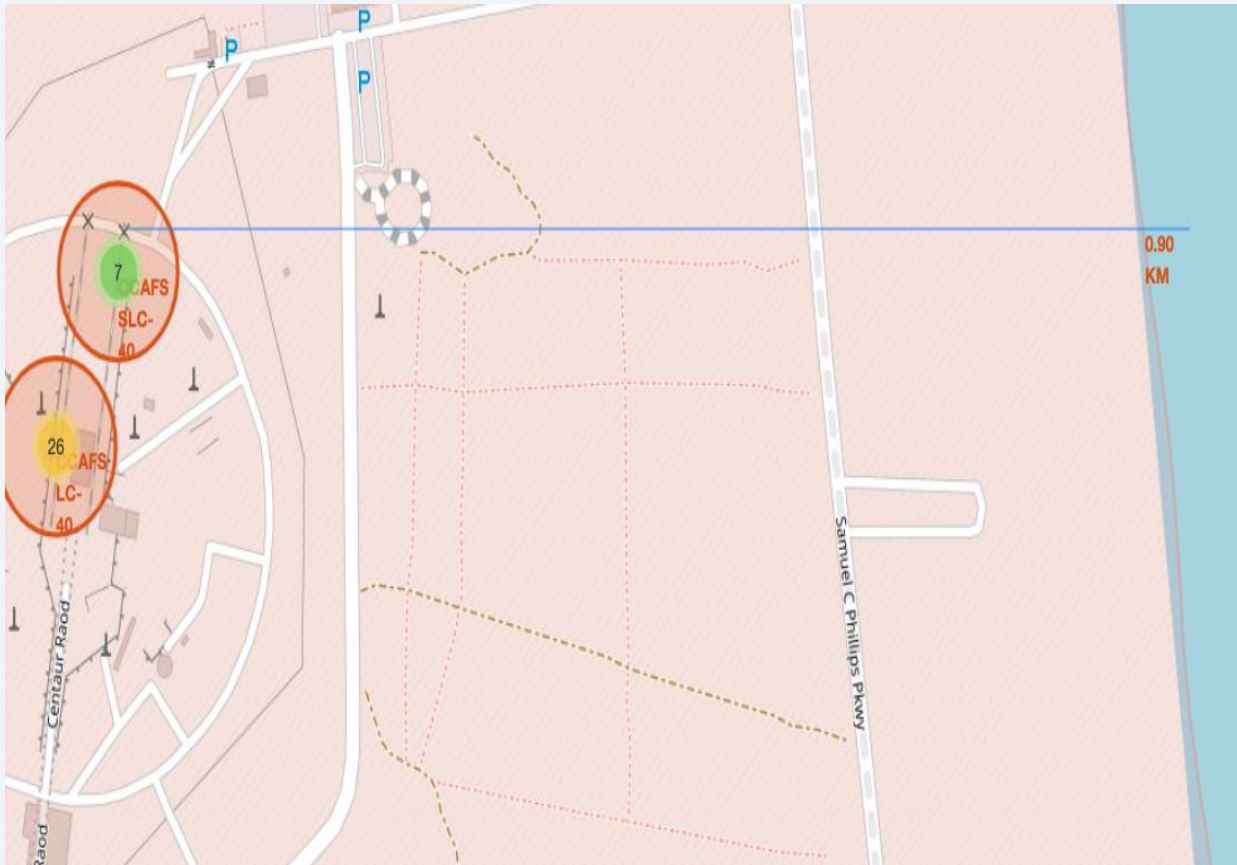


We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

