IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

João Águas
26-Mar-2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection (SpaceX API, Webscrapping)
  - Data wrangling (Preparing data, dealing with null values, filtering the important data)
  - EDA - Data visualization
  - EDA – SQL
  - Geospatial data visualization with Folium
  - Interactive dashboard data visualization with Plotly Dash
  - Predictive analysis
- Summary of all results
  - EDA results
  - Interactive data visualization analysis
  - Predictive results analysis

Introduction

- SpaceX earned their success on the space market with the capacity to reuse the first stage of their launchers, this revolutionized space operations making it more affordable.
- SpaceY wants to understand the success rate of the SpaceX reusable first stage in order to improve the efficience of its offers. Hence the question what is the success rate of the reusable first stage, and what are the involved parameters of that success?

Section 1

Methodology

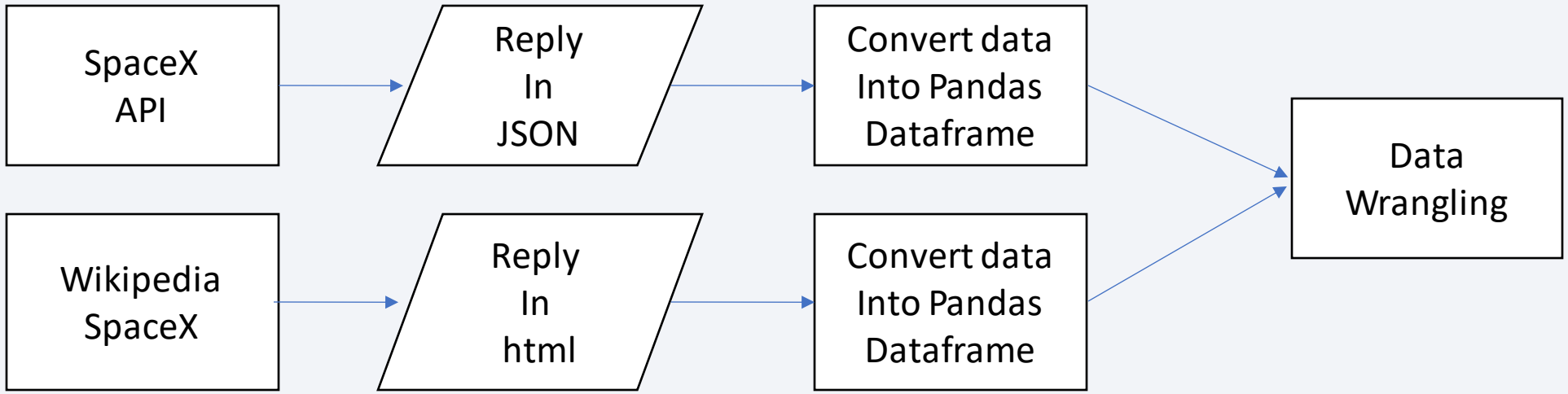
Methodology

Executive Summary

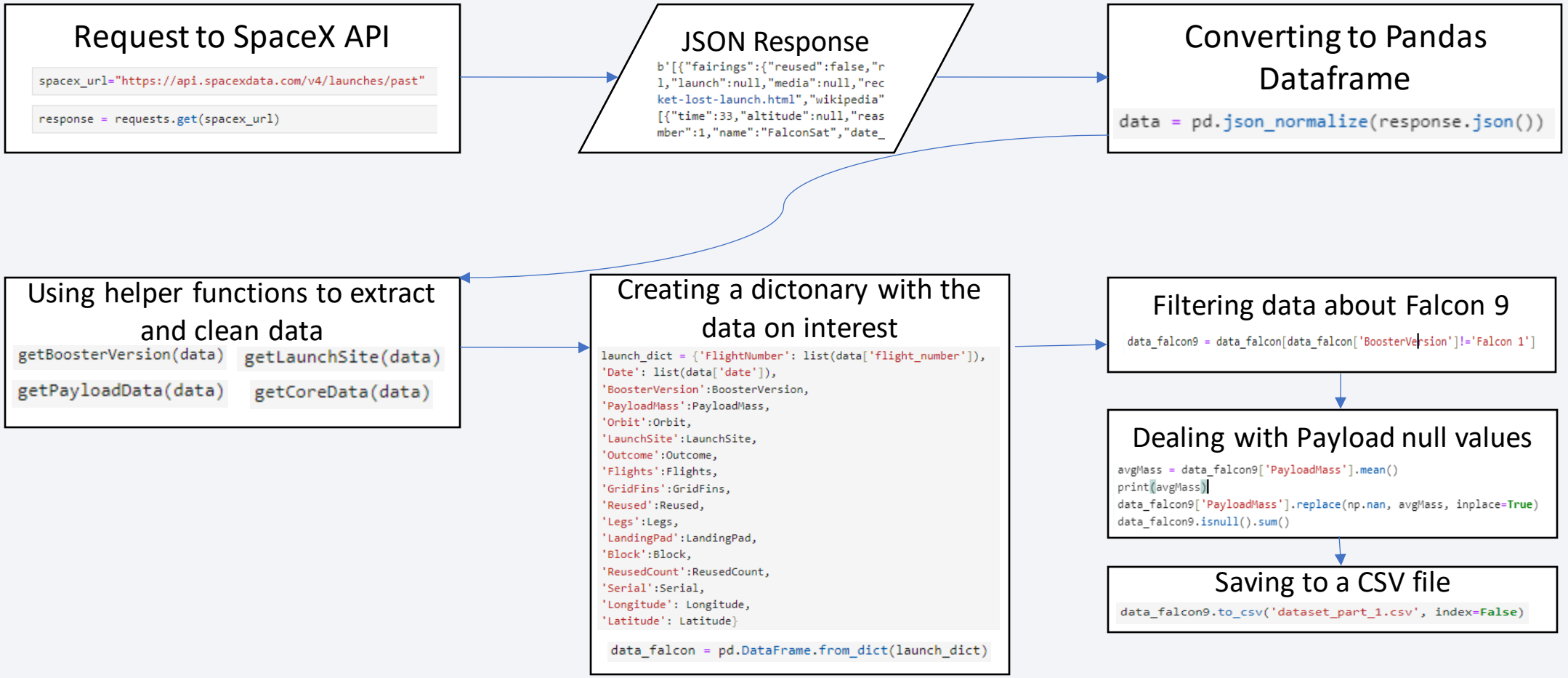
- Data collection methodology:
  - SpaceX API
  - Webscraping from Wikipedia
- Perform data wrangling
  - Selected falcon 9 launches, corrected null data, and used One Hot Encoding for the landing outcomes.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - KNN, LR, DT, and SVM with Gridsearch to optimize the models parameters.

Data Collection

- The data used in this project was gathered from:
  - The SpaceX API
  - Webscraping from the SpaceX Wikipedia page.



Data Collection – SpaceX API



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

Data Collection - Scraping

Sending and getting data from wikipedia

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
response = requests.get(static_url)
```

Creating beautifulsoup object and parsing

```
soup = BeautifulSoup(response.text, 'html.parser')
```

Detecting the tables

```
html_tables=soup.find_all('table', class_='wikitable')
```

Getting the names of the columns

```
column_names = []
for col in html_tables[1].find_all('th'):
    column_names.append(col.text.split('\n')[0])
```

Creating a dictonary with the data

```
launch_dict= dict.fromkeys(column_names)
# Remove an irrelvant column
del launch_dict['Date andtime (UTC)']
# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Creating data dictionaries

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)
            datatimelist=date_time(row[0])

