IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Roque Lois Taboada Iglesias
08/05/2023

Outline

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

Executive Summary

- Summary of methodologies

    -Data Collection trough API

    -Data Collection with Web Scraping

    -Data Wrangling

    -Exploratory Data Analysis with SQL

    -Exploratory Data Analysis with Data Visualization

    -Visual Analytics with Folium

    -Machine Learning Prediction

- Summary of all results

    -EDA Results

    -Interactive analytics

    -Predictive analysis

Introduction

- Project background and context

SpaceX advertises Falcon 9 rocket launches on its website, with acost 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.

- Problems you want to find answers

In this capstone, we will predict if the Falcon 9 first stage will land successfully.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - The data was collected using SpaceX REST API and web scrapping
- Perform data wrangling
  - The data was processed using one-hot encoding from 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
  - Build, tune and evaluate classification models using ML

Data Collection

- Data collection is the procedure of collecting, measuring, and analyzing accurate insights for research using standard validated techniques. The data was collected by API REST and Web Scrapping.

- For API REST

Space X Rest API call -> API returns JSON file -> Make a DF from JSON -> Clean and export Data

- For Web Scrapping

Get HTML response from Wikipedia -> Extract data using BeutifulSoap -> Make a DF -> Export Data

Data Collection – SpaceX API

- Get request for rocket launch data using API
- Normalize JSON file to a DF
- Transform Data
- Create a dictionary and conver to DF
- Filter the dataframe to only include Falcon 9 launches
- Dealing with Missing Values
- Export to file
- https://github.com/Roqchain/Applied-Data-Science-Capstone/blob/main/1jupyter-labs-spacex-data-collection-api.ipynb

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

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

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

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

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

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

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

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

```
# Create a data from launch_dict
df = pd.DataFrame.from_dict(launch_dict)
df
```

```
print(df['BoosterVersion'].value_counts())
data_falcon9 = df[df['BoosterVersion'] !='Falcon 1']
print(data_falcon9['BoosterVersion'].value_counts())
```

```
# Calculate the mean value of PayloadMass column
PayloadMass_mean = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, PayloadMass_mean)
data_falcon9.isnull().sum()
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

- Request the Falcon9 Launch Wiki page from its URL
- Create a BeautifulSoup object from the HTML response
- Find tables
- Extract all column/variable names from the HTML table header
- Create a Dictionary, fill up with the data
- Convert it in a DF and export it as CSV

- https://github.com/Roqchain/Applied-Data-Science-Capstone/blob/main/2jupyter-labs-webscraping.ipynb

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

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

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

```
# 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')
```

```
column_names = []
# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)
```

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# 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']=[]
```

```
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)
            #print(date)
```

```
df=pd.DataFrame(launch_dict)
```

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

Data Wrangling

https://github.com/Roqchain/Applied-Data-Science-Capstone/blob/main/3_Lab2_DataWrangling.ipynb

In the dataset we have:

-True Ocean, True RLTS, True ASDS meaning the booster land successfully

-False Ocean, False RLTS, False ASDS meaning the booster did not land successfully

We need to transform the above variables in categorical variables

1 Calculate the number of launches on each site

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

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

2 Calculate the number and occurrence of each orbit

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

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

3 Calculate the number and occurence of mission outcome per orbit type

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

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

4 Create a landing outcome label from Outcome column

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

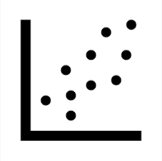
5 Export to file

```
df.to_csv("dataset_part_2.csv", index=False)
```

EDA with Data Visualization

https://github.com/Roqchain/Applied-Data-Science-Capstone/blob/main/5IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

- Scatter Graphs



- Flight Number VS Payload Mass

- Flight Number VS Launch Site

- Payload Mass VS Launch Site

- Orbit Type VS Flight Number

- Payload Mass VS Orbit Type

- Orbit Type VS Payload Mass

Used to show the relationship between two varibales

- Bar Graph

We use this Graph for Visualize the relationship between success rate of each orbit type

- Line Graph

We use this Graph for Visualize the launch success yearly trend

EDA with SQL

https://github.com/Roqchain/Applied-Data-Science-Capstone/blob/main/4jupyter-labs-eda-sql-coursera_sqllite.ipynb

- **SQL queries to understand the dataset**
- 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 successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

Build an Interactive Map with Folium

- Markers, circles, lines, etc. created and added to the folium map:

- Mark all launch sites on a map
- Mark the success/failed launches for each site on the map
- For each launch result in spacex_df data frame, add a folium.Marker to marker_cluster
- Calculate the distances between a launch site to its proximities
- Mark down a point on the closest coastline using MousePosition and calculate the distance between the coastline point and the launch site.
- Draw a PolyLine between a launch site to the selected coastline point
- Draw a line between a launch site to its closest city, railway, highway, etc.

https://github.com/Roqchain/Applied-Data-Science-Capstone/blob/main/6IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- We built a Plotly Dash application for users to perform interactive visual analytics on SpaceX launch data in real-time.

This dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart. I was guided to build this dashboard application via the following tasks:

TASK 1: Add a Launch Site Drop-down Input Component

TASK 2: Add a callback function to render success-pie-chart based on selected site dropdown

TASK 3: Add a Range Slider to Select Payload

TASK 4: Add a callback function to render the success-payload-scatter-chart scatter plot

https://github.com/Roqchain/Applied-Data-Science-Capstone/blob/main/7spacex_dash_app.py

Predictive Analysis (Classification)

https://github.com/Roqchain/Applied-Data-Science-Capstone/blob/main/8IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20(1).ipynb

Data preparation
- Load dataset
- Normalize data
- Split data into train and test

Model preparation
- ML algorithms selection
- Set MLA parameters
- Train the models

Model evaluation
- Get the best hyperparameters for each model
- Calculate the accuracy and other metrics fo the models
- Plot the confusión matrix

Model comparison
- Comparison of the models
- Select the best model

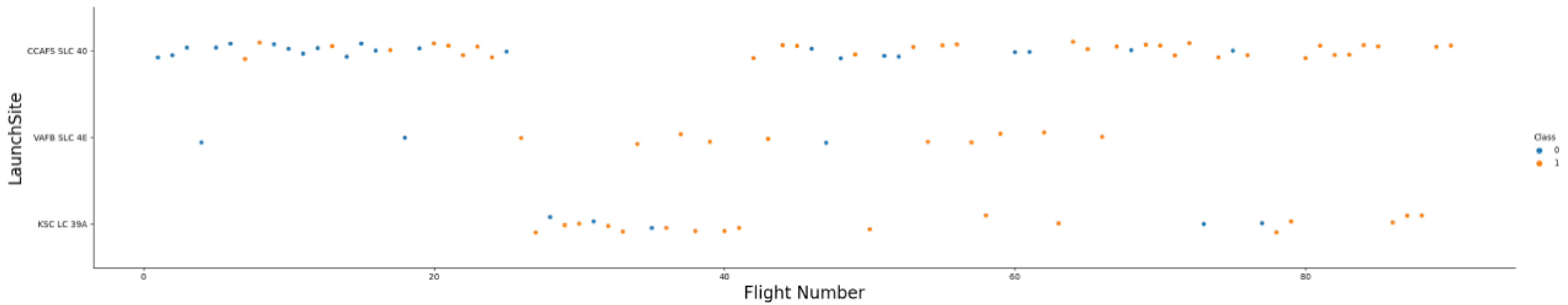
See the Notebook for more details if You want ☺