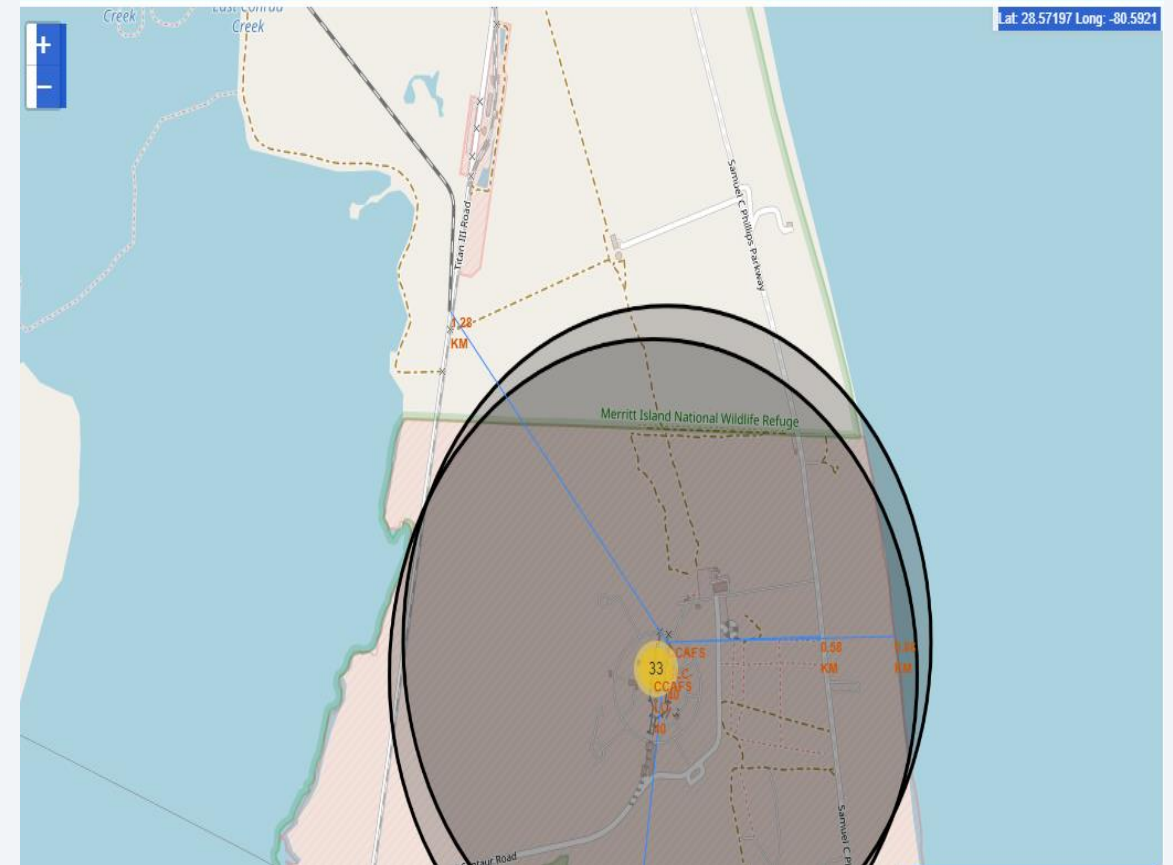
Markers showing launch sites with color labels