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

Converting dictonary to Dataframe

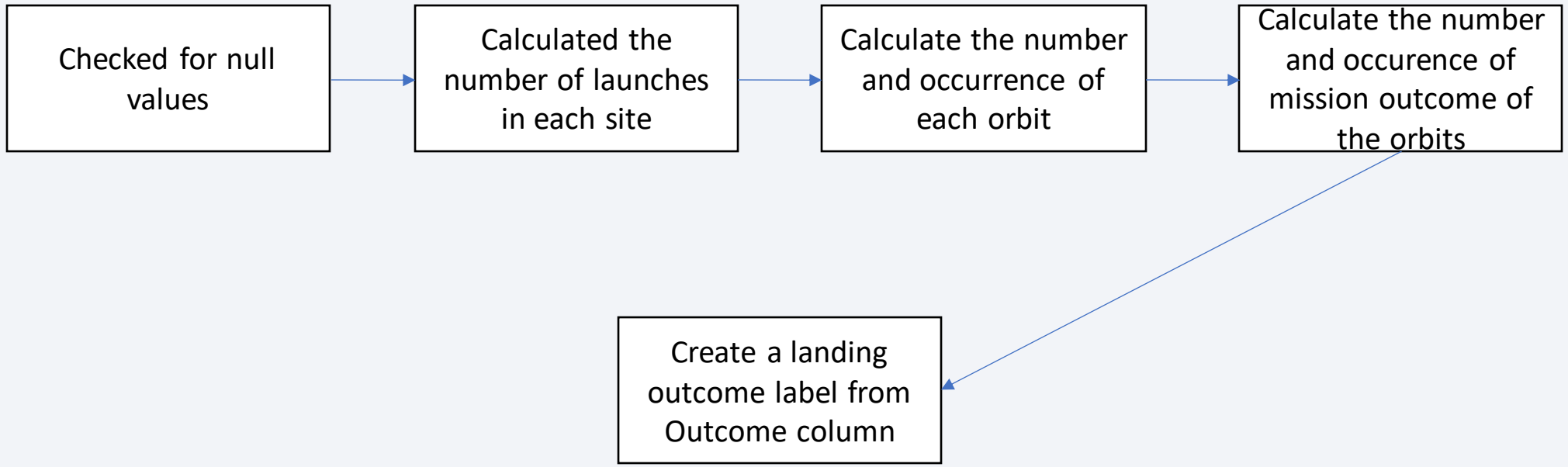
```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

Saving Dataframe to a CSV file

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

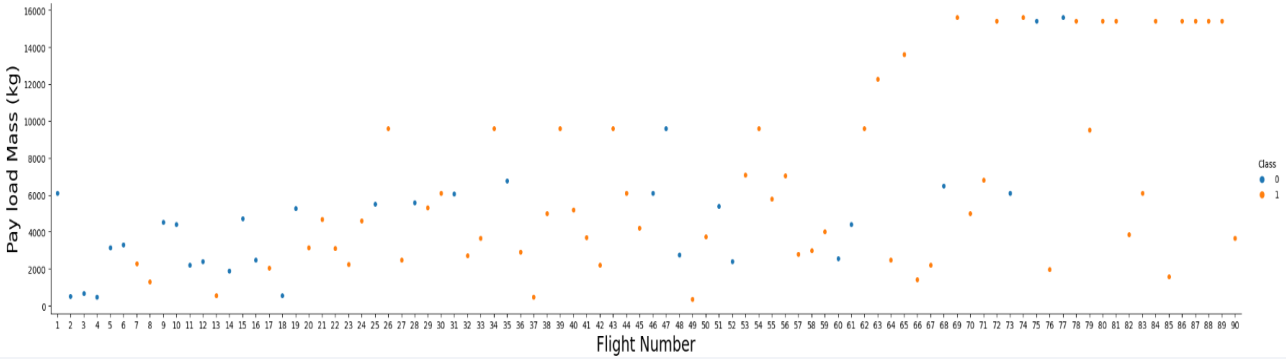
https://github.com/perseu/IBM_ML_Capstone_Project/blob/main/jupyter-labs-webscraping.ipynb

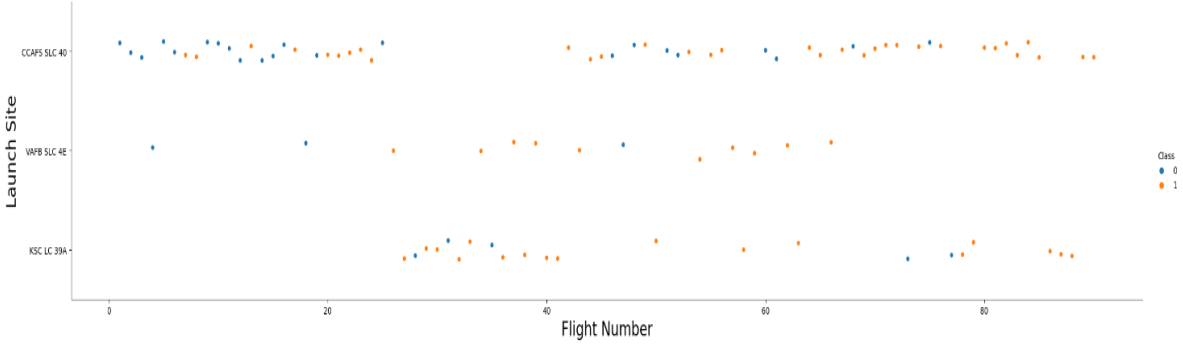
Data Wrangling

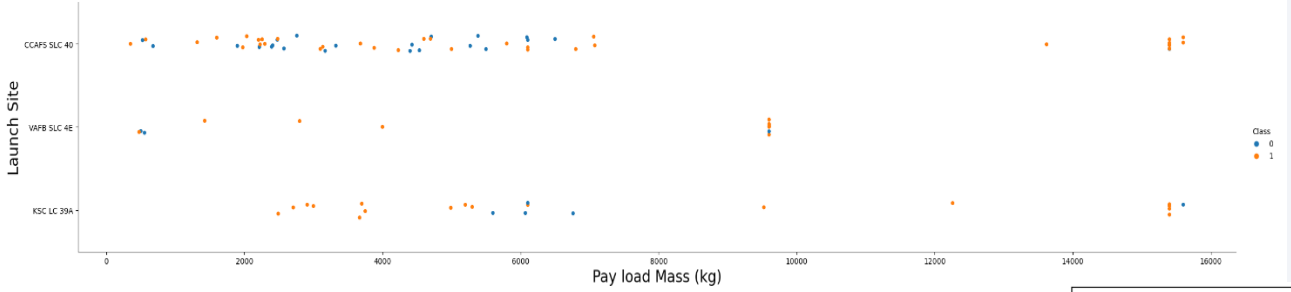


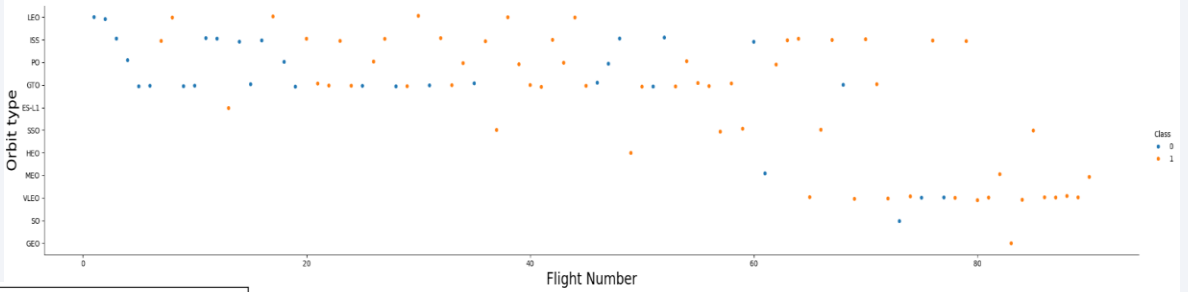
https://github.com/perseu/IBM_ML_Capstone_Project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

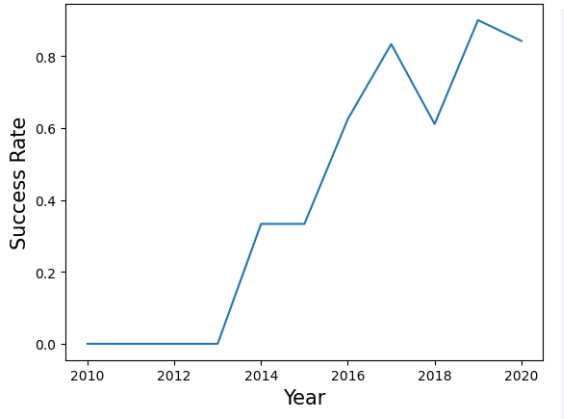








Note: Class = 0 (blue), means failure
Class = 1 (orange), means success



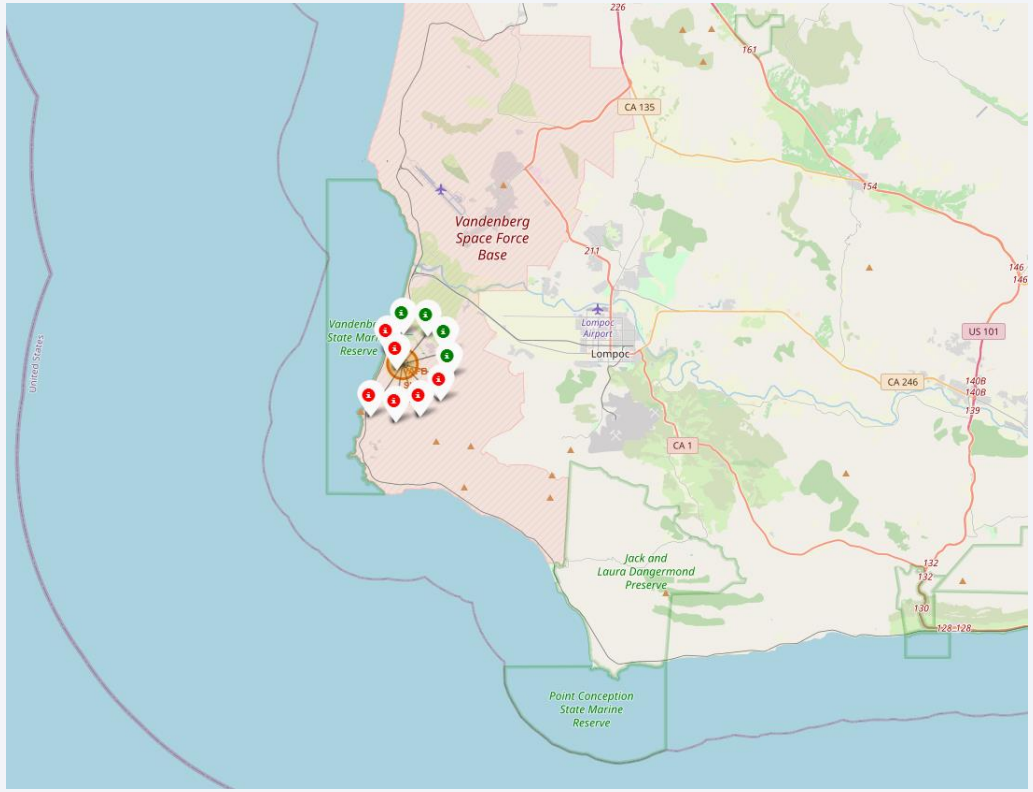
https://github.com/perseu/IBM_ML_Capstone_Project/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

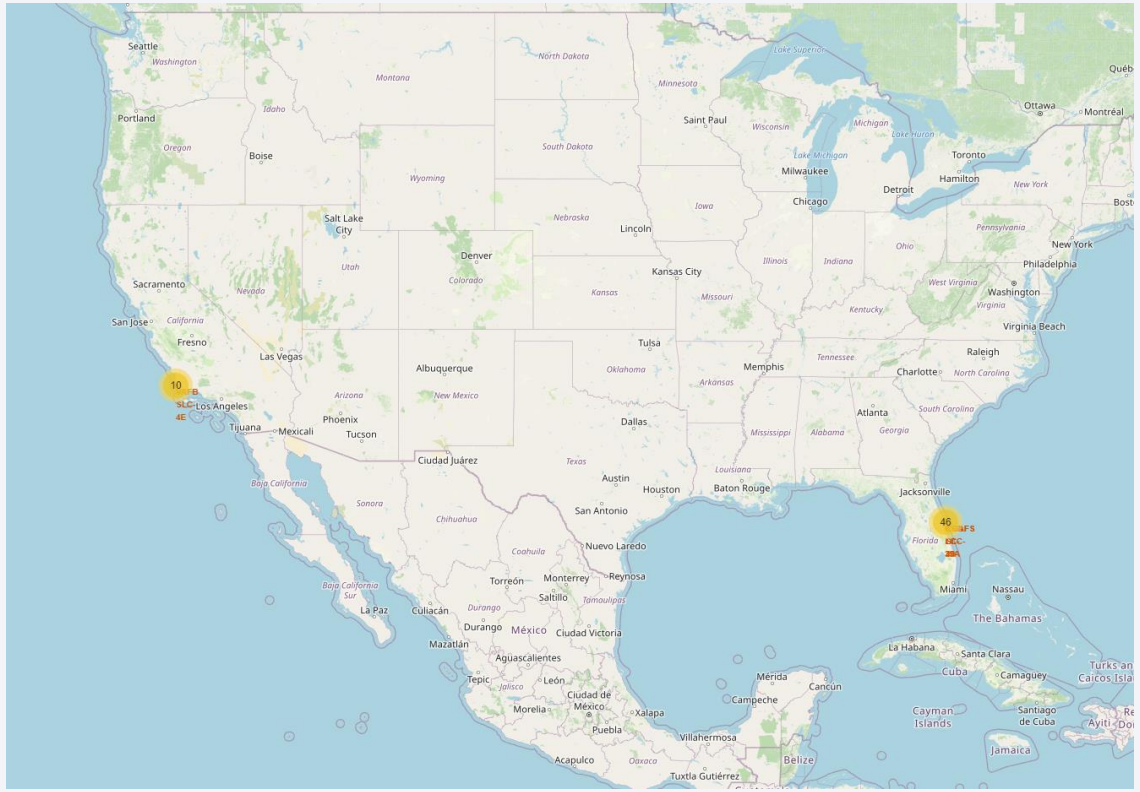
EDA with SQL

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

https://github.com/perseu/IBM_ML_Capstone_Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium



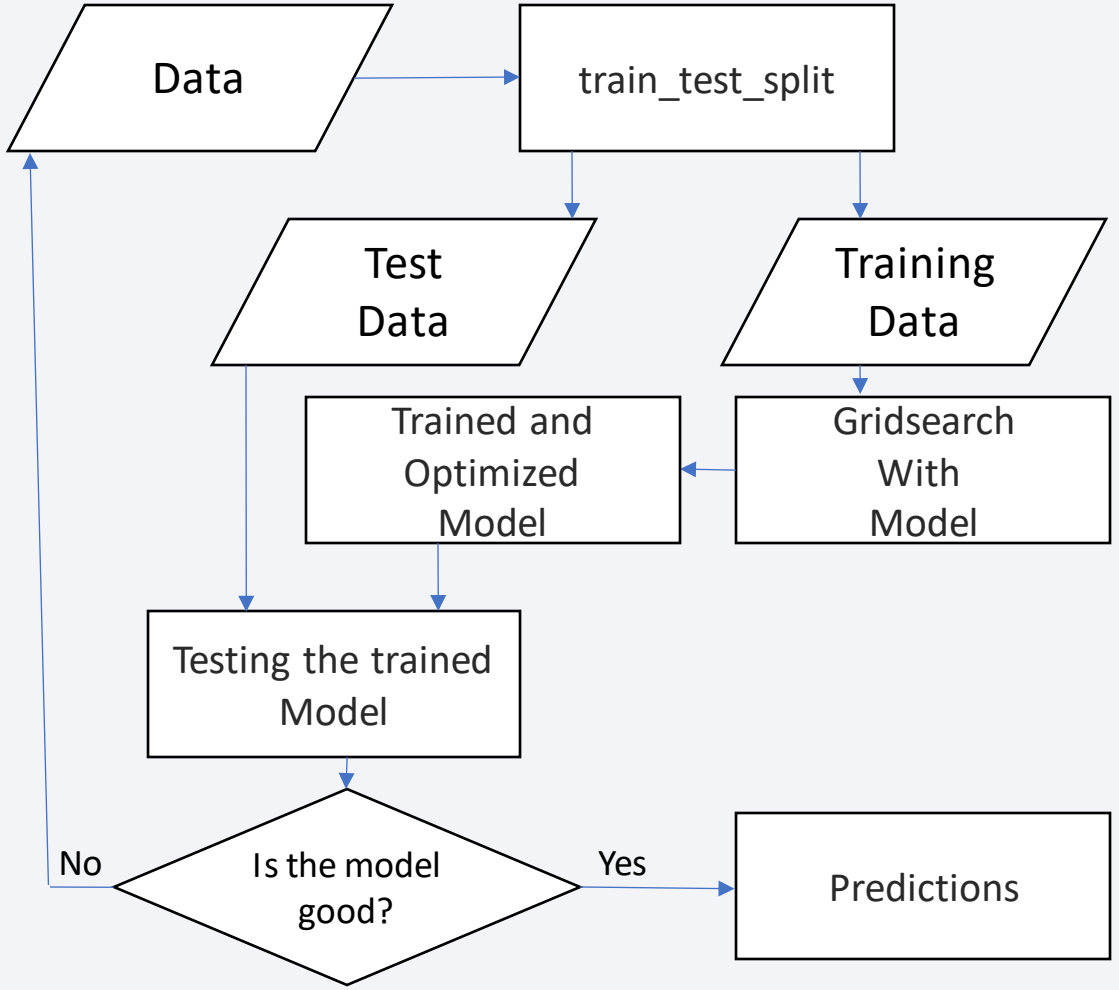


This interactive map shows the location of the launch sites. When a site is clicked it presents the success of the launches done by SpaceX on that particular site.