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

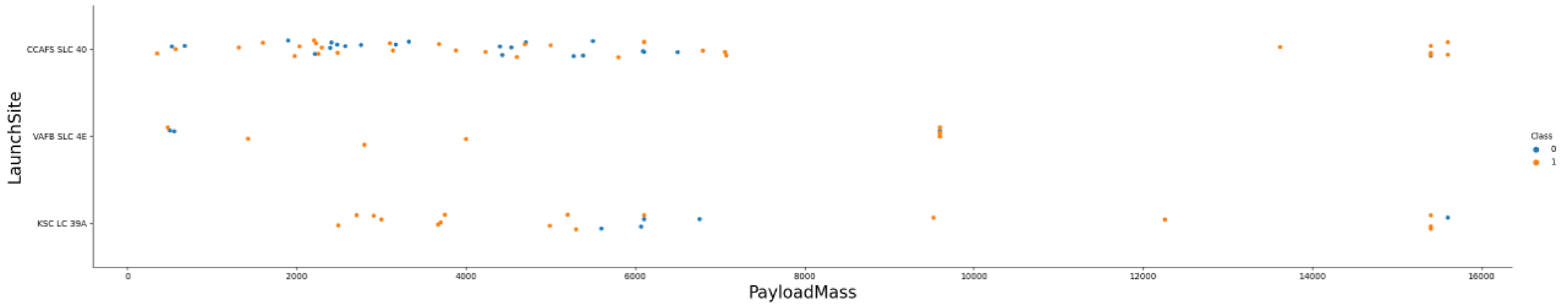
[9]:
```
### TASK 1: Visualize the relationship between Flight Number and Launch Site
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



Of course, it is clearly seen that by increasing the number of flights, the probability of success increases considerably.
The lower probability of success of SLC 40 is probably due to the fact that it was where most of the first attempts were launched, since the last 9 launches have been successful.

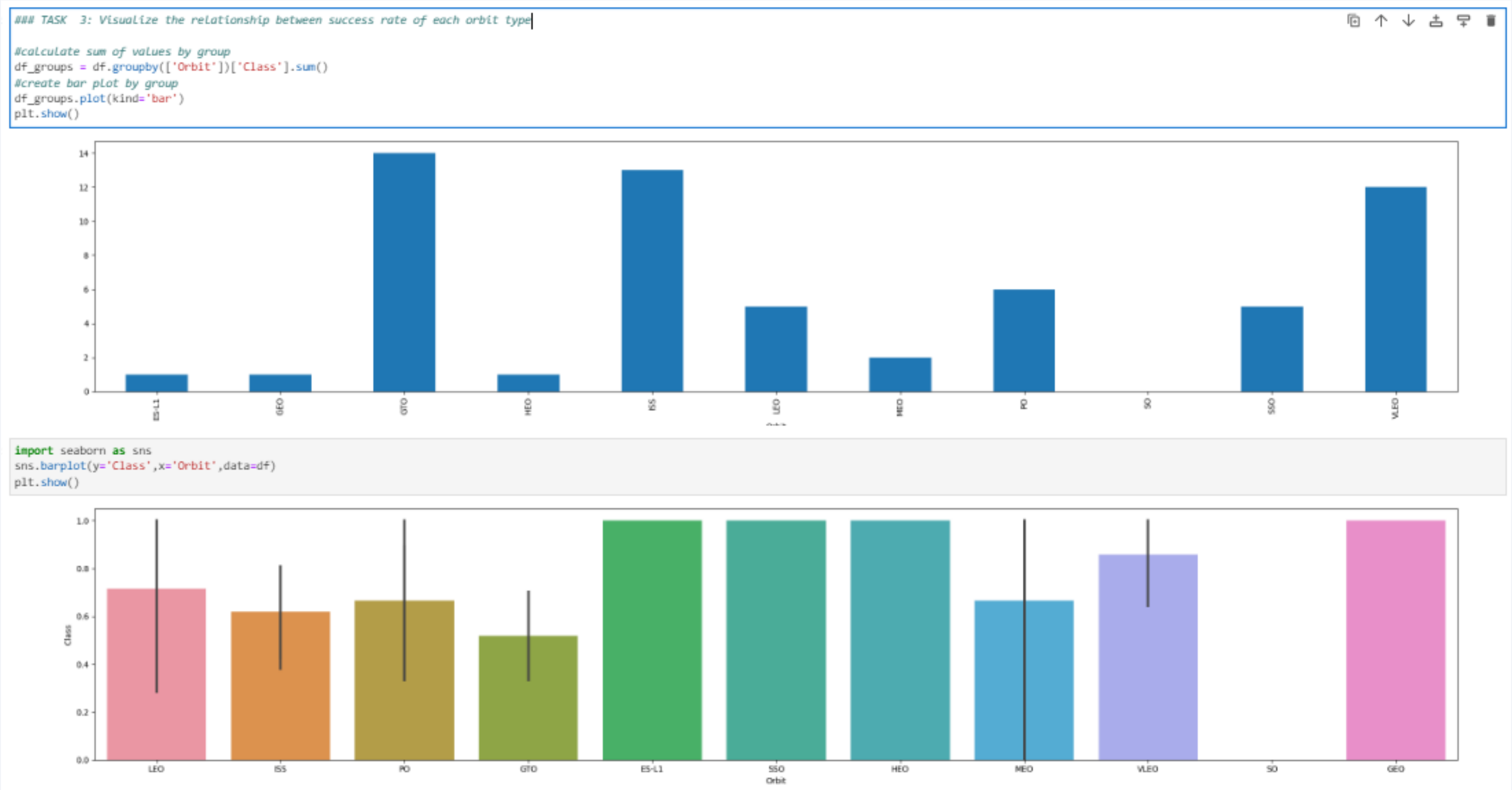
Payload vs. Launch Site

```
### TASK 2: Visualize the relationship between Payload and Launch Site
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



We also want to observe if there is any relationship between launch sites and their payload mass.
In CCAFS there are many attempts with loads less than 8k kilos
In VAFB It is the one that has the least attempts and there are no rockets launched for heavy payload mass(greater than 10000).
In KSIC no attempts less than 2k kilos
There is a zone between 8k and 14k where there are very few attempts
There is no clear relationship between these two variables.

