IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Muhammad Hibatur Akmal
11/13/2022

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.

Section 1

Methodology

Methodology

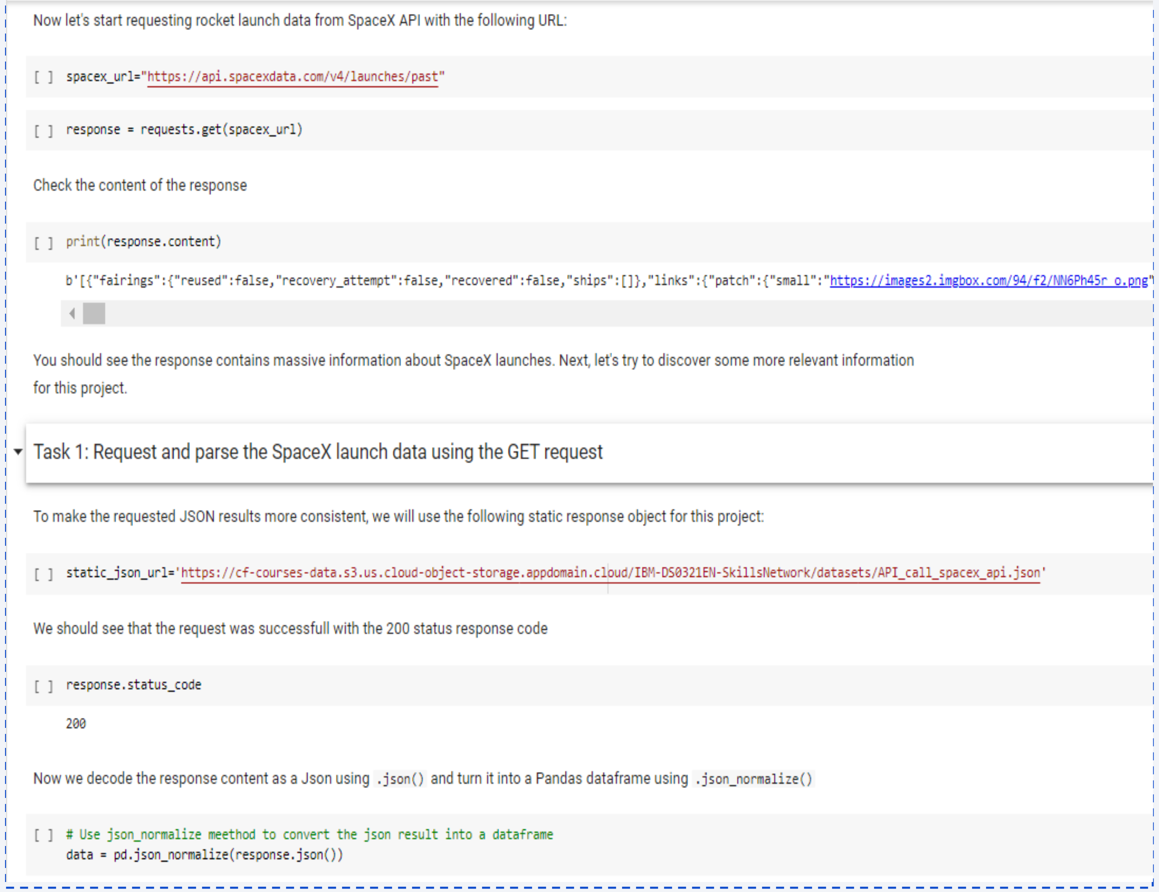
- Executive Summary
- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual 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, decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - Next, cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, 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

- 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 https://github.com/MHAkmal621/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb


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

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

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

Check the content of the response

```
[ ] print(response.content)
```

b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/NN6Ph45r_o.png"

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
[ ] static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

```
[ ] response.status_code
```

200

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

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

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 https://github.com/MHAkmal621/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

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

```
200
```

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

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

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

```
# Use soup.title attribute
soup.title
```