Launch Site distance to landmarks



- Coastline to launchpad distance



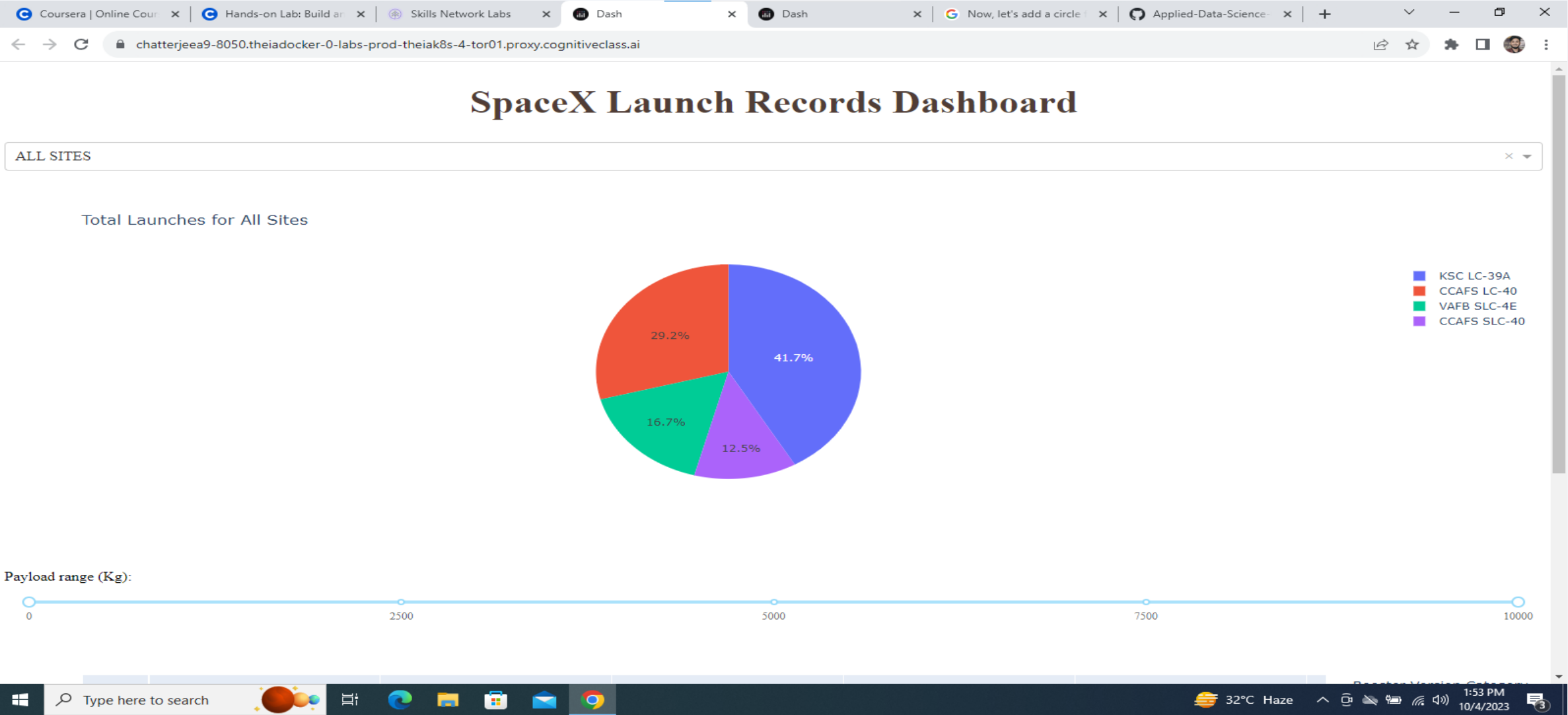