https://github.com/perseu/IBM_ML_Capstone_Project/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Predictive Analysis (Classification)

- Building and testing the models
  - Used Gridsearch with Logistic Regression, Decision Trees, Support Vector Machine, and K Nearest Neighbours to find the optimal parameters.
  - Used train_test_split to split the data into training and testing sets.
  - Trained the models with the training set.
  - Tested with the testing set.



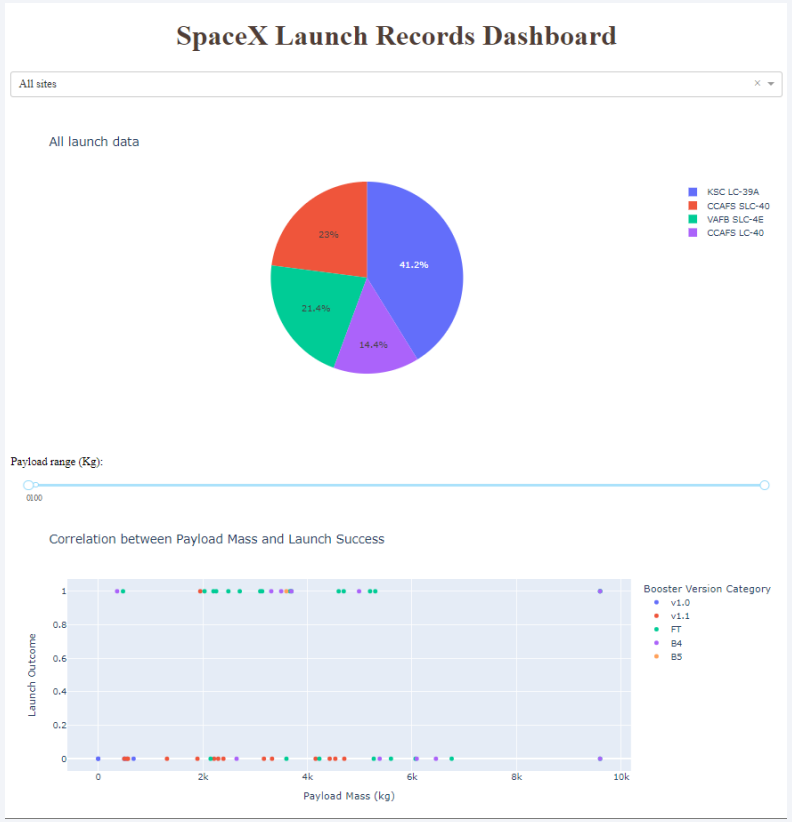
https://github.com/perseu/IBM_ML_Capstone_Project/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

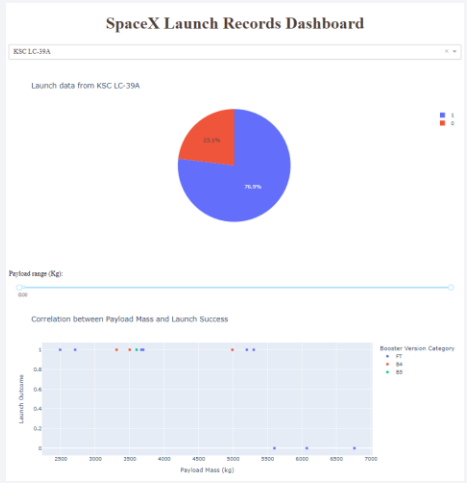
Results

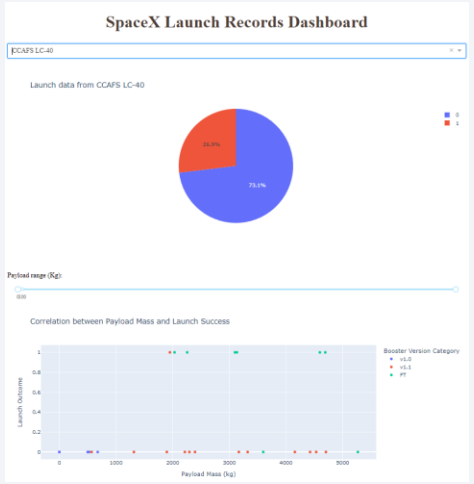
- Exploratory data analysis results
  - The payload mass per launch as increased with time.
  - The launch site with the highest success rate is VAFB SLC 4E, although it is the least used. The most used is CCAFS SLC 40, with the lowest success rate.
  - The site used to launch the greater mass payload was CCAFS SLC 40.
  - The orbits with highest success rates are: GEO, HEO, SSO, and ES-L1. Yet, these are the ones with lowest number of launches. (GEO, HEO, ES-L1 have only 1 launch, while SSO has 5)
  - The orbits with higher number of launches are VLEO (14 launches, 86% success rate) and LEO (7 launches, 71% success rate)