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

TASK 3: Create a data frame by parsing the launch HTML tables

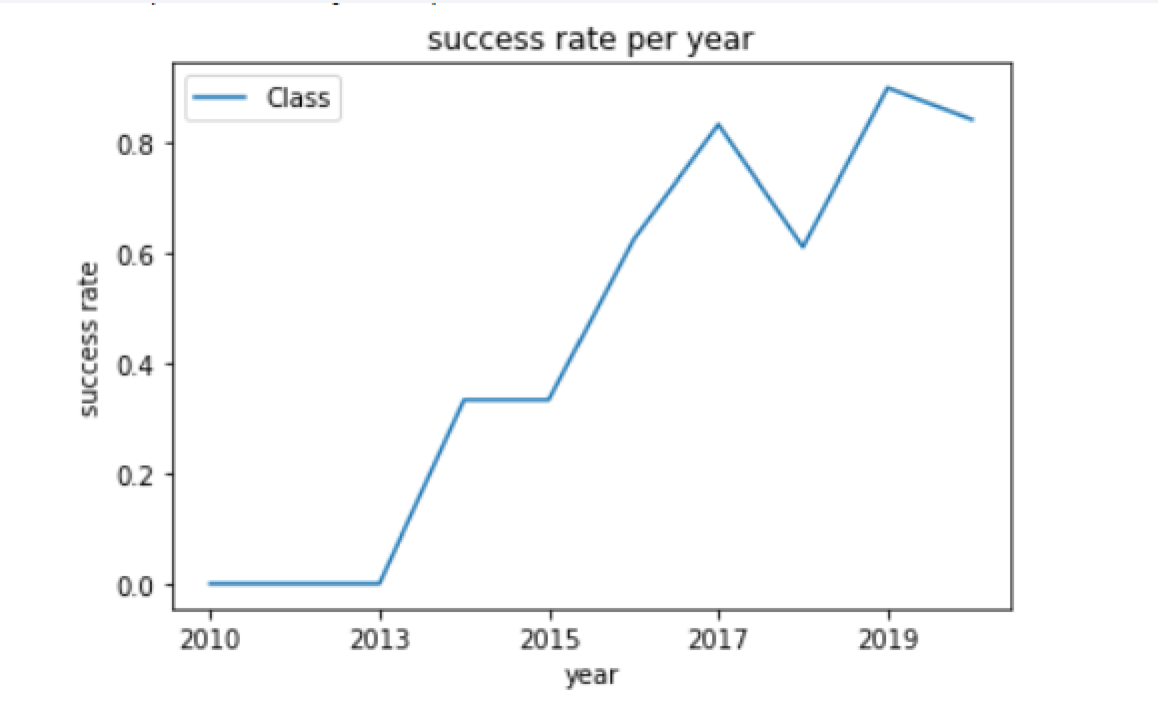
We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

```
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']=[]
```

Data Wrangling

- Performed exploratory data analysis and determined the training labels.
- Calculated the number of launches at each site, and the number and occurrence of each orbits
- Created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/MHAkmal621/IBM-Applied-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization



- Explored the data by visualizing the relationship between success rate and year to get the average launch success yearly trend.

- The link to the notebook is https://github.com/MHAkmal621/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- Loaded the SpaceX dataset into a IBM Db2 on cloud database without leaving the jupyter notebook.
- 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 https://github.com/MHAkmal621/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera.ipynb

Build an Interactive Map with Folium

- 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.
- 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.
- 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.
- The link to the notebook is https://github.com/MHAkmal621/IBM-Applied-Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Built an interactive dashboard with Plotly dash
- Plotted pie charts showing the total launches by a certain sites
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is https://github.com/MHAkmal621/IBM-Applied-Data-Science-Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- Built different machine learning models and tune different hyperparameters using GridSearchCV.
- Used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- Found the best performing classification model.
- The link to the notebook is https://github.com/MHAkmal621/IBM-Applied-Data-Science-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