- Rail track to launchpad distance



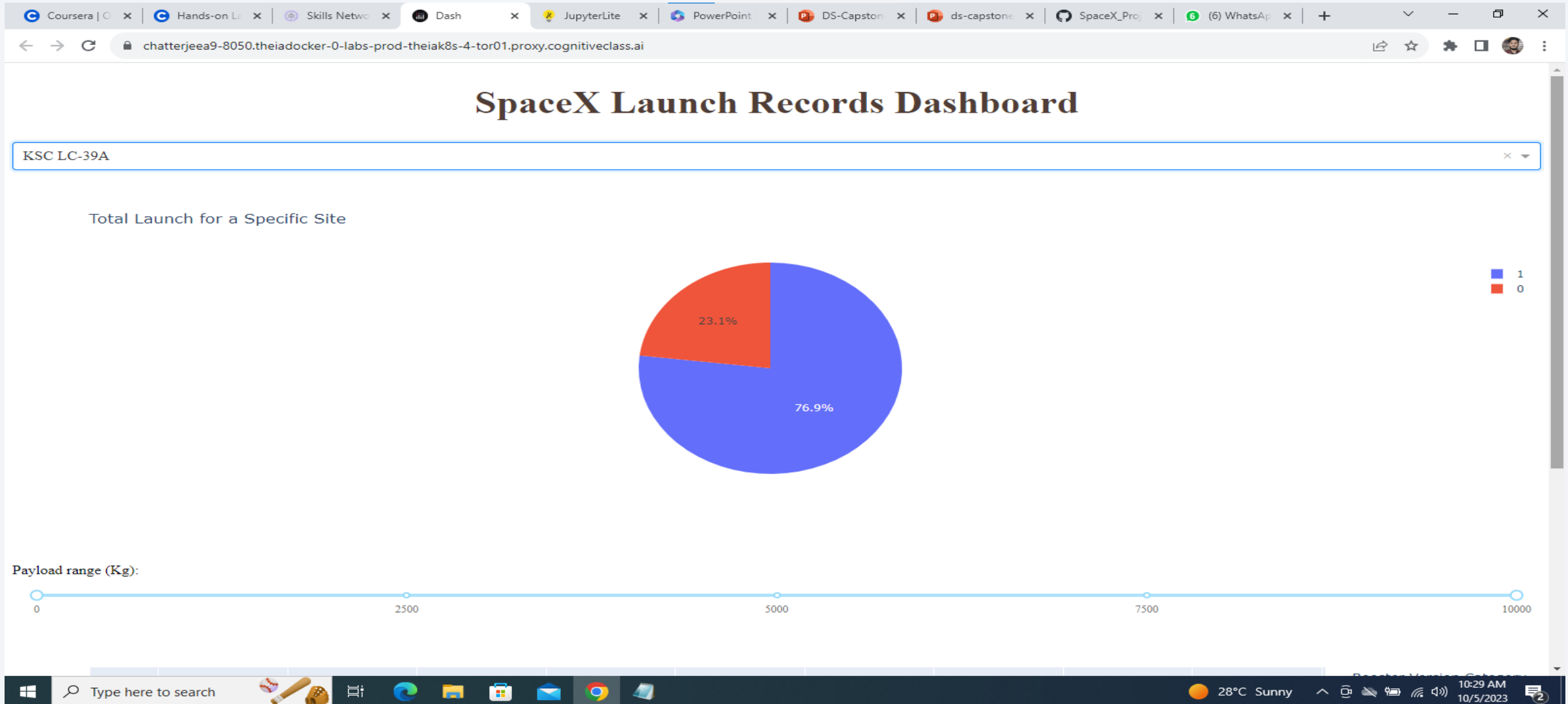
Section 4

Build a Dashboard with Plotly Dash

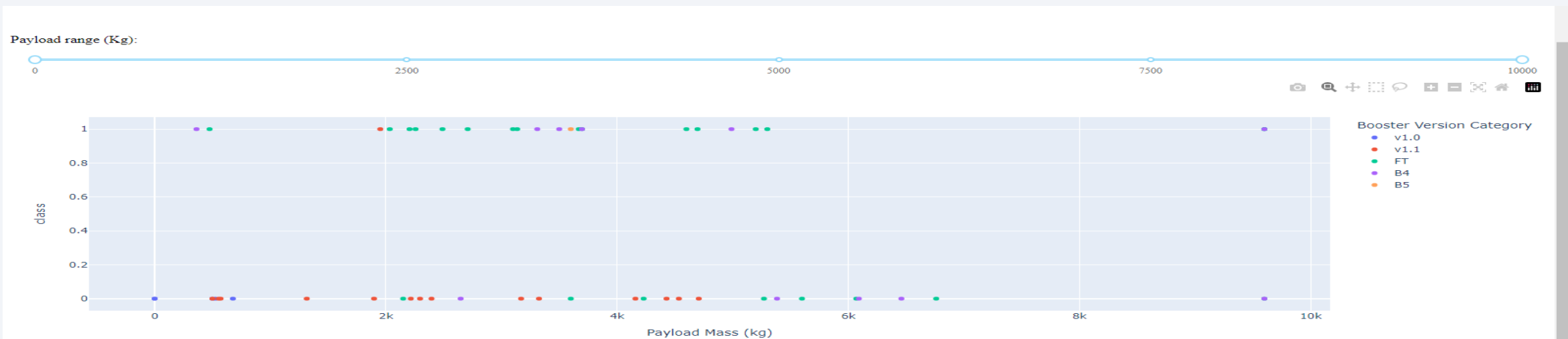