Success Rate vs. Orbit Type



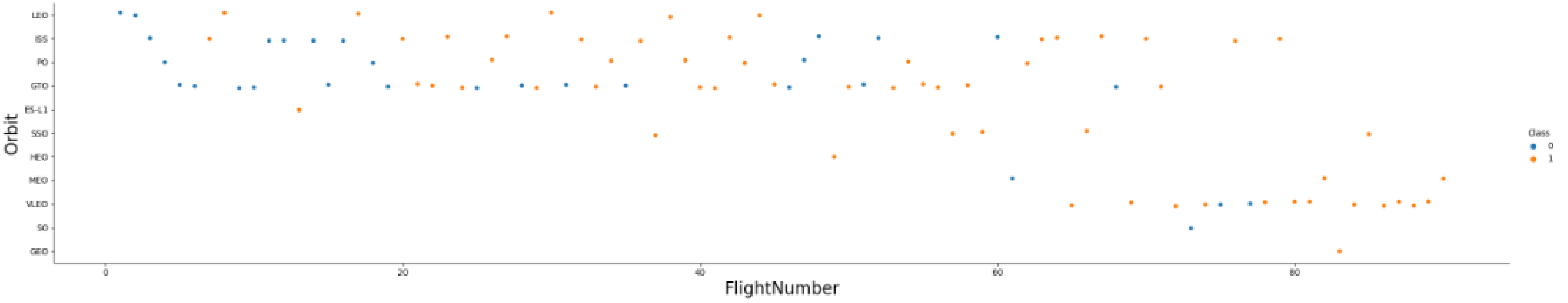
Total attempts up, % success down.

GTO, ISS and COMO are the ones that have the most attempts.
SO has no intent.

ES-L1, SSO, HEO and GEO orbits have a 100% success rate

Flight Number vs. Orbit Type

```
### TASK  4: Visualize the relationship between FlightNumber and Orbit type
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

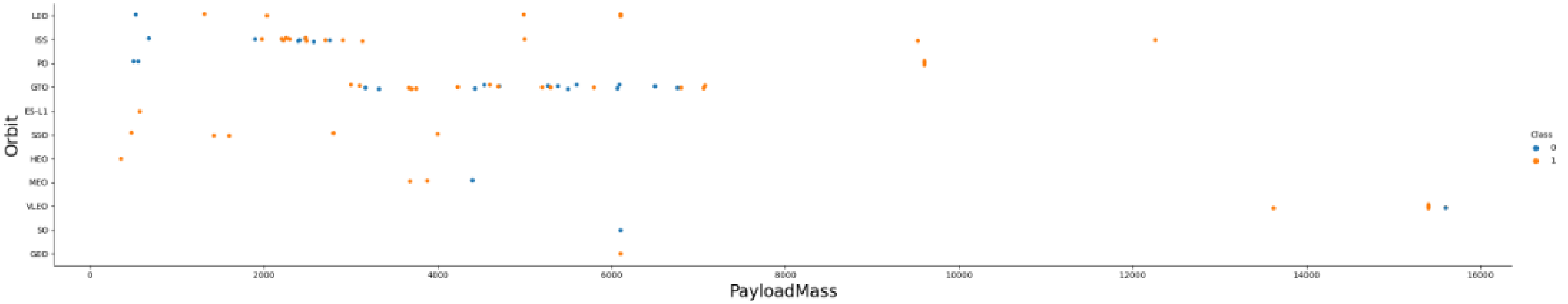


In the LEO orbit the Success appears related to the number of flights

On the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type

```
### TASK  5: Visualize the relationship between Payload and Orbit type
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



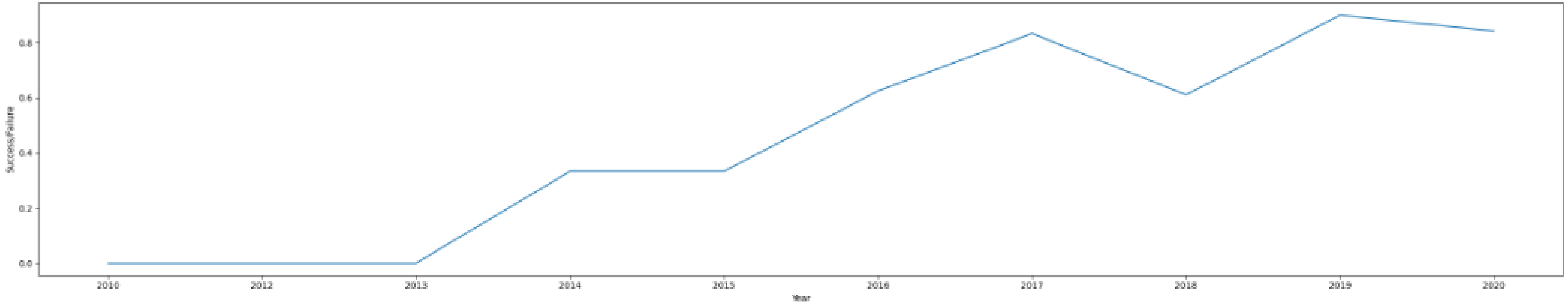
With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

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

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

```
plt.plot(average_by_year["Year"],average_by_year["Class"])
plt.xlabel("Year")
plt.ylabel("Success/Failure")
plt.show()
```



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

All Launch Site Names

```
%sql select Unique(LAUNCH_SITE) from SPACEX;
```

```
 * ibm_db_sa://jnv63939:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLU
DB
   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'

```
%sql select * from SPACEX where LAUNCH_SITE like 'CCA%' limit 5;
```

```
 * ibm_db_sa://jnv63939:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
   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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

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

```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEX where customer = 'NASA (CRS)';
```

```
 * ibm_db_sa://jnv63939:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
   sqlite:///my_data1.db
Done.
```

payloadmass
45596

Average Payload Mass by F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) as F9v11avg from SPACEX where booster_version like 'F9 v1.1';
```

```
 * ibm_db_sa://jnv63939:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
   sqlite:///my_data1.db
Done.
```

f9v11avg
2928

First Successful Ground Landing Date

```
%sql select min(DATE) from SPACEX where landing__outcome = 'Success (ground pad)'
```

```
 * ibm_db_sa://jnv63939:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
   sqlite:///my_data1.db
Done.
```

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select booster_version,payload_mass__kg_,landing__outcome from SPACEX where landing__outcome = 'Success (drone ship)' and payload_mass__kg_ > 4000 and payload_mass__kg_ < 6000
```

```
 * ibm_db_sa://jnv63939:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
   sqlite:///my_data1.db
Done.
```

booster_version	payload_mass__kg_	landing__outcome
F9 FT B1022	4696	Success (drone ship)
F9 FT B1026	4600	Success (drone ship)
F9 FT B1021.2	5300	Success (drone ship)
F9 FT B1031.2	5200	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

```
%sql select mission_outcome,count(MISSION_OUTCOME) as missionoutcomes from SPACEX GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://jnv63939:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
   sqlite:///my_data1.db
Done.
```

mission_outcome	missionoutcomes
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

```
%sql select booster_version, payload_mass__kg_  from SPACEX where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEX)
```

```
 * ibm_db_sa://jnv63939:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
   sqlite:///my_data1.db
Done.
```

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

```
%sql select DATE, booster_version, launch_site, landing__outcome  from SPACEX where landing__outcome = 'Failure (drone ship)' and Date like '2015%'
```

```
 * ibm_db_sa://jnv63939:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
   sqlite:///my_data1.db
Done.
```