EXPLORATORY DATA ANALYSIS RESULTS

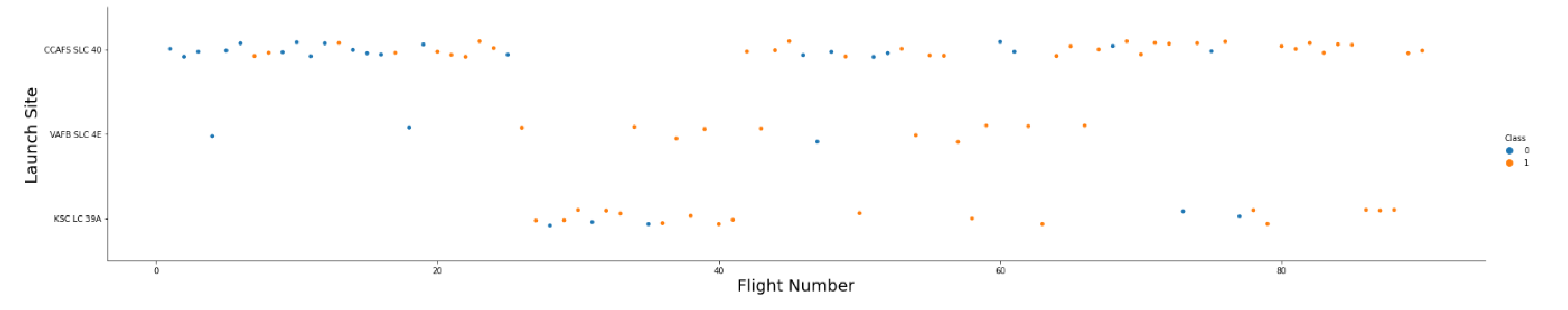
INTERACTIVE ANALYTICS DEMO IN SCREENSHOTS

PREDICTIVE ANALYSIS RESULTS

Section 2

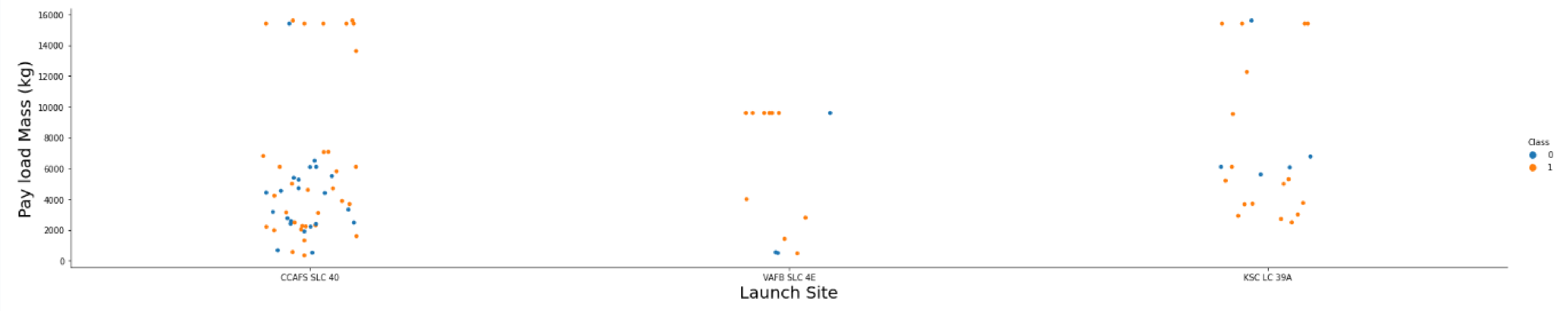
Insights drawn from EDA

Flight Number vs. Launch Site



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

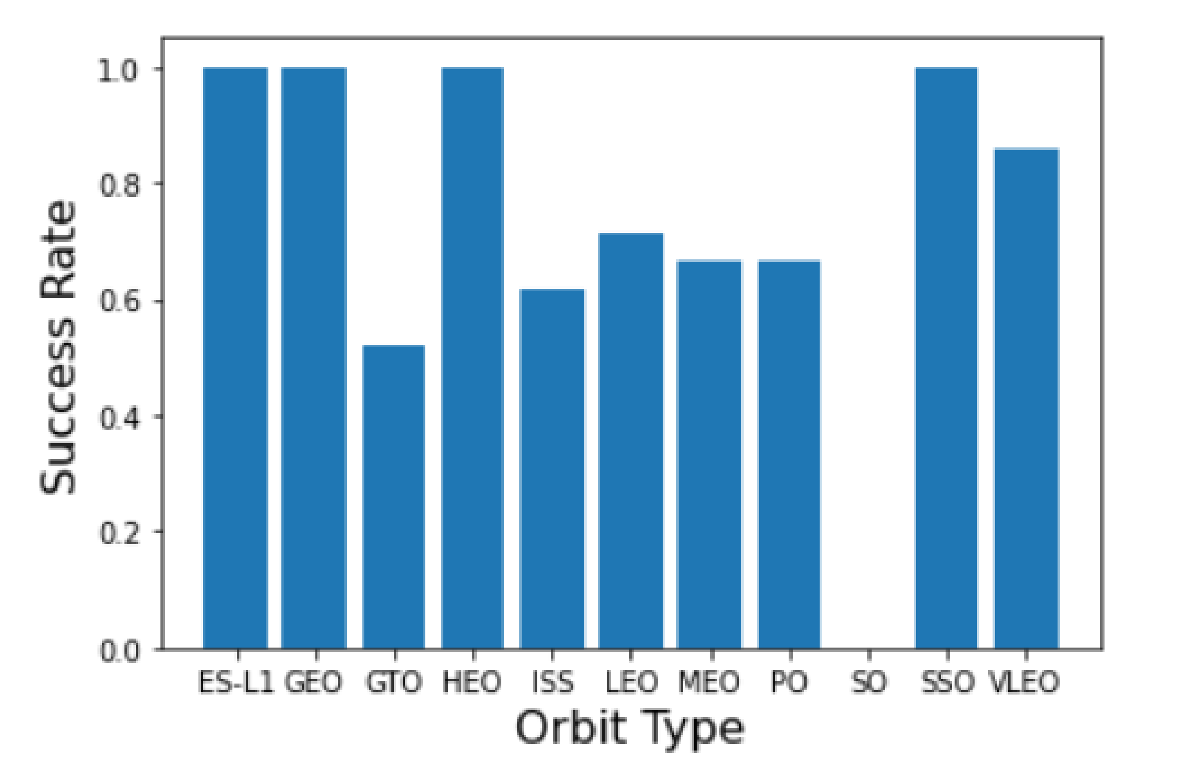
Payload vs. Launch Site