  - The highest payload mass was launched to a VLEO type orbit.
  - As a first order approximation we can see the success rate climbing since 2013 until 2020 to around 80% success rate.

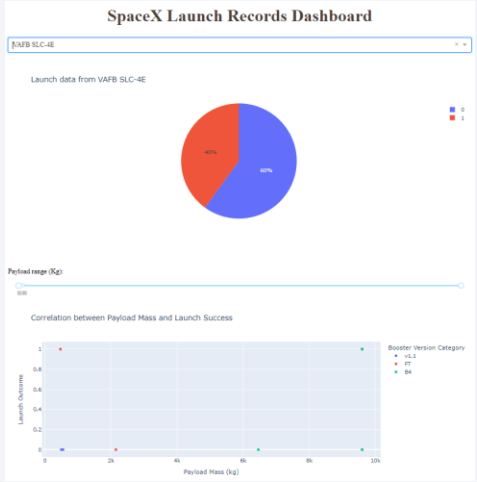
Results

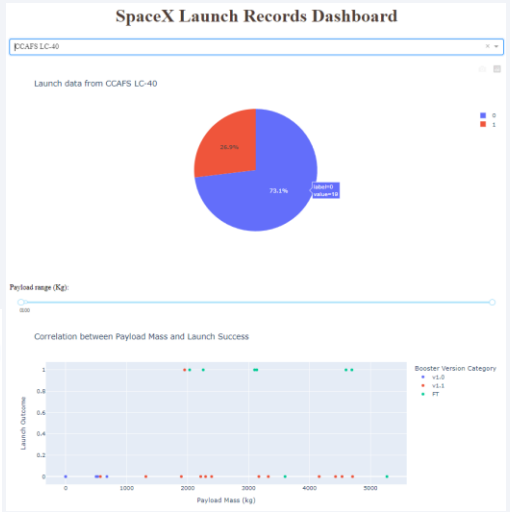
- Interactive analytics demo in screenshots









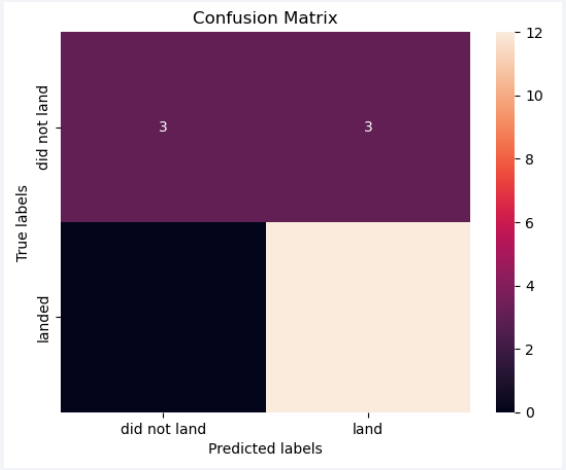


https://github.com/perseu/IBM_ML_Capstone_Project/blob/main/spacex_dash_app.py

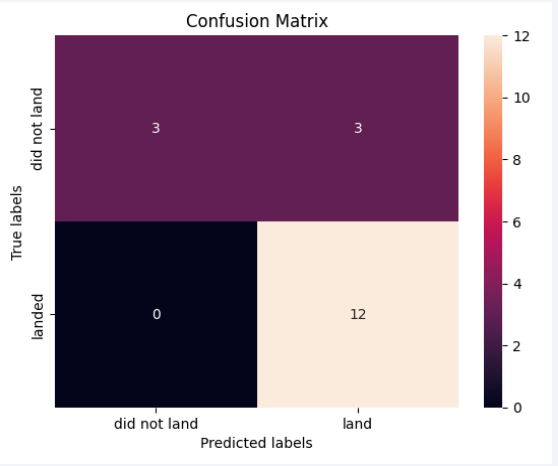
Results

- Predictive analysis results
  - Logistic Regression
    - Best parameters: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
    - Accuracy with test sample: 0.83
  - Support Vector Machine
    - Best parameters: {'C': 1.0, 'gamma': 0.0316, 'kernel': 'sigmoid'}
    - Accuracy with test sample: 0.85
  - Decision Tree
    - Best parameters: {'criterion': 'gini', 'max_depth': 14, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
    - Accuracy with test sample: 0.87
  - K-Nearest Neighbour
    - Best parameters: {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
    - Accuracy with test sample: 0.83

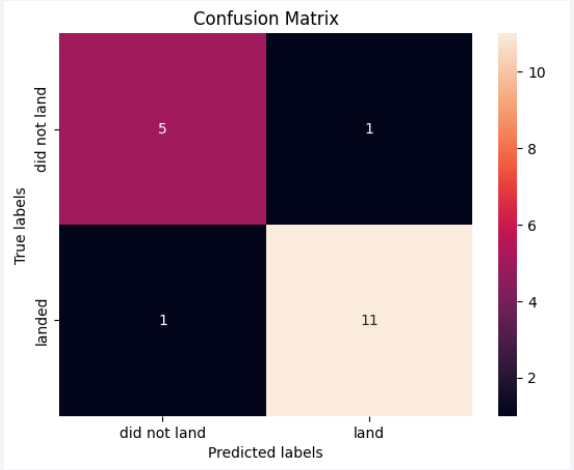
Logistic Regression



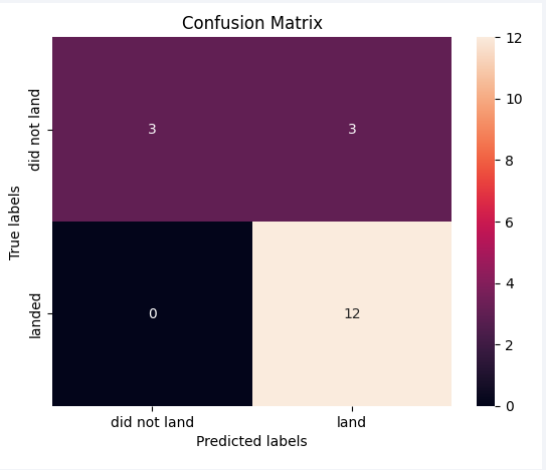
Support Vector Machine



Decision Tree

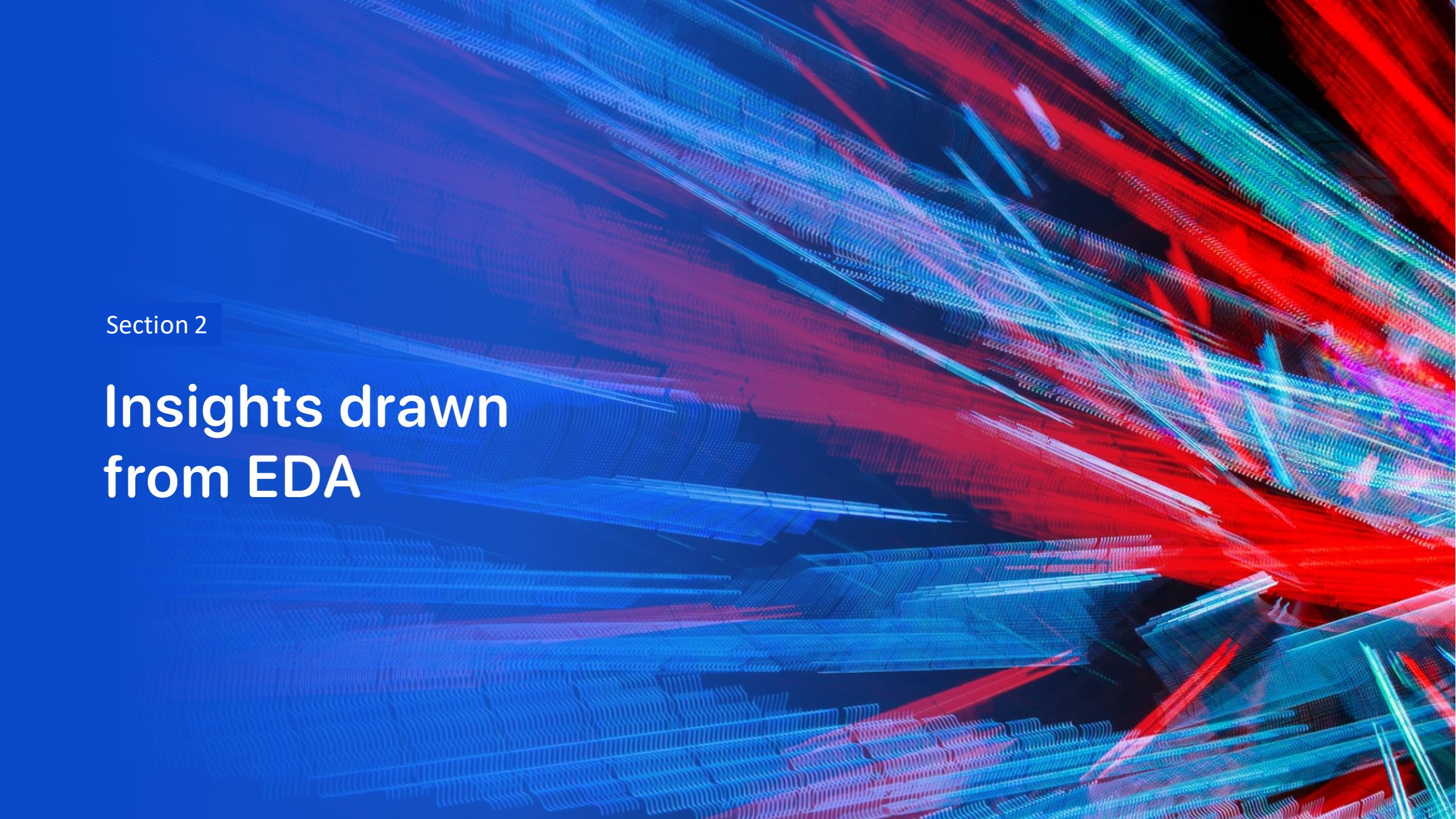


K-Nearest Neighbour