DATE	booster_version	launch_site	landing__outcome
2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
%sql SELECT * FROM SPACEX WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' and landing__outcome like 'Success%' ORDER BY DATE DESC;
```

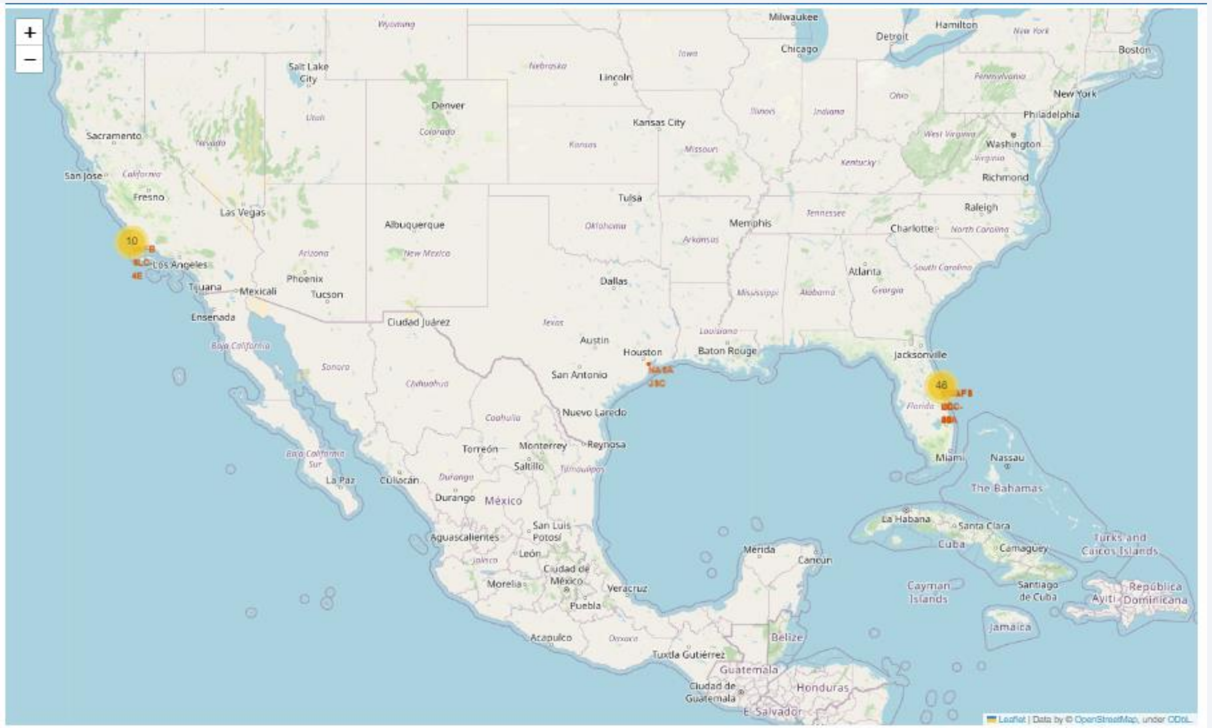
```
 * ibm_db_sa://jnv63939:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
   sqlite:///my_data1.db
Done.
```

landing__outcome
Success (ground pad)
Success (drone ship)
Success (drone ship)
Success (ground pad)
Success (drone ship)
Success (drone ship)
Success (drone ship)
Success (ground pad)

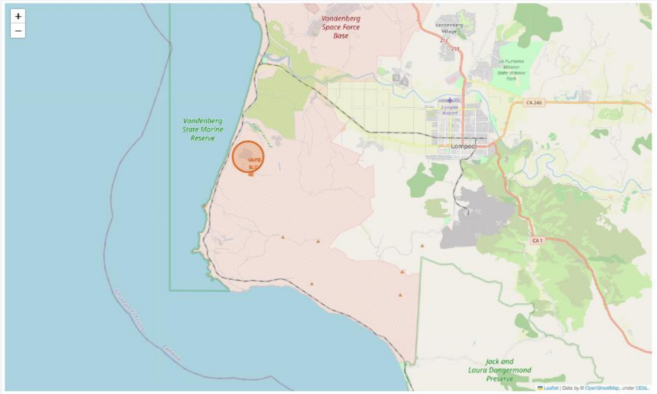
Section 3