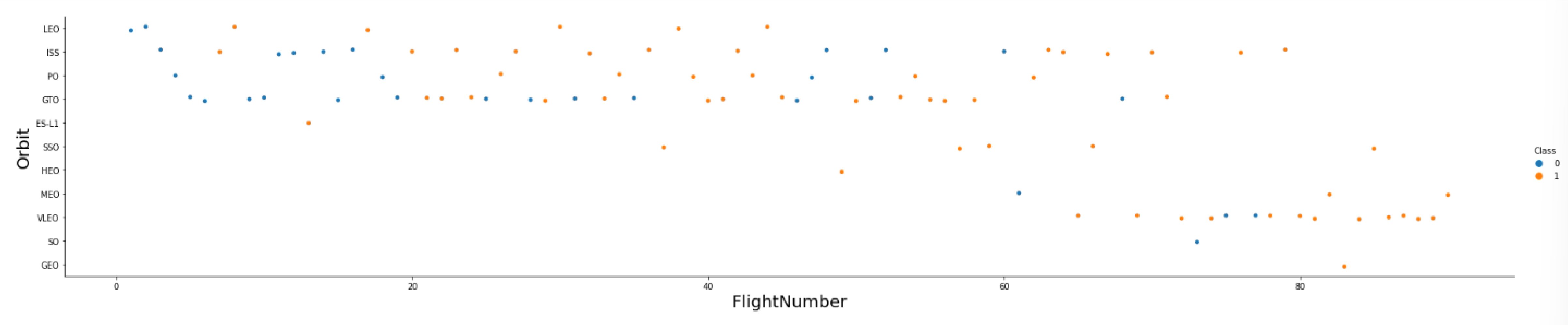
- From the plot, we know that a larger payload mass equal to 10000 makes the success rate higher

Success Rate vs. Orbit Type

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

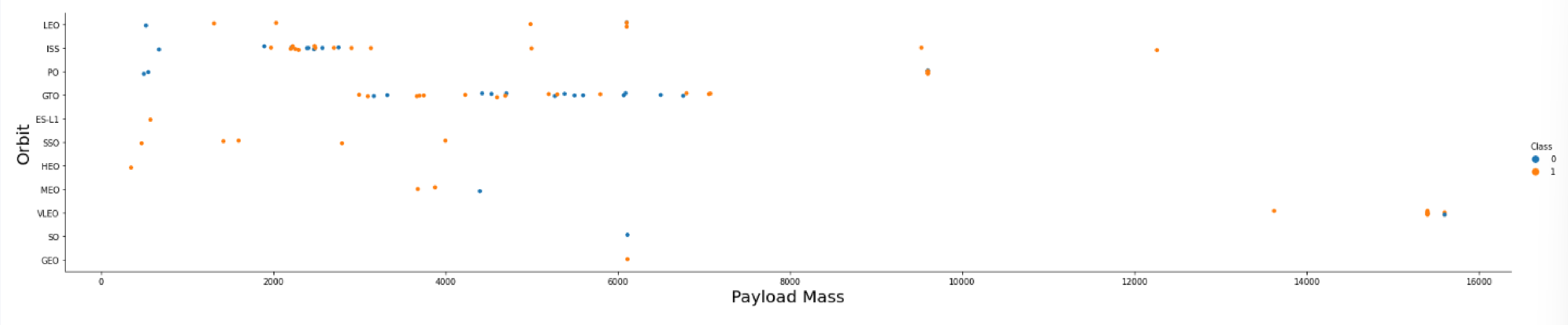


Flight Number vs. Orbit Type



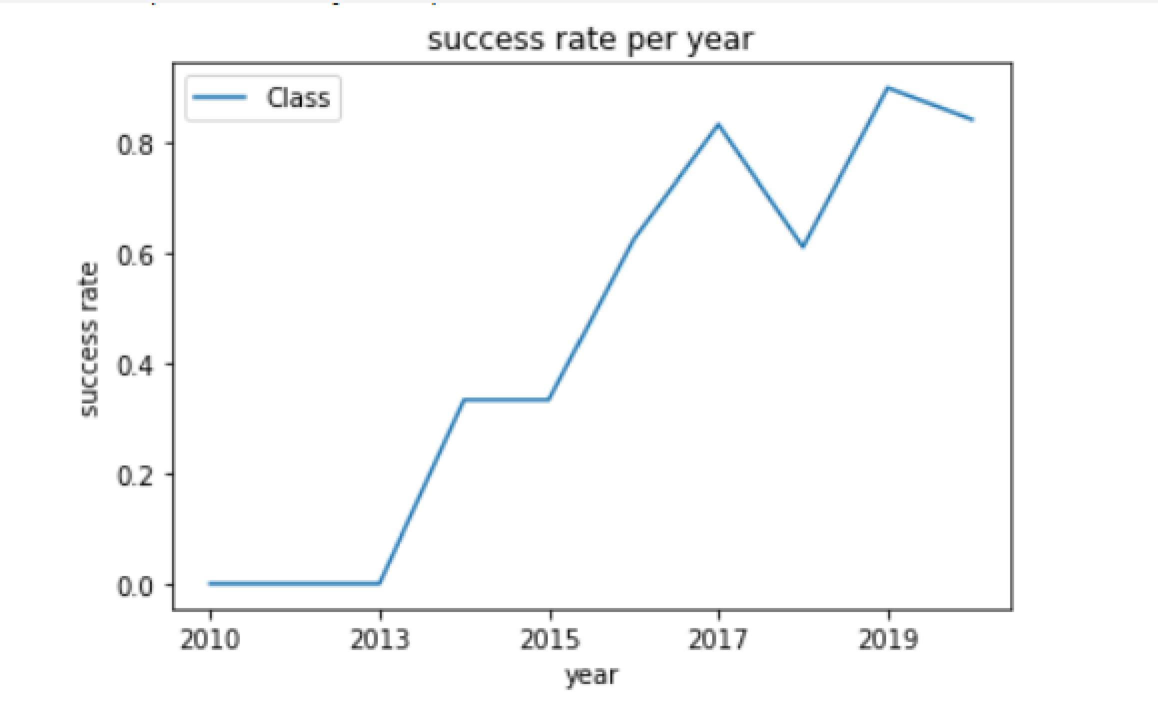
- The plot above shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.