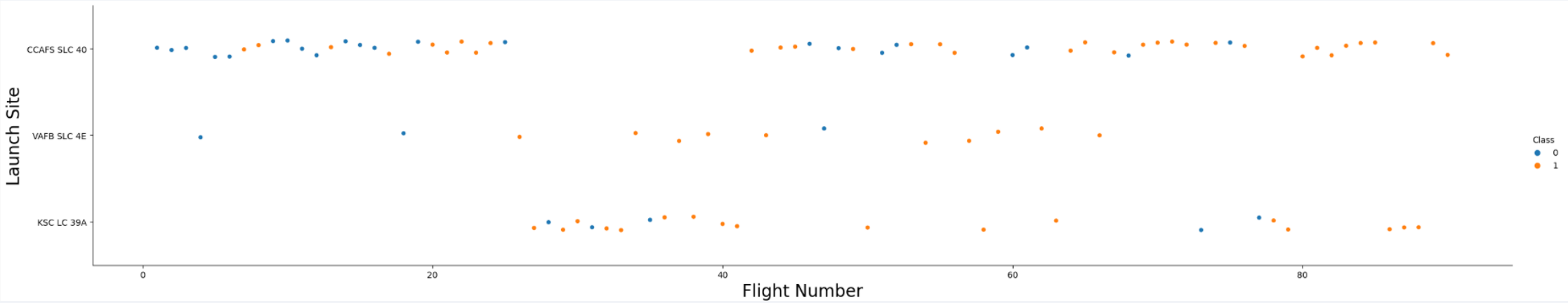


Section 2

Insights drawn from EDA

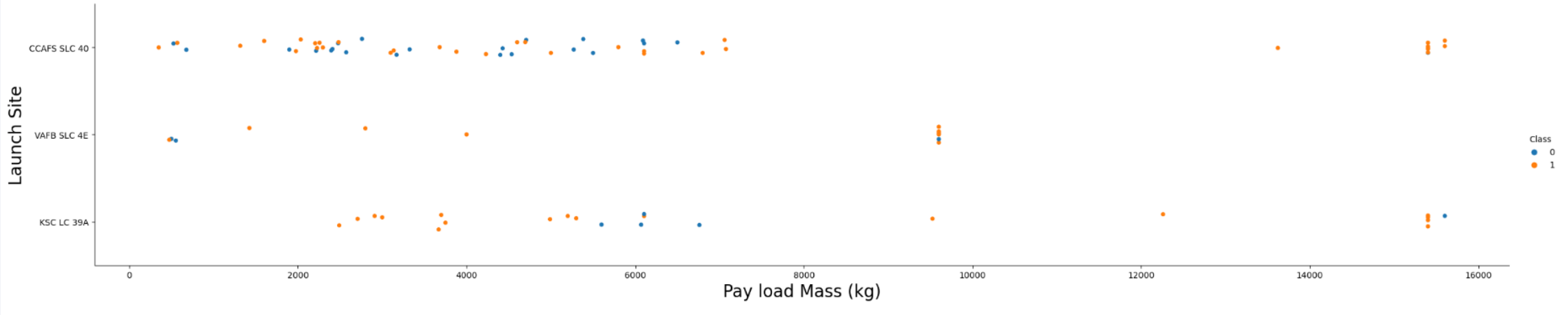


Flight Number vs. Launch Site



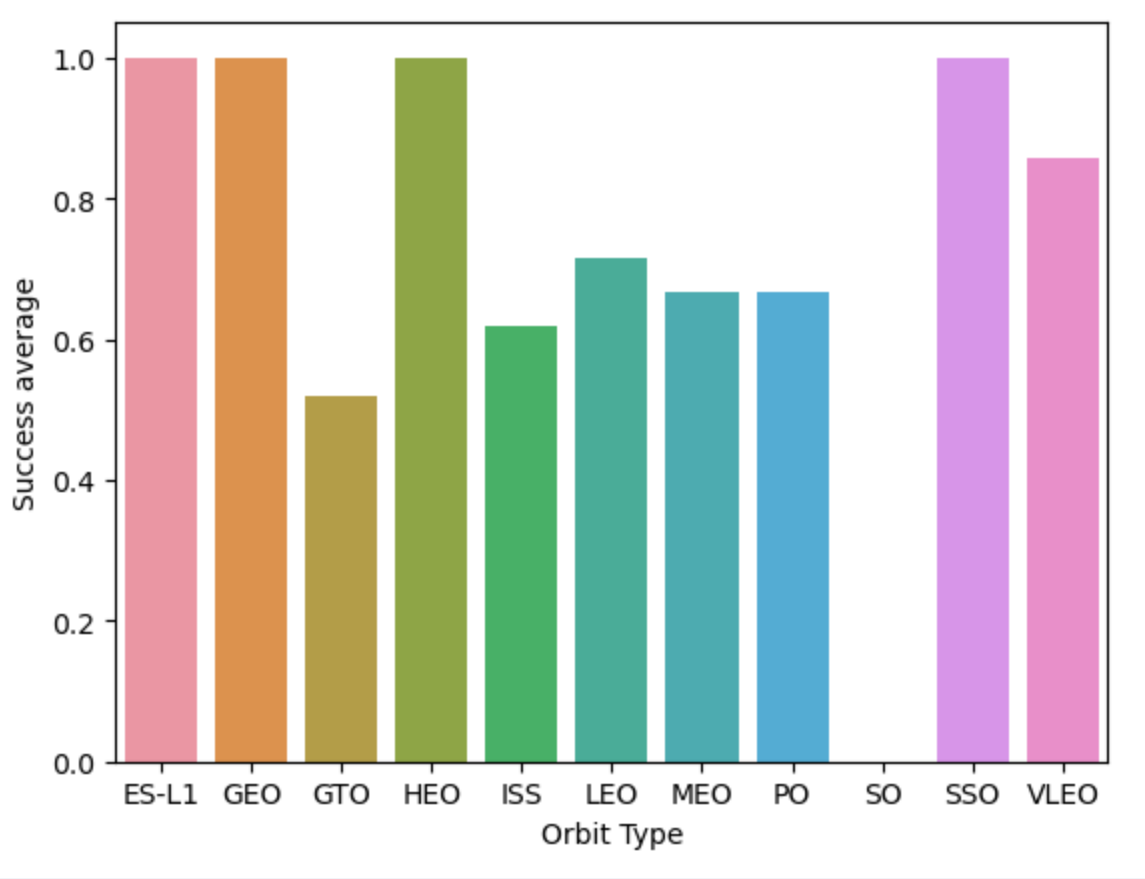
- The most used launch site was CCAFS SLC 40.
- The least used is VAFB SLC 4E.
- The most successful KSC LC 39A.
- The success rate increases with time on all launch sites.

Payload vs. Launch Site



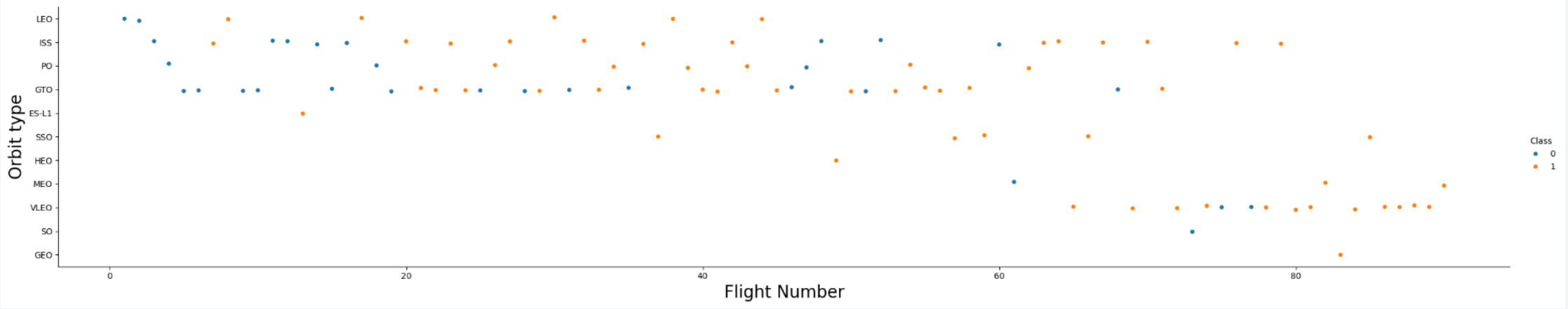
- The launch site used to deliver the greatest payload was CCAFS SLC 40.
- The highest payload delivered was 15600 kg.

Success Rate vs. Orbit Type



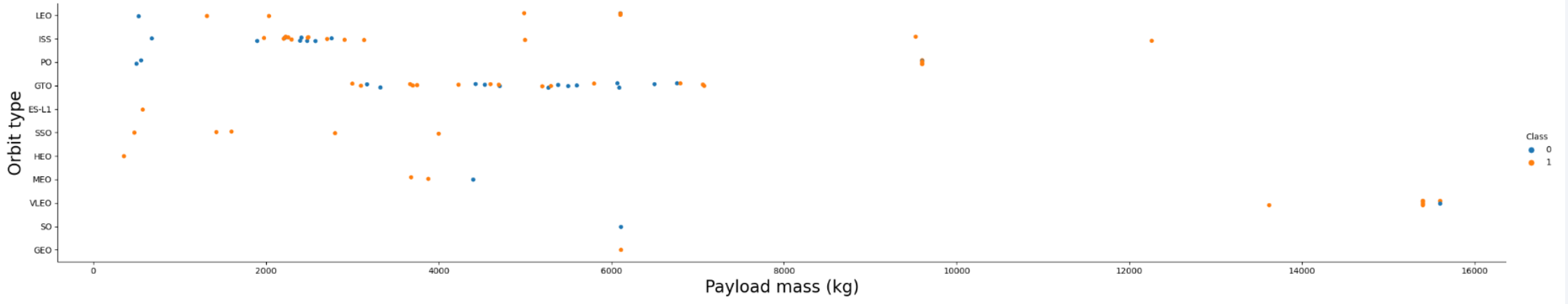
- The most successful orbits are ES-L1, GEO, HEO, and SSO. Yet, these orbits have only 1 launch.
- The orbits with higher success rate and more launches are VLEO and LEO

Flight Number vs. Orbit Type



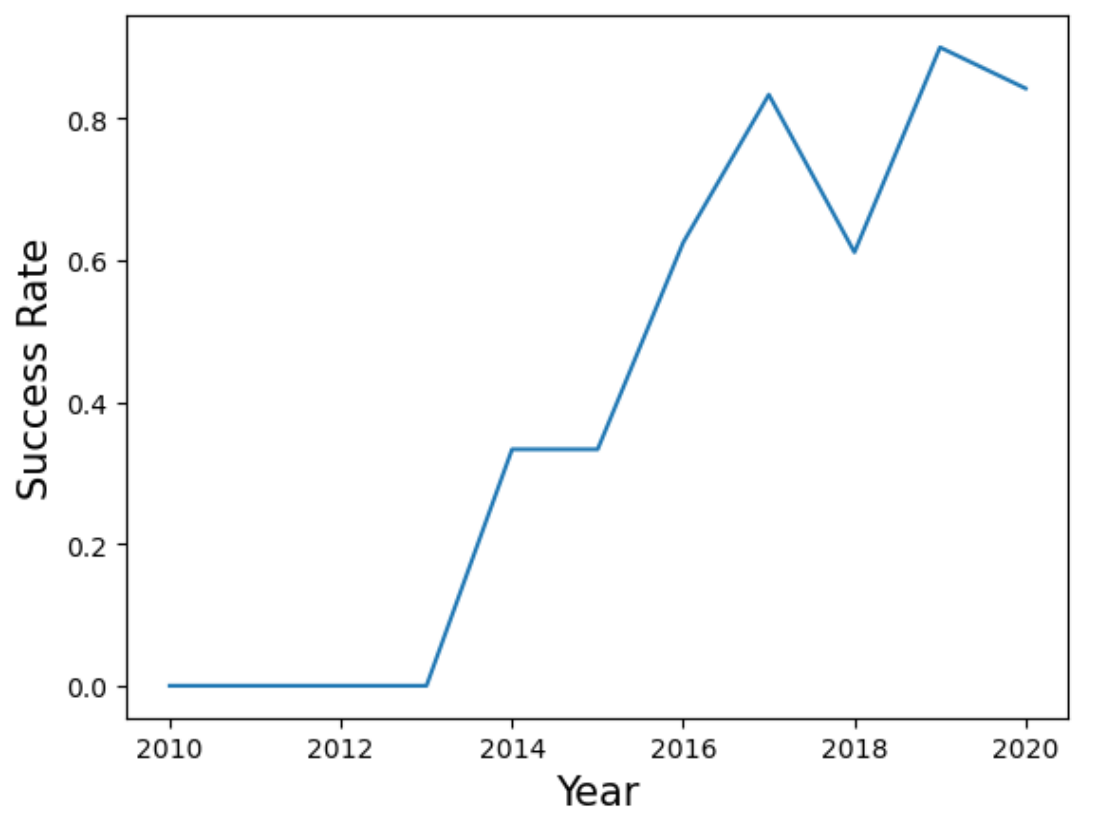
- The first successful lauch was to the ISS.
- The orbits with most launches are LEO and VLEO.
- VLEO orbit launches start much later.
- The 100% success rate orbit type launches contain only a single launch.

Payload vs. Orbit Type



- The highest mass payload was delivered to a VLEO orbit.

Launch Success Yearly Trend



- As a first order approach we may see a steady climb in the success rate since 2013
- There is a significant drop in success around 2018.

All Launch Site Names