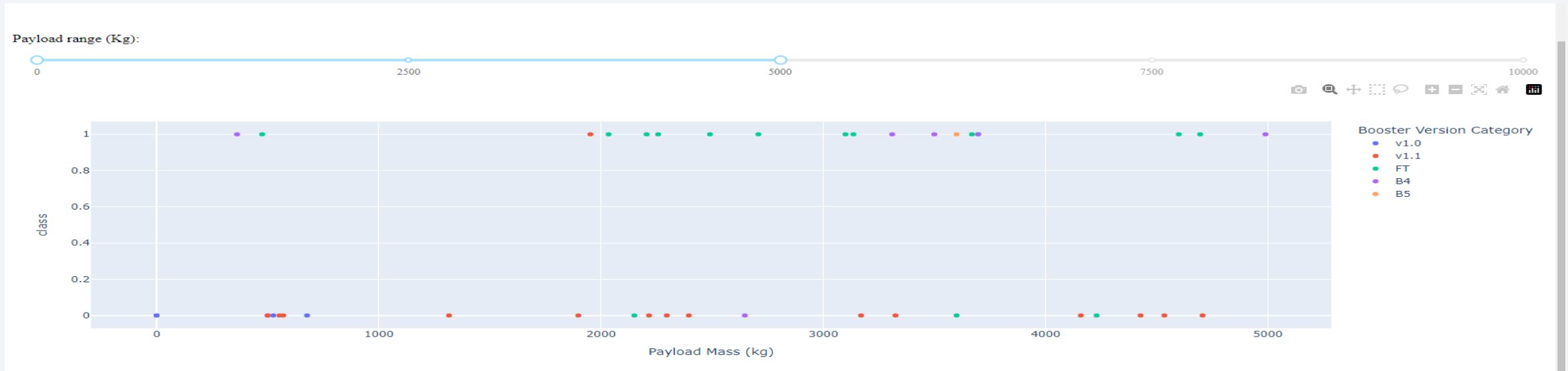
Pie chart showing the success percentage achieved by each launch site