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

```
[ ] %%sql
    select distinct launch_site
    from spacexdataset
```

```
 * ibm_db_sa://nrx97436:***@ea286ace-86c7-4d5b-8580-
Done.
```

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

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

Launch Site Names Begin with 'CCA'

- Used the query below to display 5 records where launch sites begin with `CCA`

```
[ ] %%sql
    select *
    from spacexdataset
    where launch_site like 'CCA%' Limit 5
```

```
 * ibm_db_sa://nrx97436:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
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

- calculated the total payload carried by boosters from NASA as 45596 using the query below

```
[ ]  %%sql
     select sum(payload_mass__kg_)
     from spacexdataset
     where customer = 'NASA (CRS)'

      * ibm_db_sa://nrx97436:***@ea286ace-86c7
     Done.
       1
     45596
```

Average Payload Mass by F9 v1.1

```
%%sql
select avg(payload_mass__kg_)
from spacexdataset
where booster_version = 'F9 v1.1'
```

```
 * ibm_db_sa://nrx97436:***@ea286ace-
Done.
  1
2928
```

- calculated the average payload mass carried by booster version F9 v1.1 as 2928

First Successful Ground Landing Date

- Observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

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

```
 * ibm_db_sa://nrx97436:***@ea286ace-86c7-4d5b-8580
Done.
```

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- 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

```
[ ] %%sql
    select booster_version
    from spacexdataset
    where landing__outcome = 'Success (drone ship)' and payload_mass__kg_ between 4000 and 6000
```

```
 * ibm_db_sa://nrx97436:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

```
[ ]  %%sql
     select mission_outcome, count(mission_outcome)
     from spacexdataset
     group by mission_outcome
```

```
 * ibm_db_sa://nrx97436:***@ea286ace-86c7-4d5b-8580-:
Done.
```

mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

- Calculated how many failure, success, and unclear status of mission outcome

Boosters Carried Maximum Payload

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

```
[ ]  %%sql
     select booster_version, payload_mass__kg_
     from spacexdataset
     where  payload_mass__kg_ = (select max(payload_mass__kg_) from spacexdataset)
```

```
 * ibm_db_sa://nrx97436:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.
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

- 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

```
[ ]  %%sql
     select "DATE", landing__outcome, booster_version, launch_site
     from spacexdataset
     where landing__outcome = 'Failure (drone ship)' and "DATE" like '%2015%'
```

```
 * ibm_db_sa://nrx97436:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.
Done.
```

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

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

- 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 2010-03-20.
- Aapplied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

```
#order by count(landing__outcome) desc

%%sql
select landing__outcome, count(*) as launch_counts
from spacexdataset
where  landing__outcome in ('Failure (drone ship)', 'Success (drone ship)', 'Controlled (ocean)', 'Success (ground pad)',
                            'Failure (parachute)', 'Uncontrolled (ocean)', 'Precluded (drone ship)')
AND "DATE" BETWEEN '2010-06-04' and '2017-03-20'
group by landing__outcome
order by launch_counts
```

```
 * ibm_db_sa://nrx97436:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

landing__outcome	launch_counts
Precluded (drone ship)	1
Failure (parachute)	2
Uncontrolled (ocean)	2
Controlled (ocean)	3
Success (ground pad)	3
Failure (drone ship)	5
Success (drone ship)	5