```
%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

The SQL query is selecting the distinct values of the column Launch_Site that belongs to the table SPACEXTABLE

Launch Site Names Begin with '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

We're selecting everything from table SPACEXTABLE that has a substring CCA on the values of the column Launch_Site.

Total Payload Mass

```
%sql select SUM(PAYLOAD_MASS__KG_) as SUM_PAYLOAD_MASS_CRS from SPACEXTABLE where SPACEXTABLE.Payload like '%CRS%'
```

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

SUM_PAYLOAD_MASS_CRS
111268

Selecting all the entries from PAYLOAD_MASS__KG_ from the rows where the column Payload has the substring CRS, and then suming the selected PAYLOAD_MASS__KG_ values.

Average Payload Mass by F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) as 'Average Payload mass F9 v1.1' from SPACEXTABLE where SPACEXTABLE.Booster_Version like '%F9 v1.1%'
```

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

Average Payload mass F9 v1.1
2534.6666666666665

Calculating the average value from the PAYLOAD_MASS__KG_ column, for the rows selected on the Booster_Version that contain the substring 'F9 v1.1'.

First Successful Ground Landing Date

```
%sql select min(Date) as 'First Time' from SPACEXTABLE where SPACEXTABLE.Landing_Outcome == 'Success (ground pad)'
```

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

First Time
2015-12-22

From the Landing_Outcome column we select all that have the string 'Success (ground pad)', and then checking which of these row has the minimum value on the column 'Date'.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select distinct(Booster_Version) from (select Booster_Version, PAYLOAD_MASS__KG_, Landing_Outcome from SPACEXTABLE where SPACEXTABLE.PAYLOAD_MASS__KG_ between 4000 and 6000) where Landing_Outcome == 'Success (drone ship)'
```

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

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

We start with a nested query where we select Booster_Version, PAYLOAD_MASS__KG_ and Landing_Outcome, where the PAYLOAD_MASS__KG_ column has values between 4000 and 6000, and from that result we filter by the Landing_Outcome where its value is 'Success (drone ship)'

Total Number of Successful and Failure Mission Outcomes

```
%sql select sum(case when Mission_Outcome like '%uccess%' then 1 else 0 end) as 'Success Count', sum(case when Mission_Outcome like '%ailure%' then 1 else 0 end) as 'Failure Count' from SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

Success Count	Failure Count
100	1

Selecting the sum of two cases, when the string 'uccess' is found it adds 1 to the column 'Success Count', otherwize adds 0. The same is done for the new column 'Failure Count' when the substring 'ailure' is found adds 1, otherwise adds 0.

Boosters Carried Maximum Payload

```
%sql select distinct(Booster_Version), (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE) from SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

Booster_Version	(select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)
F9 v1.0 B0003	15600
F9 v1.0 B0004	15600
F9 v1.0 B0005	15600
F9 v1.0 B0006	15600
F9 v1.0 B0007	15600
F9 v1.1 B1003	15600
F9 v1.1	15600

Starting by a nested query where we select all rows with maximum value of payload, then we select the distinct Booster_Version values from that table that carry the maximum payload.

2015 Launch Records

```
%sql select substr(Date,6,2) as month, substr(Date,0,5) as year, Booster_Version, Launch_Site, Landing_Outcome from SPACEXTABLE where year == '2015' and SPACEXTABLE.Landing_Outcome == 'Failure (drone ship)'
```

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

month	year	Booster_Version	Launch_Site	Landing_Outcome
01	2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

We're selecting the month, year, Booster_Version, Launch_Site and Landing_outcome columns, but where the year corresponds to '2015' and the Landing_Outcome is 'Failure (drone ship)'.
We're selecting the failed attempts to land on the drone ship.

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

```
%sql select Landing_Outcome as 'Landing Outcome', count(*) as 'Counts' from (select * from SPACEXTABLE where Date <= '2017-03-20' and Date >= '2010-06-04') group by Landing_Outcome
```

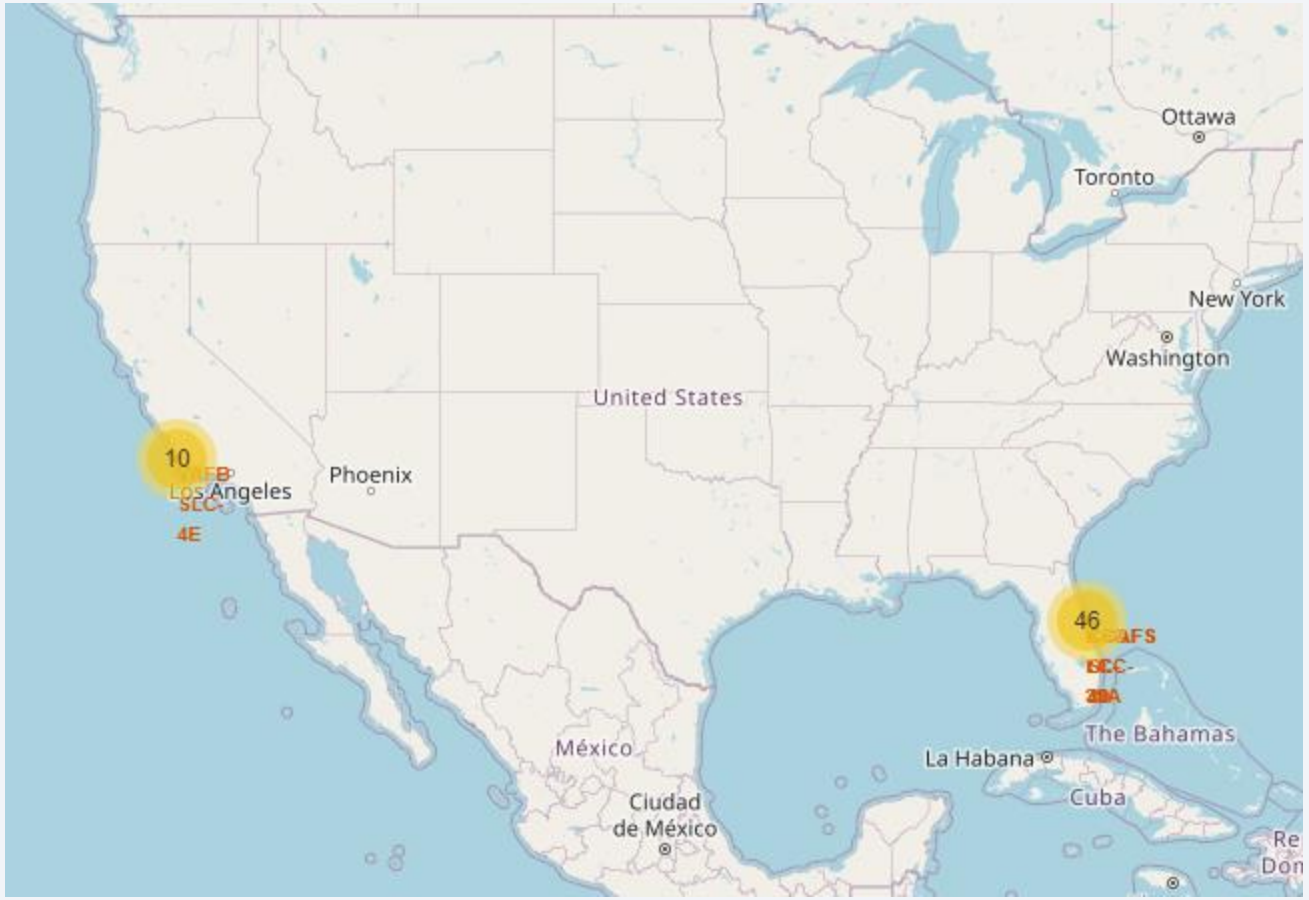
```
 * sqlite:///my_data1.db
Done.
```

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

We use a nested query to first select the time interval that we want to observe, then that result is used to count the number of times that a Landing Outcome happened, and it is grouped by Landing Outcome for it to work.


Section 3
Launch Sites
Proximities Analysis

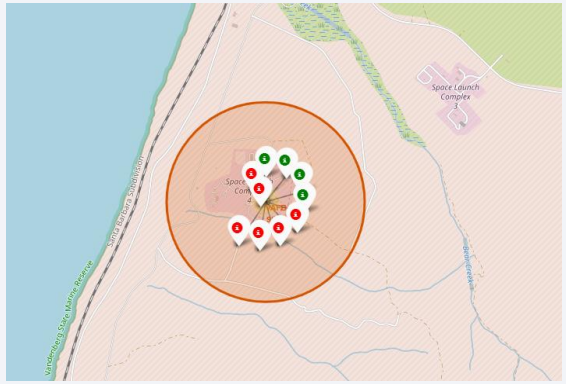
Launch site locations



- VAFB-SLC-4E is in the West Coast near Los Angeles
- KSC LC 39A, CCAFS LC 40, and CCAFS SLC 40 are located near Cape Canaveral