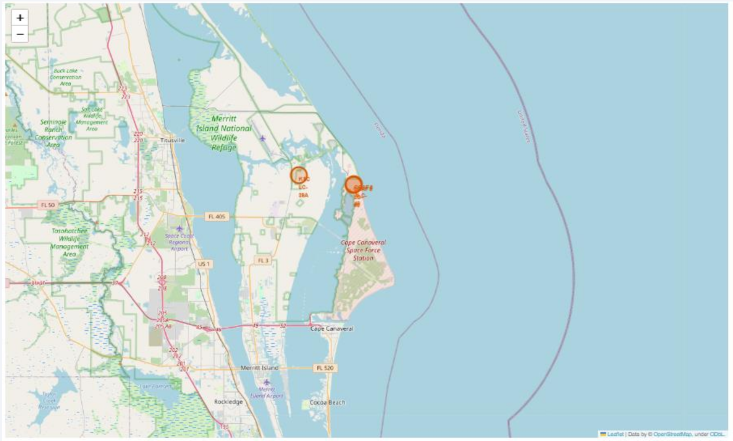
Launch Sites Proximities Analysis

General Folium Map

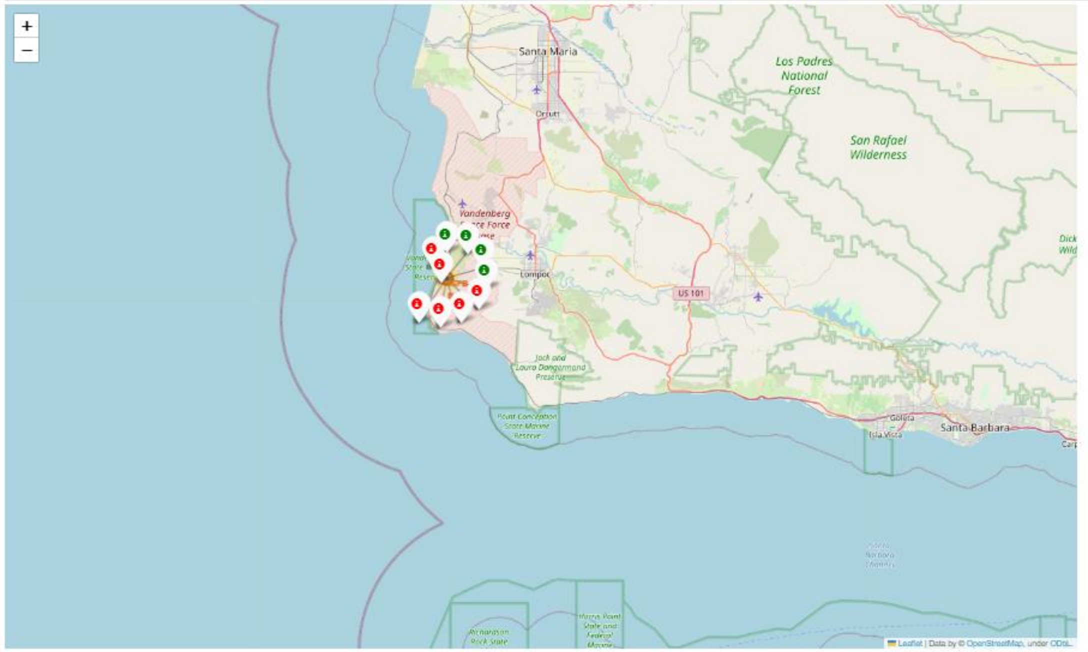


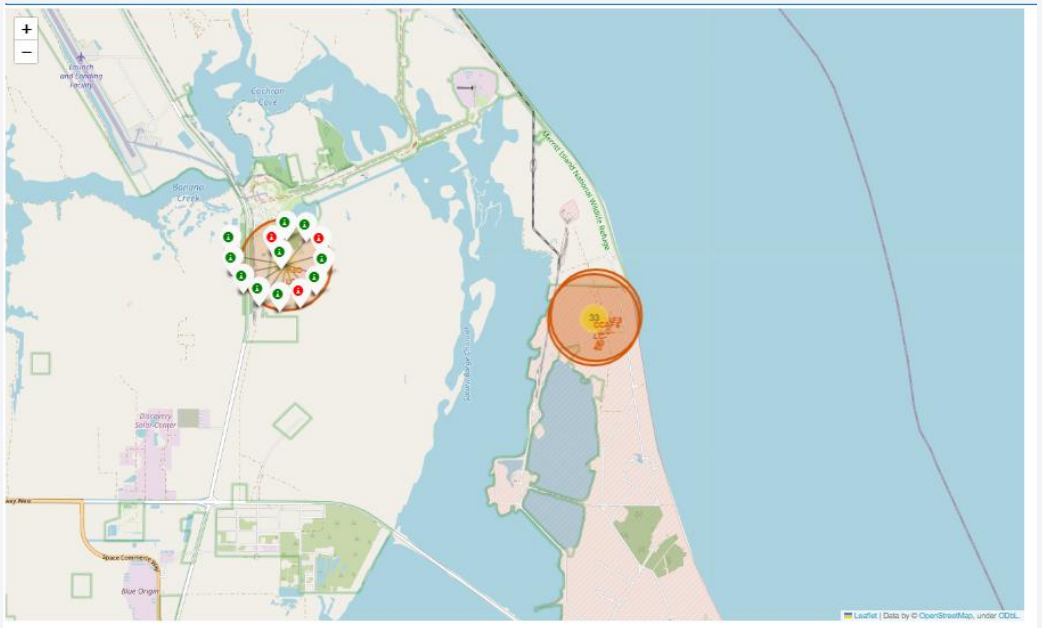
We can see the 3 sites here



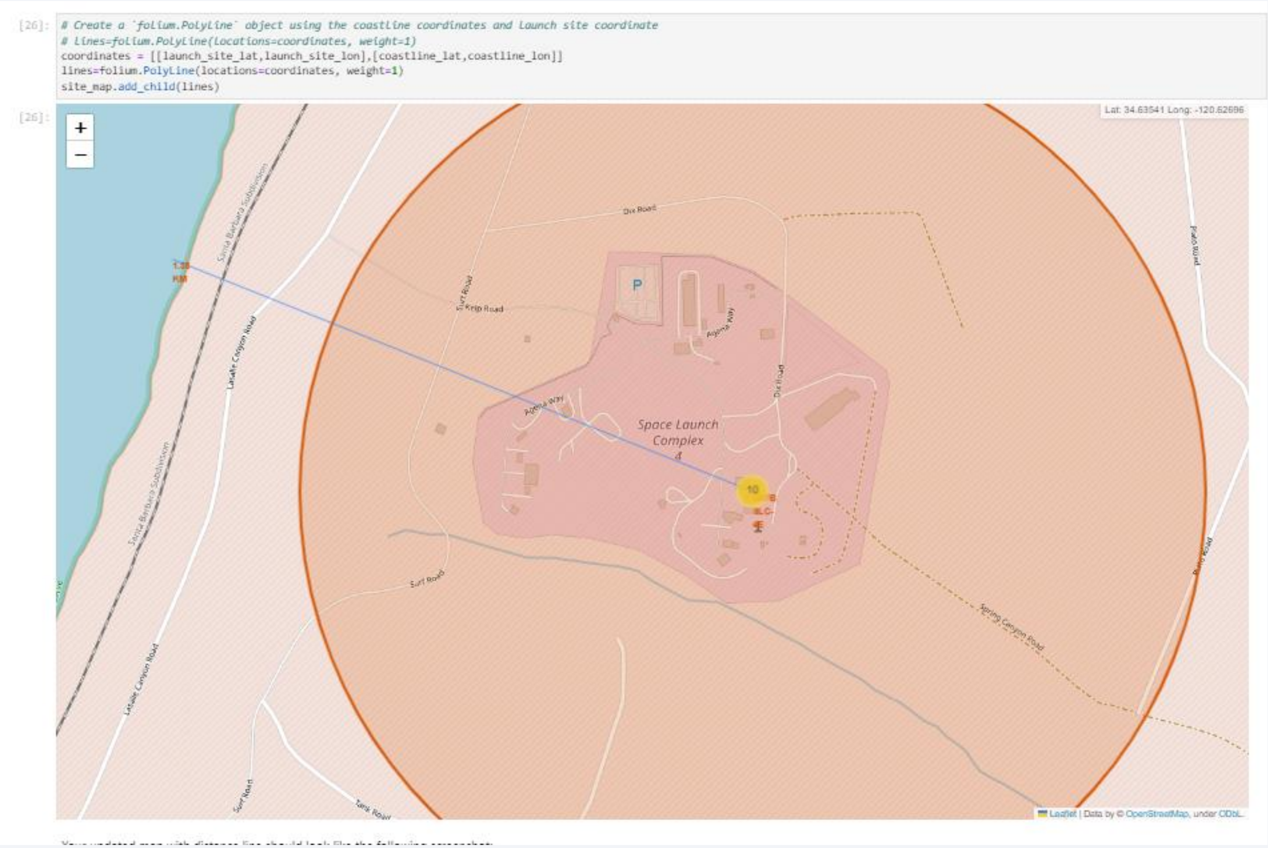


color-labeled launch outcomes





Distance

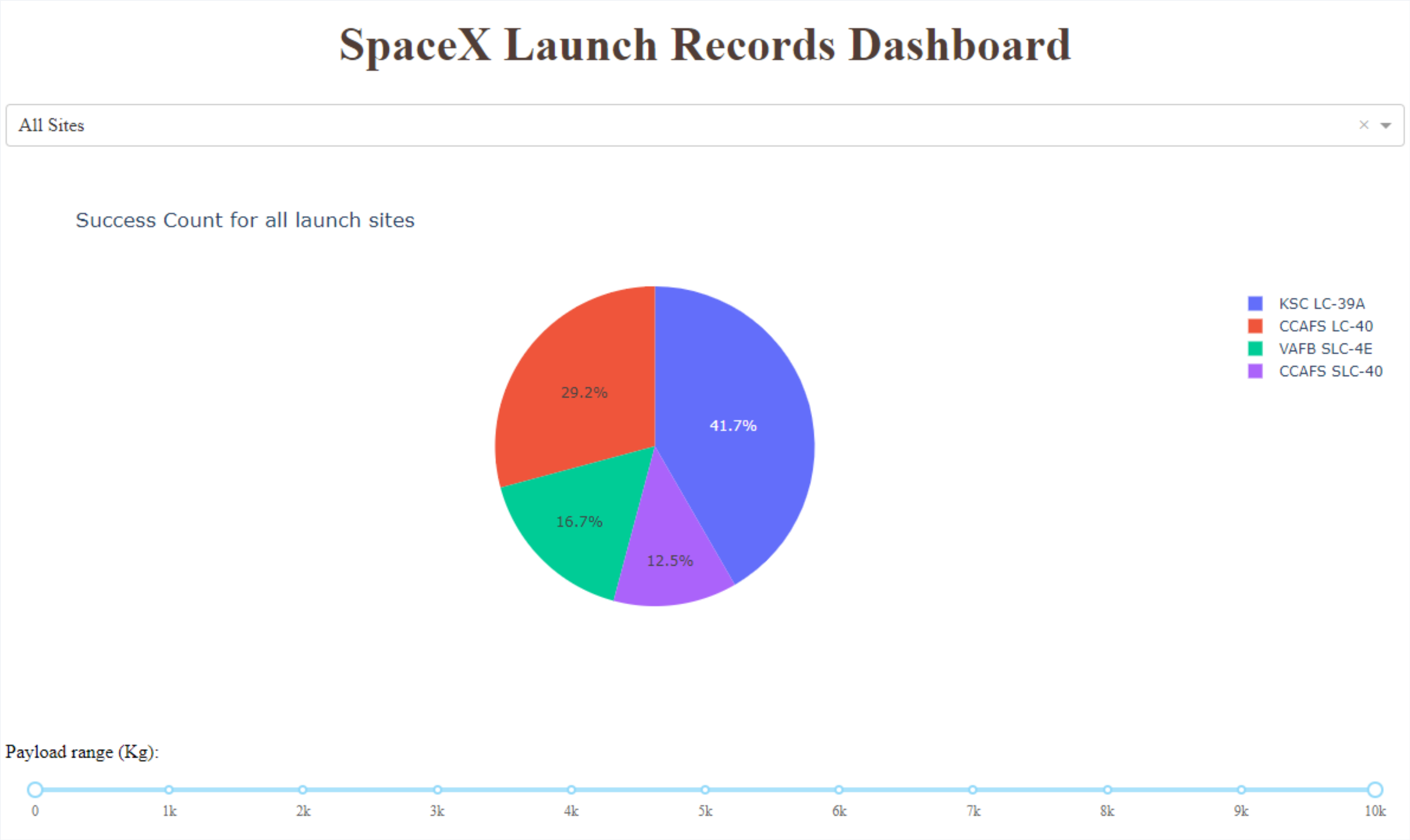


Distance to the shore

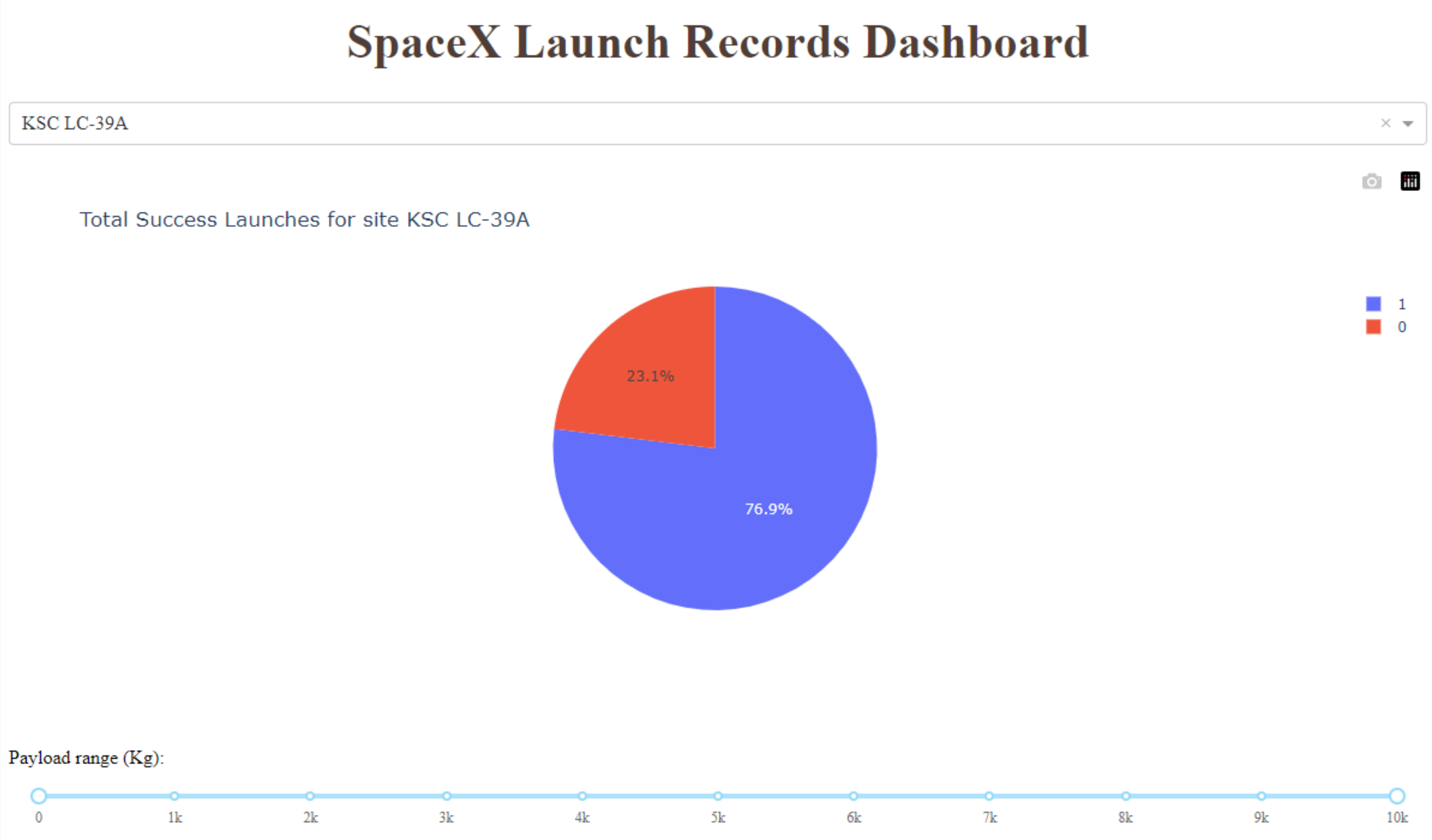
Section 4

Build a Dashboard with Plotly Dash

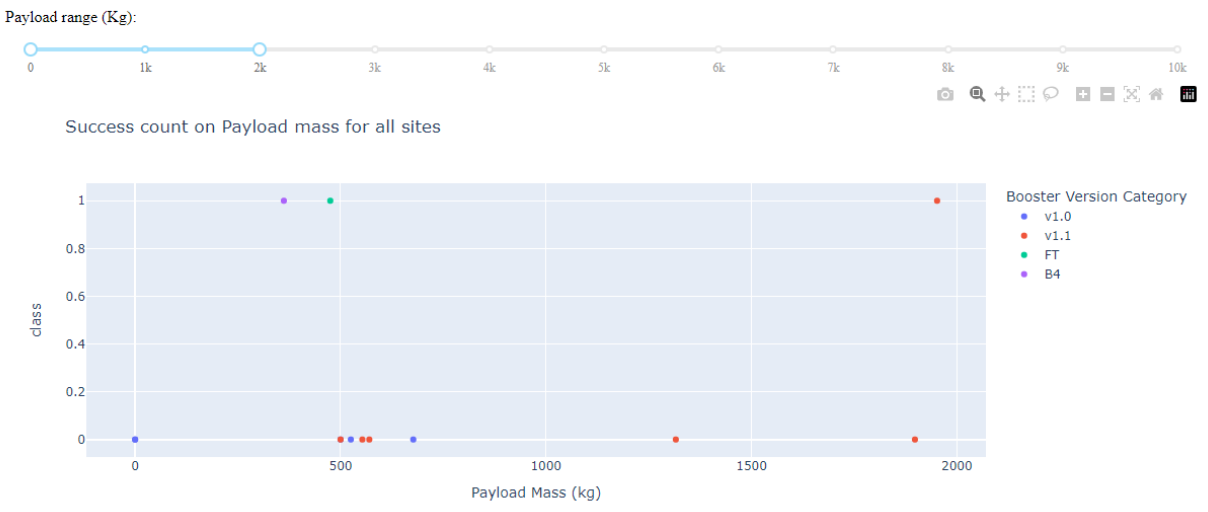
Success count for all sites piechart



Launch site with highest launch success ratio

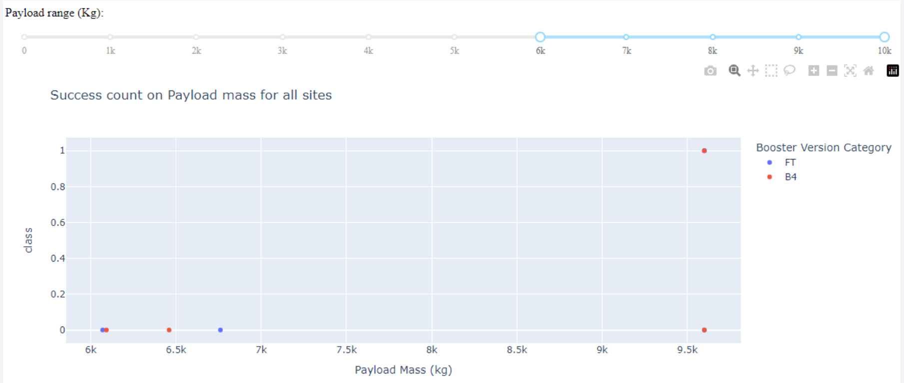


Payload vs. Launch Outcome scatter plot