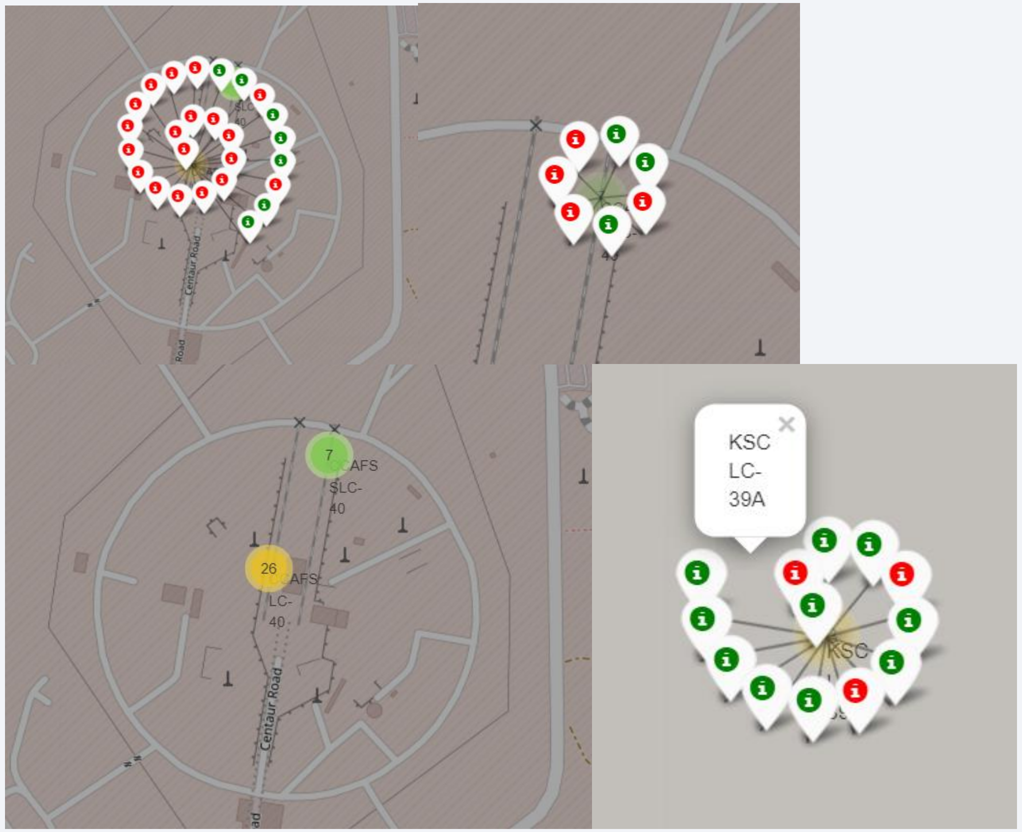
Section 3

Launch Sites Proximities Analysis

All launch sites global map markers

- SpaceX launch sites are in the United States Of America coasts. Florida and California