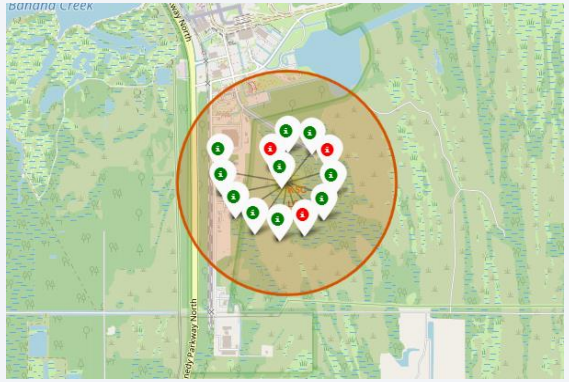
Launch site landings outcome

VAFB-SLC-4E

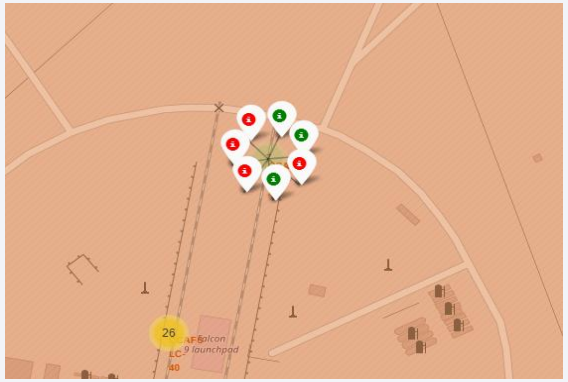


CCAFS LC 40

KSC LC 39A

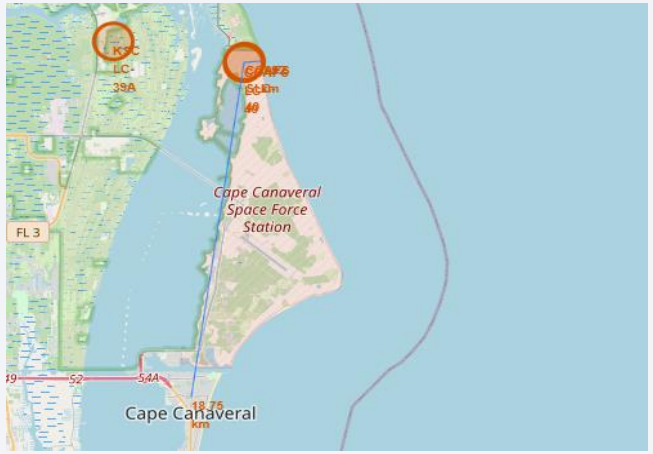


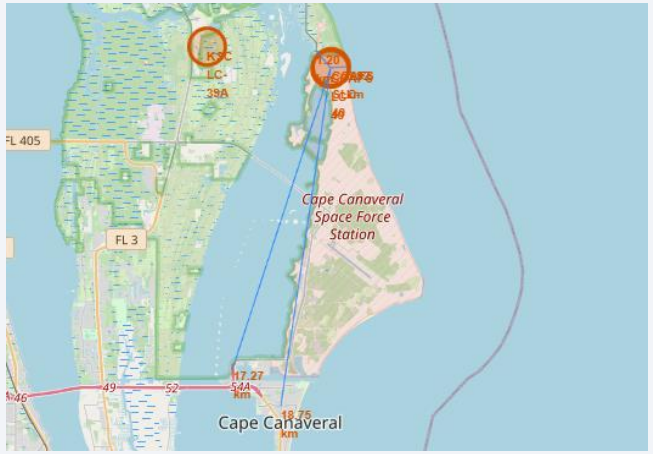
CCAFS SLC 40



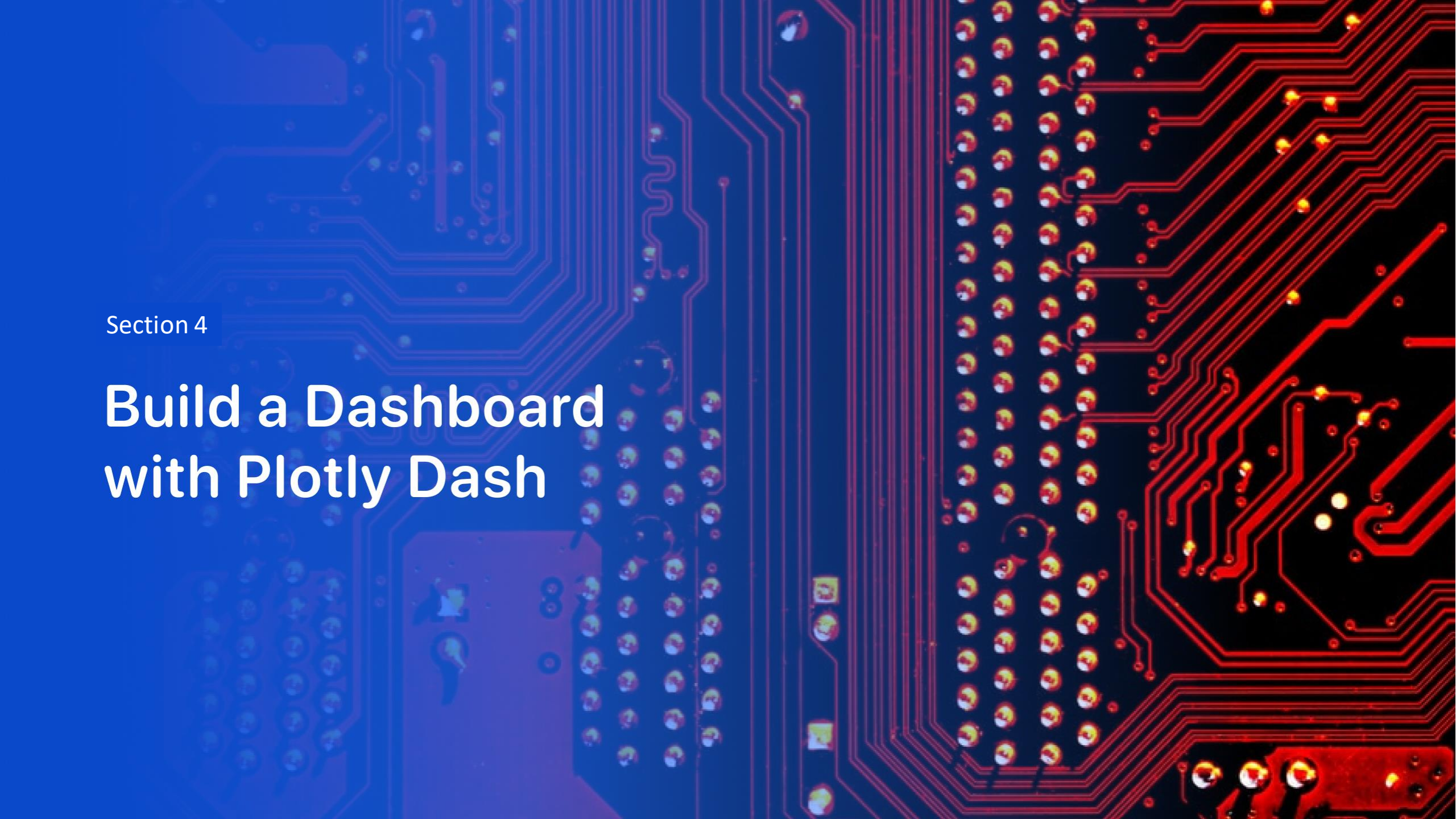
- The most used launch site is CCAFS LC 40, although it is not the most successful.
- The least used is CCAFS SLC 40
- The most successful launch site is KSC LC 39A

Distances of interest

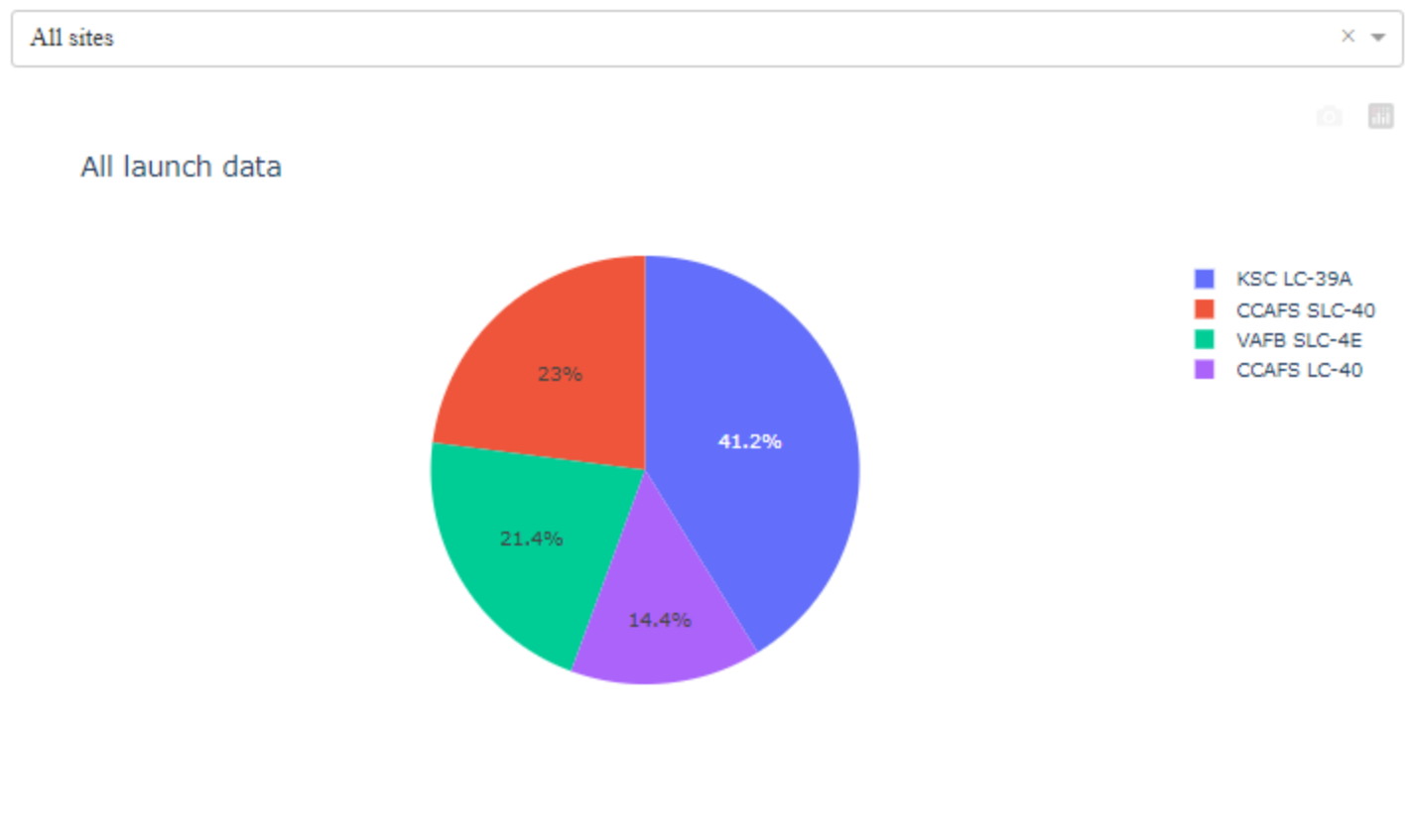




- Distance to the ocean 0.92km
- Distance to the railway 1.20km
- Distance to Cape Canaveral 18.75km
- Distance to the highway access 17.27km