Pie chart showing the Launch site with the highest launch success ratio



Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider





Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy

TASK 12

Find the method performs best:

```
[32]: Report = pd.DataFrame({'Method' : ['Test Data Accuracy']})
      knn_accuracy = knn_cv.score(X_test,Y_test)
      svm_accuracy = svm_cv.score(X_test,Y_test)
      Decision_tree_accuracy=tree_cv.score(X_test, Y_test)
      Logistic_Regression=logreg_cv.score(X_test, Y_test)

      Report['Logistic_Reg'] = [Logistic_Regression]
      Report['SVM'] = [svm_accuracy]
      Report['Decision Tree'] = [Decision_tree_accuracy]
      Report['KNN'] = [knn_accuracy]

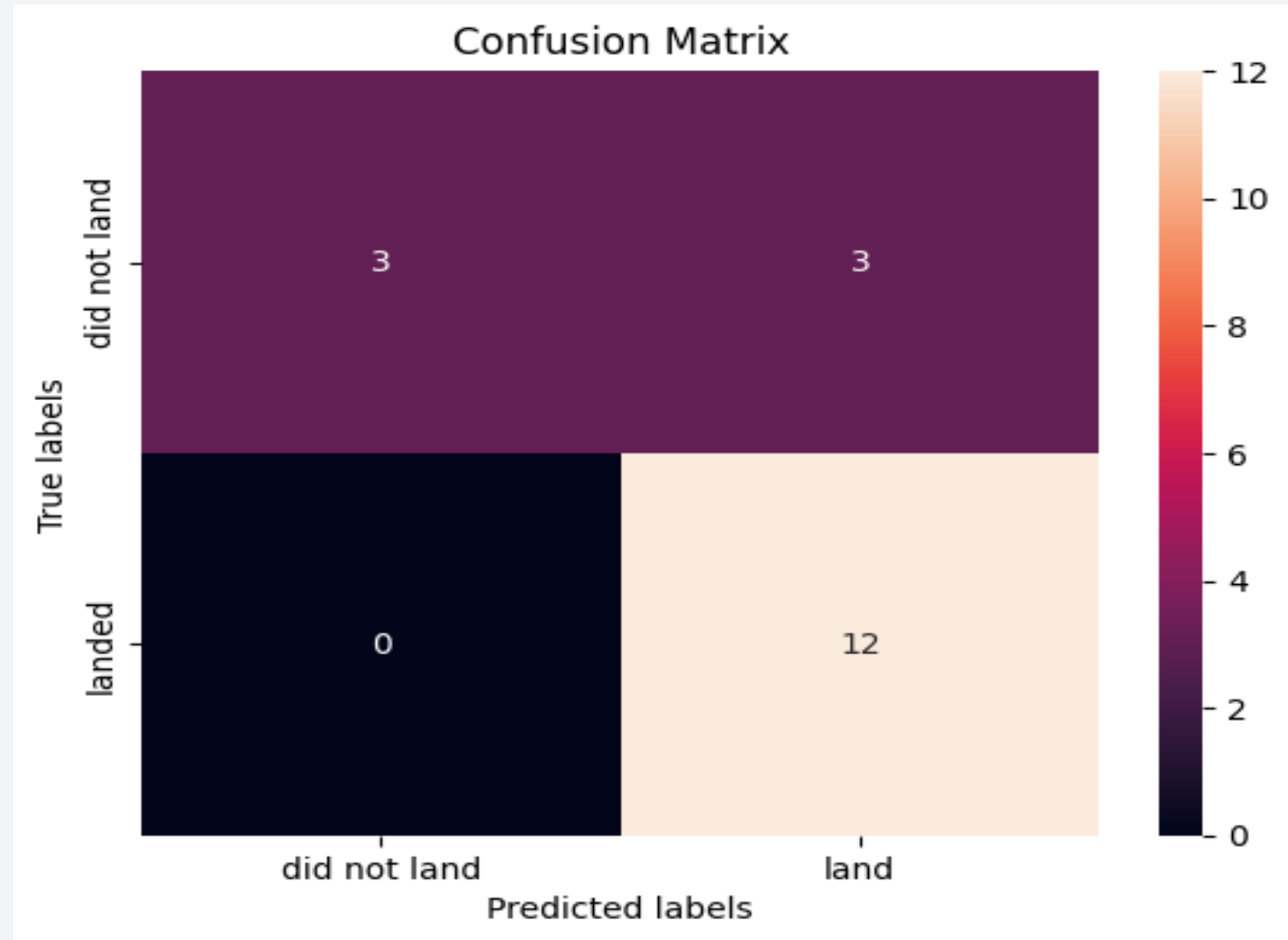
      Report.transpose()
```

```
[32]:
```