Markers showing launch sites with color labels

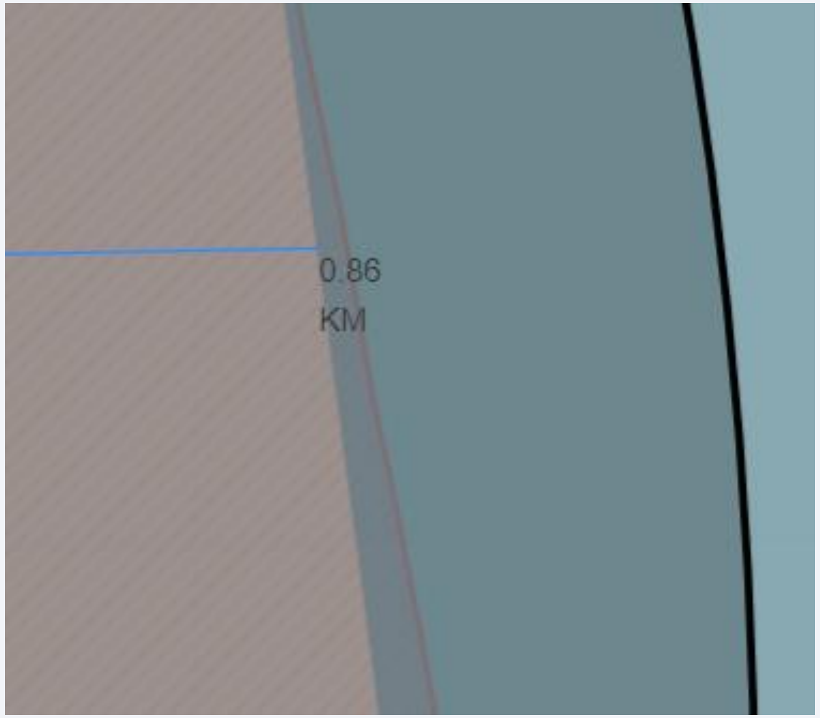


- Florida Launch Sites

- California Launch Sites

Launch Site distance to landmarks

Distance to Coast

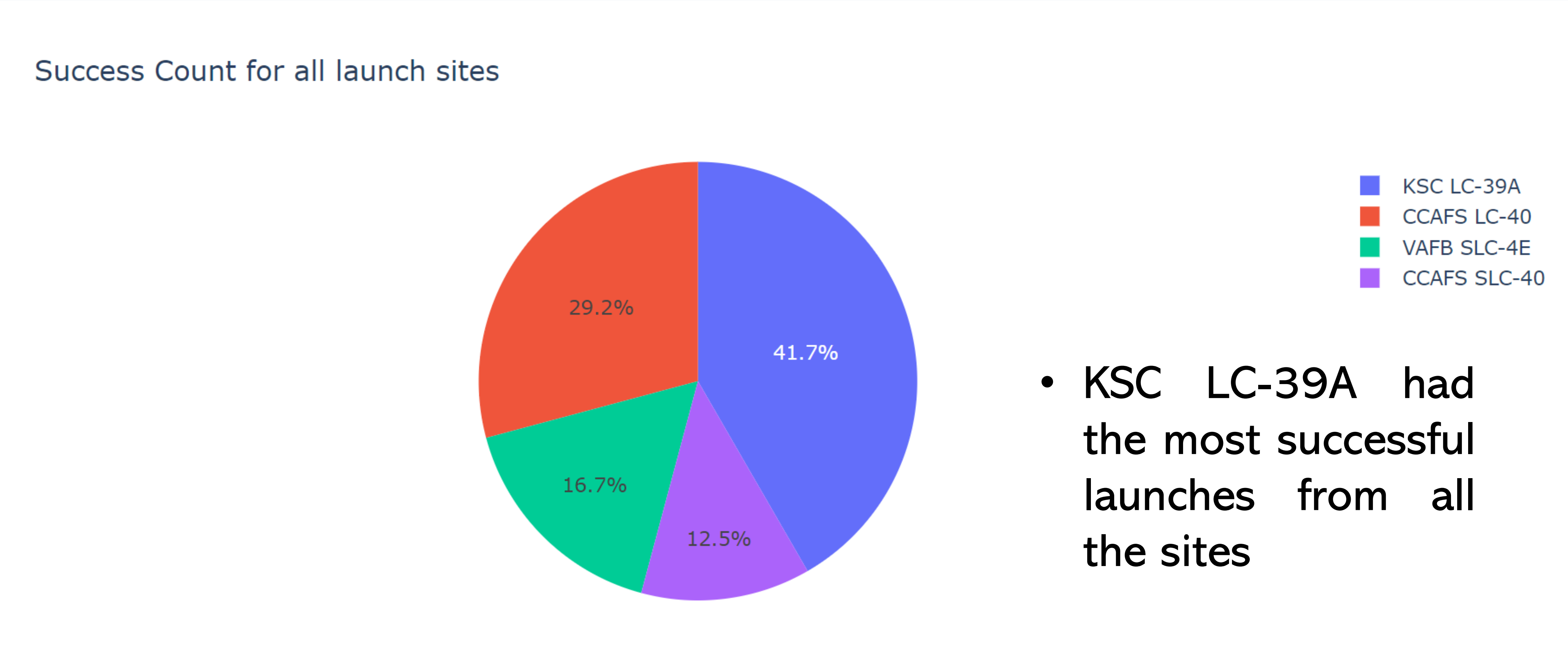


Distance to Coastline

Section 4

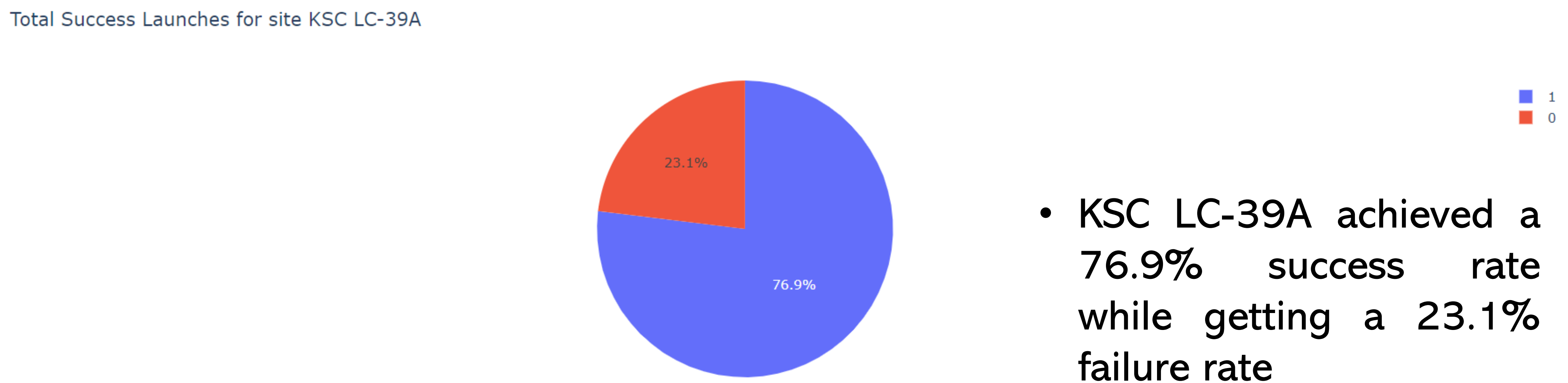
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



- KSC LC-39A had the most successful launches from all the sites

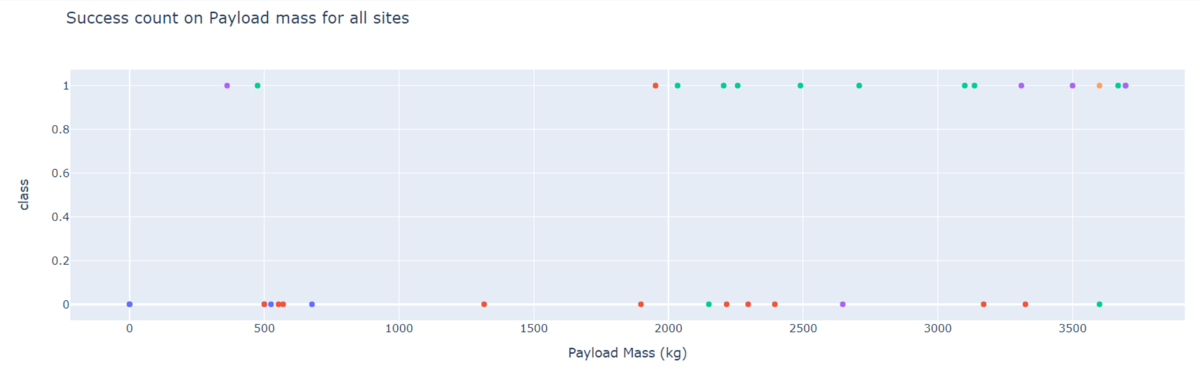
Pie chart showing the success percentage achieved by each launch site