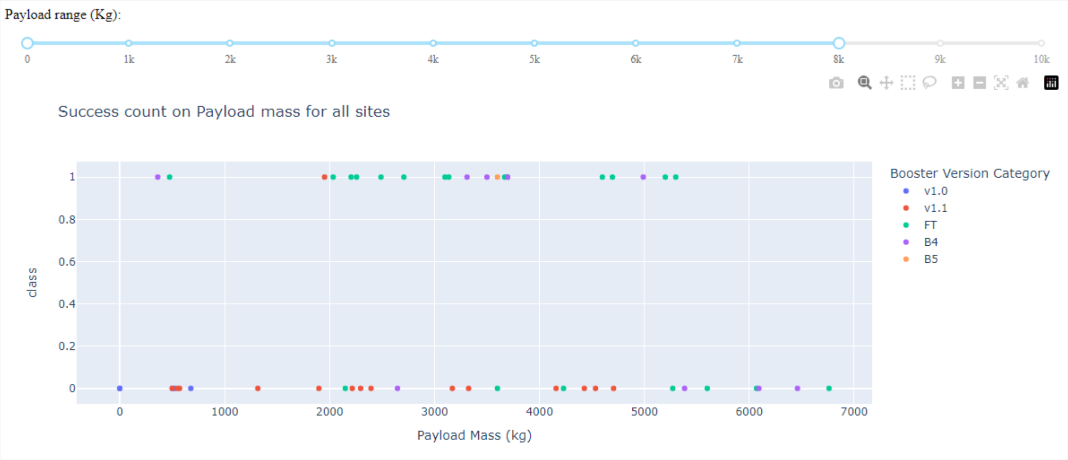
There are no attemps for payload of 2k with B4 Booster Version

This is the best for high masses



The Booster Version with Hight sucess rate is FT

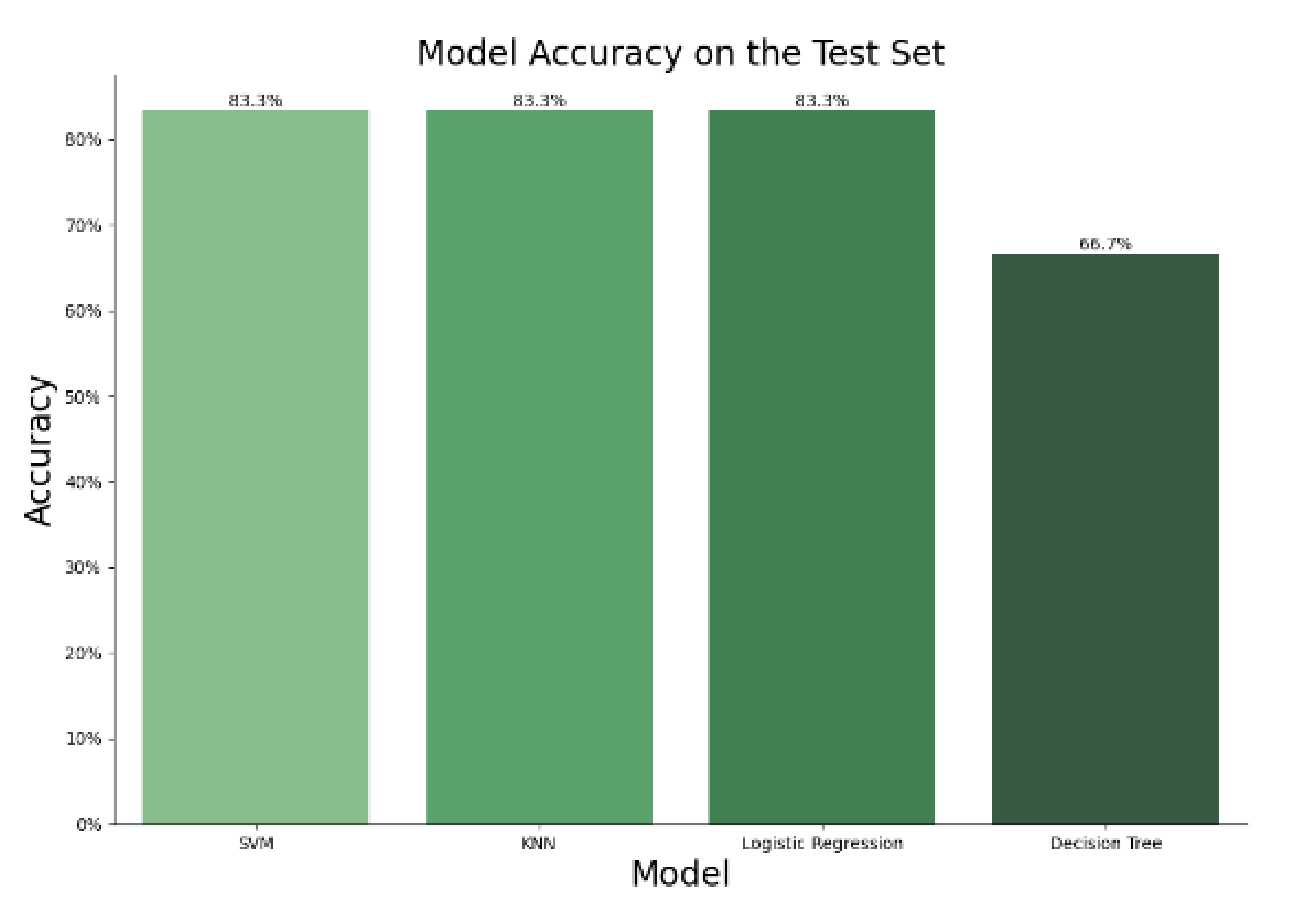
The worse is v1.1

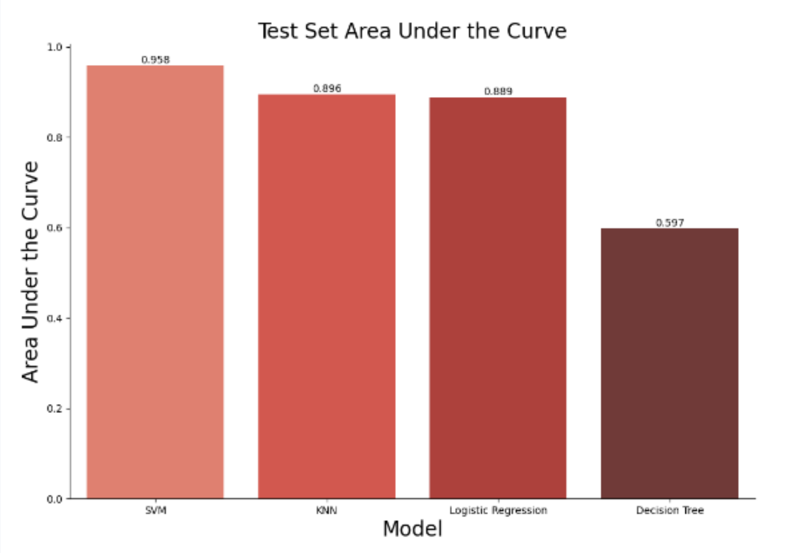


Section 5

Predictive Analysis (Classification)

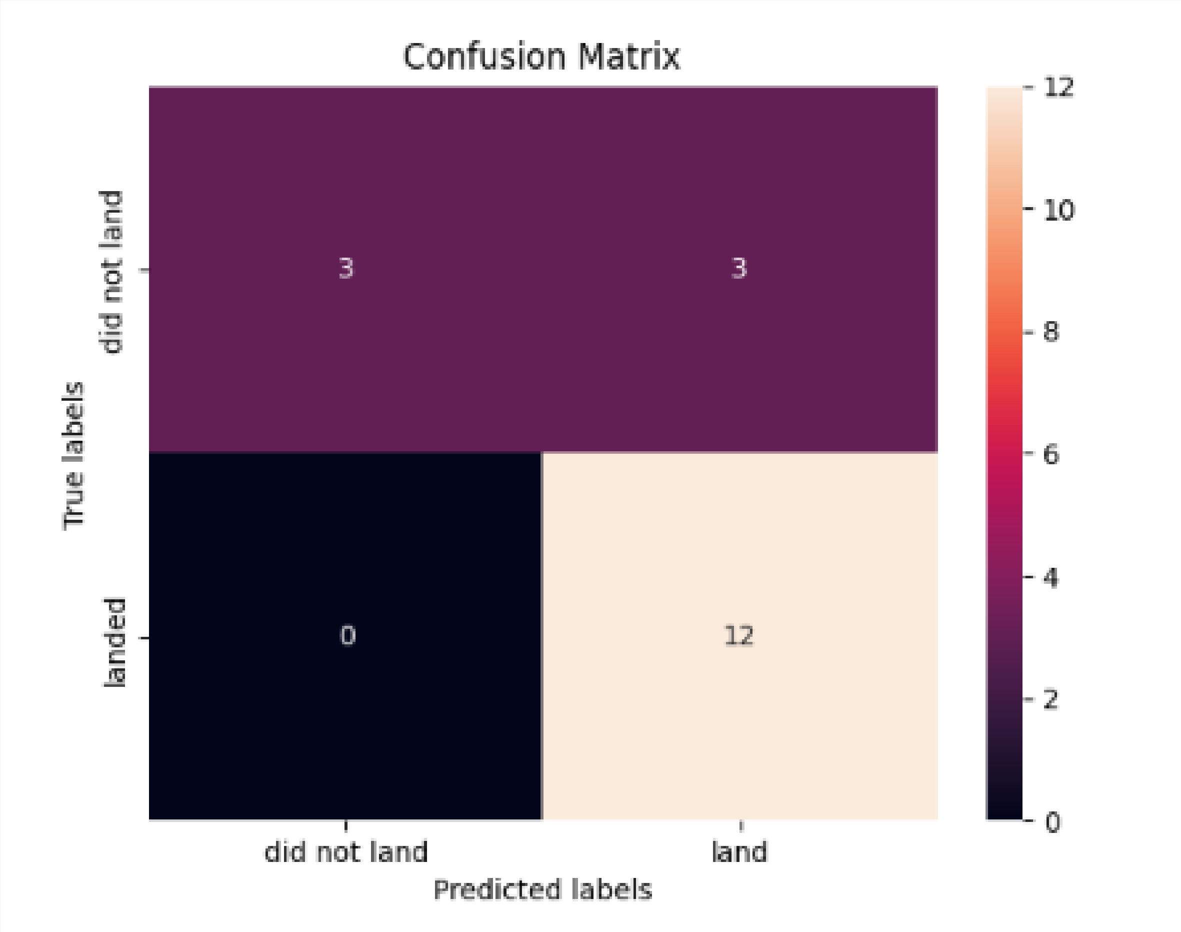
Classification Accuracy





SVM, KNN and Logistic Regression have the same ACC.
However, SVM has the best AUC-ROC , this is why i choose this model as the better option

Confusion Matrix



Confusion Matrix is the visual representation of the Actual VS Predicted values. It measures the performance of our Machine Learning classification model and looks like a table-like structure.

TP (3): True Positive: The values which were actually positive and were predicted positive.
FP (3): False Positive: The values which were actually negative but falsely predicted as positive. Also known as Type I Error.
FN (0): False Negative: The values which were actually positive but falsely predicted as negative. Also known as Type II Error.
TN (12): True Negative: The values which were actually negative and were predicted negative.

Conclusions

- The success or not of a launch can be explained by several factors, but propabily the number of previous launches is the mos important variable. We can see this by looking the Launch Success Yearly Trend from the D22. The evidence of this is that, from 2013 to 2020, the success rate has increased directly proportional to the passing of the years. This is surely due to the accumulation of knowledge between sets
- ES-L1, SSO, HEO and GEO orbits have a 100% success rate being the orbits in which there are better results
- The site with more launch success rate is KSCLC-39 with a success rate of 76,92% (10/13)
- I choose the SVM model because even though SVM, KNN and Logistic Regression have the same ACC, The first has more AUC-ROC than the others

Thank you!