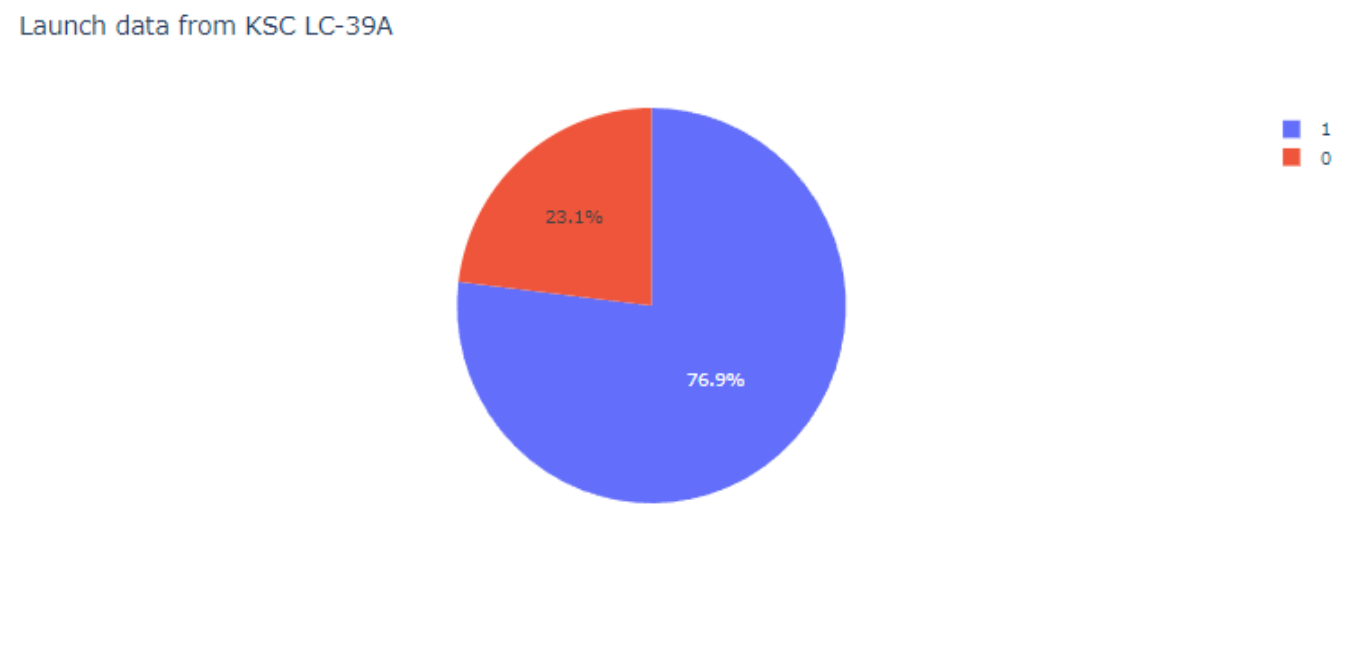

Section 4
Build a Dashboard with Plotly Dash

Dashboard results: Launch sites success rate



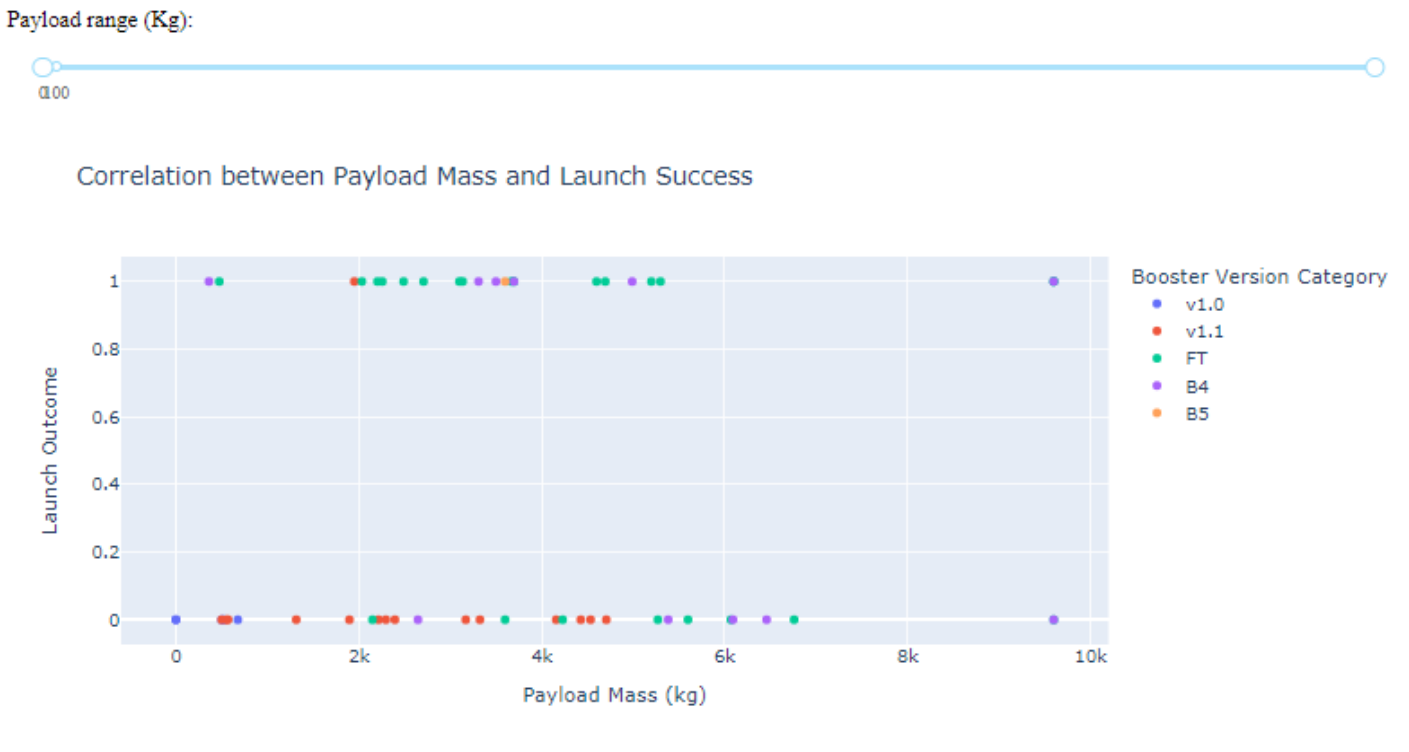
- The most successful is KSC LC-39A
- The least successful is CCAFS LC-40

Dashboard results: Highest success rate



- The second most used launch site has the highest success rate.

Dashboard results: Payloads and Outcomes

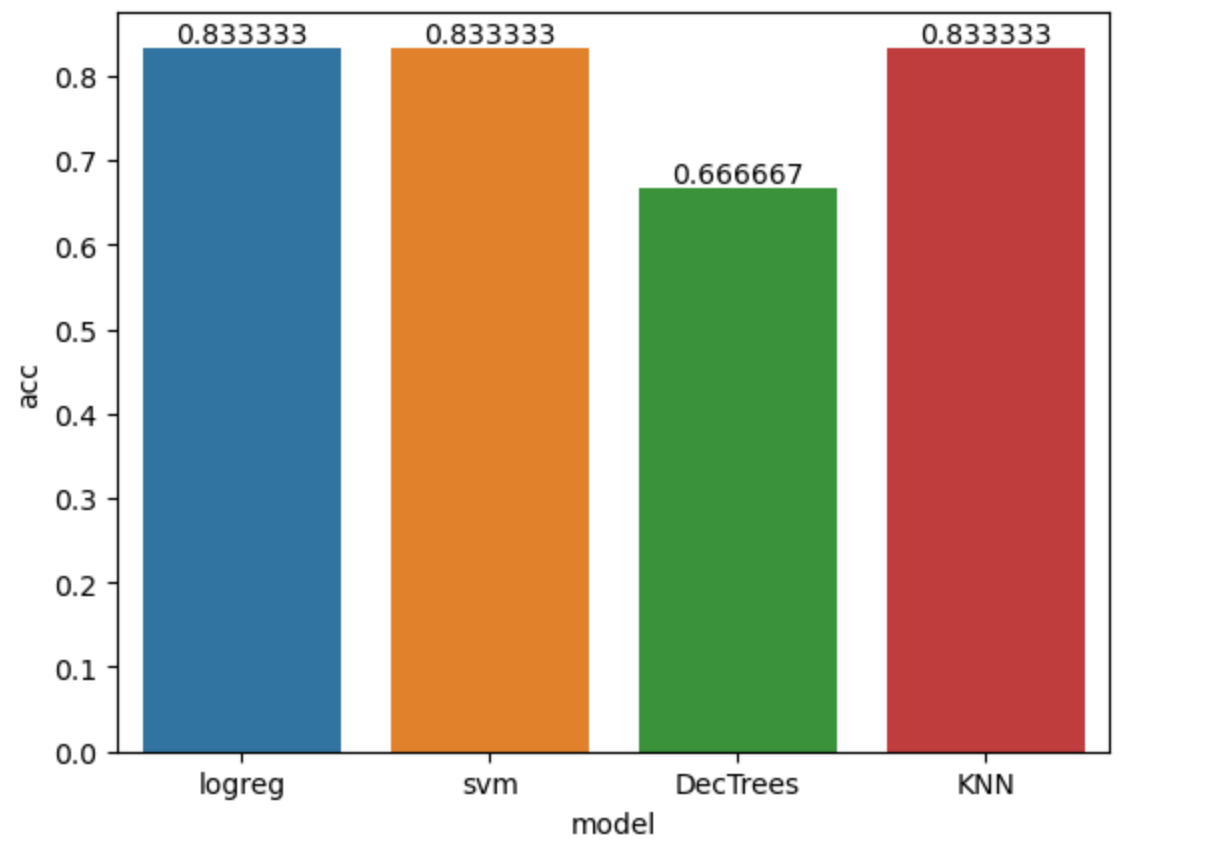


- The slider on the top allows to select the payload range that we need to observe.
- The booster version that deployed successfully the highest payload was v1.0

Section 5

Predictive Analysis (Classification)

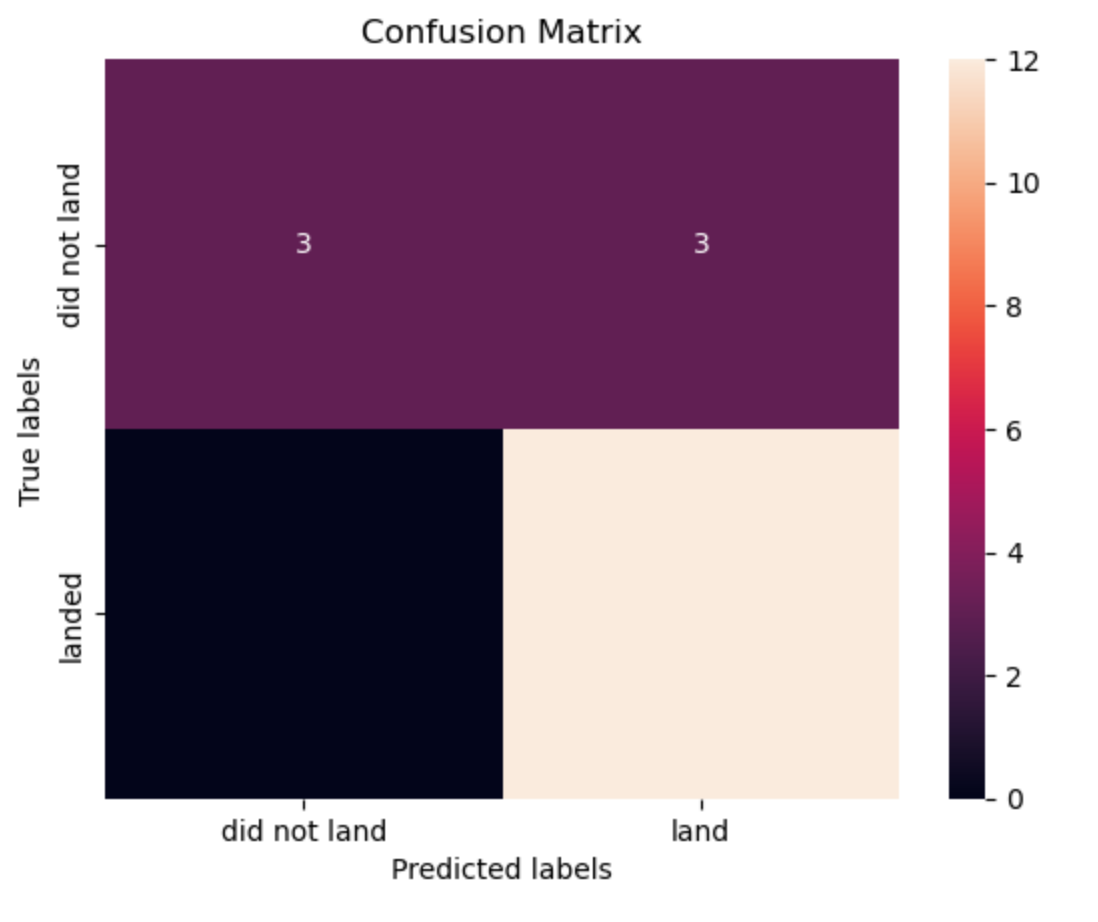
Classification Accuracy



- According to the Score calculated with the test set, Logistic Regression, SVM, and KNN achieved the highest scores, with na accuracy of 0.83.
- Decision Trees got the worst score, with an accuracy of 0.67.

Confusion Matrix

Logistic Regression



- Overall the Logistic Regression algorithm, had good results mostly regarding predicting Landing.
- Correctly predicted 12 Landings
- Correctly predicted 3 Landing failures
- Incorrectly predicted 0 failed landings
- Incorrectly predicted 3 landings.

Conclusions

- The Success Rate of Landings has increased with time almost linearly as a first approach.
- There have been more launches of payloads below 8000kg. The success rate of landings of the first stage for higher payloads is greater.
- The orbits that show 100% of success rate have few launches, being most cases one single launch, the ES-L1, GEO, HEO, and SSO. The higher landing success rate with more launches are to LEO and VLEO orbits.
- The most used launch site is CCAFS LC 40, although it is not the one with the highest success rate. The most successful is KSC LC 39A with less launches.
- The predictive models present an 83% accuracy for Logistic Regression, SVM, and KNN, with the test data set, while the Decision Trees algorithm had a worst performance with 67% of accuracy.

Thank you!