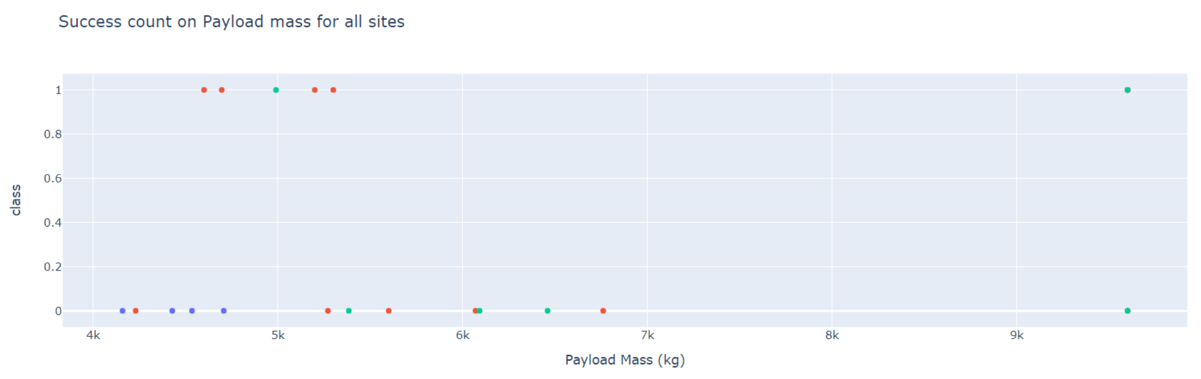
- KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

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

Low weighted Payload (0kg – 4000kg)



Heavy weighted Payload (4000kg – 10000kg)



Success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

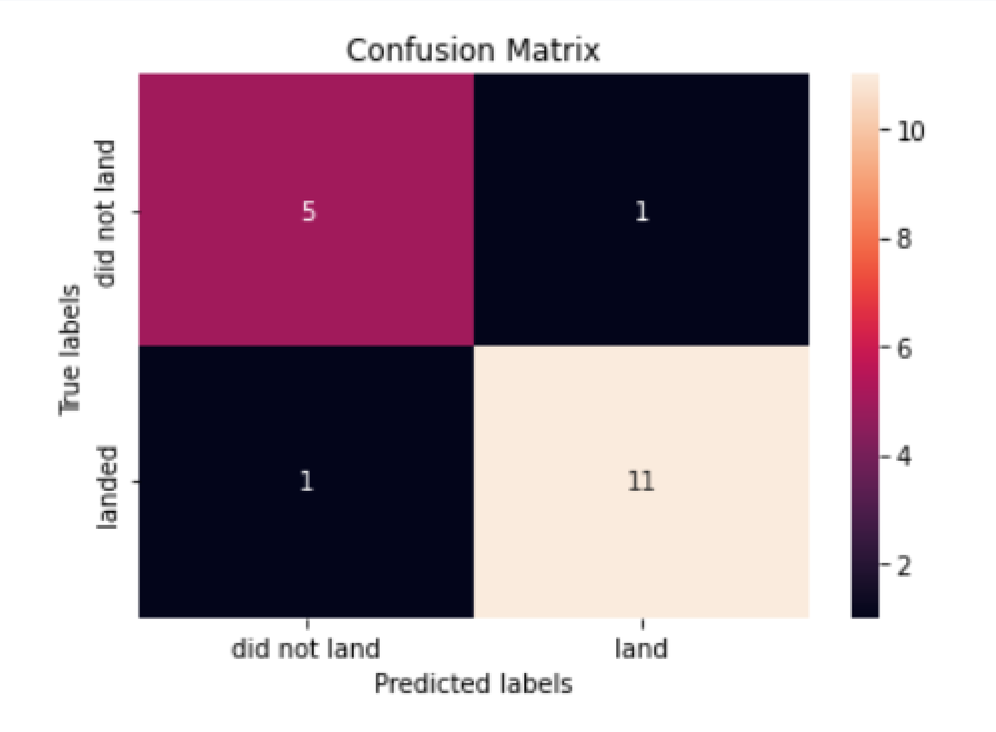
Predictive Analysis (Classification)

Classification Accuracy

	Model	Train Accuracy	Test Accuracy
0	Logistic Regression	0.84	0.83
1	SVM	0.84	0.83
2	Decision Tree	0.87	0.88
3	KNN	0.84	0.83

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

Confusion Matrix



- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The only problem is with the 1 false positives and 1 false negative.

Conclusions

Conclusion:

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 Decision tree classifier is the best machine learning algorithm for this task.

Thank you!