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

Confusion Matrix

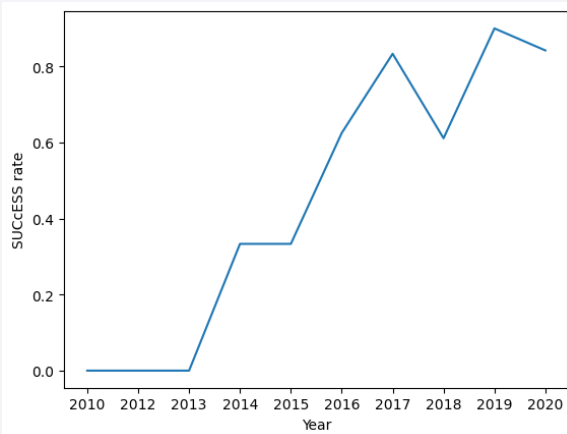
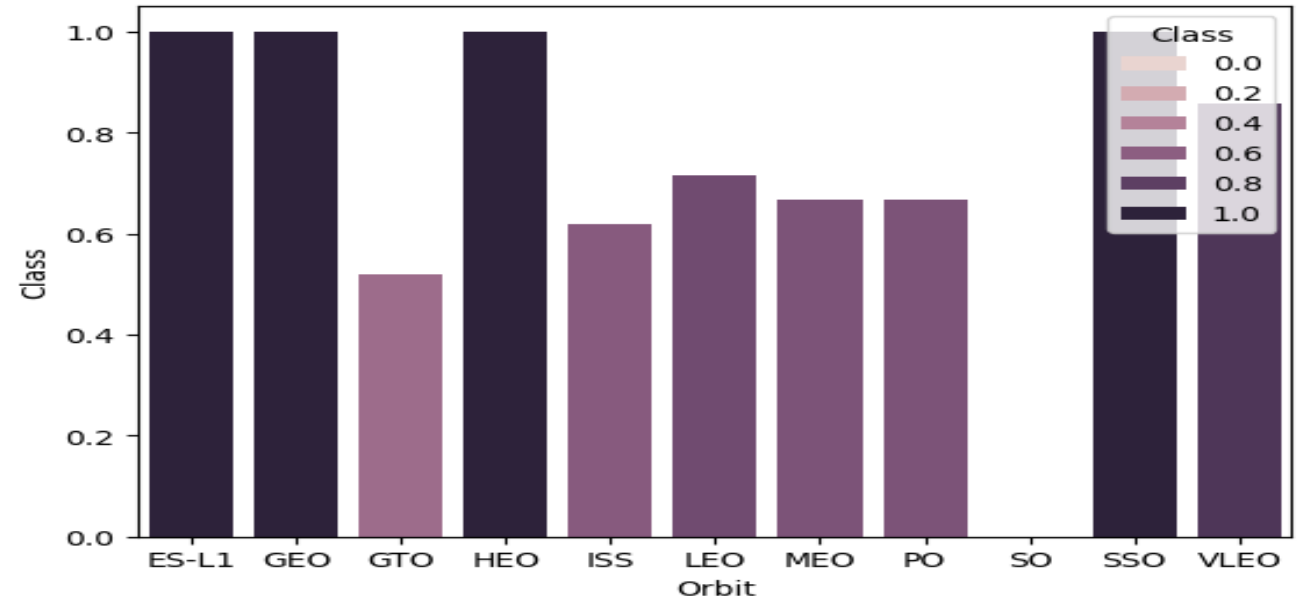
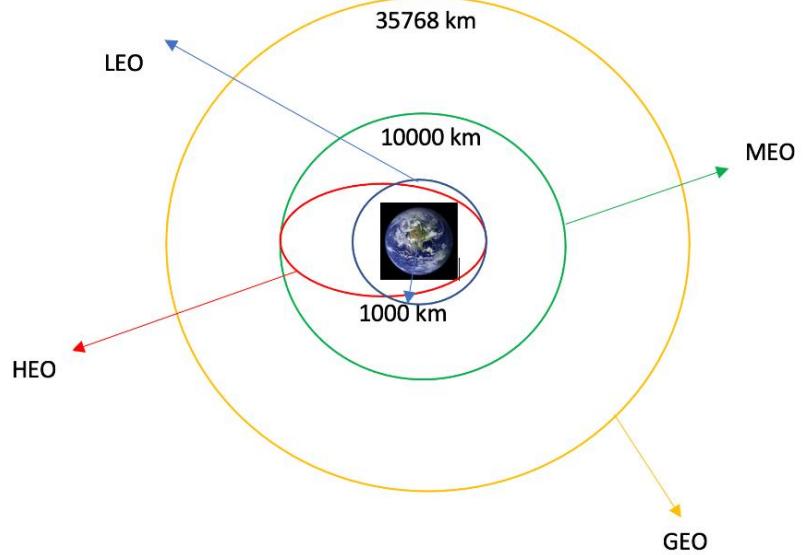
- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

- We can conclude that:
- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The best machine learning algorithm for this task

Appendix



TASK 12

Find the method performs best:

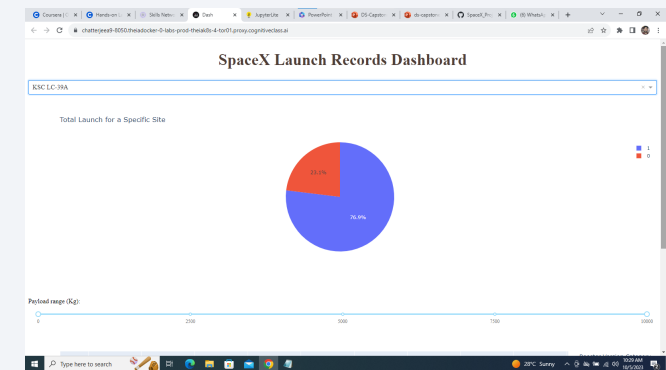
```
[32]: Report = pd.DataFrame({'Method' : ['Test Data Accuracy']})
      knn_accuracy = knn_cv.score(X_test,Y_test)
      svm_accuracy = svm_cv.score(X_test,Y_test)
      Decision_tree_accuracy=tree_cv.score(X_test, Y_test)
      Logistic_Regression=logreg_cv.score(X_test, Y_test)

      Report['Logistic_Reg'] = [Logistic_Regression]
      Report['SVM'] = [svm_accuracy]
      Report['Decision Tree'] = [Decision_tree_accuracy]
      Report['KNN'] = [knn_accuracy]

      Report.transpose()
```

```
[32]:
```

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



Thank you!

