IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Kevin Jesus Apari>
<September 4th, 2022 >

Outline

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

Executive Summary

- Summary of methodologies

 - Data Collection

 - Data wrangling.

 - Exploratory Data Analysis (EDA) using SQL

 - Interactive dashboards.

 - Classification models (Supervised learning models)

- Summary of all results.

 - EDA insights

 - interactive dashboards

 - Prediction models results

Introduction

- Project background and context.

In this project we will predict whether the first land will be successful or not, in order to determine the cost of launching.

We will go through all the process that modelling data requires, from data collection using API and data wrangling to build a couple of supervised learning models to accomplish our outlined goal.

- Problems you want to find answers.

- what are the features that influence in the successful of the Falcon 9 landing.

Section 1

Methodology

Methodology

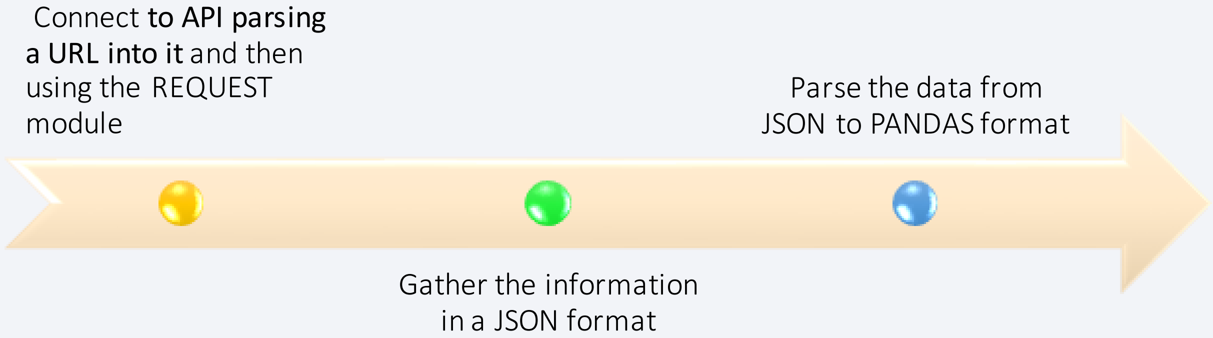
Executive Summary

- Data collection methodology:
  - Space X REST API
  - Web scraping
- Perform data wrangling
  - One hot encoding for certain features in order to stay aligned with the Machine Learning model assumption we use (I.e. classification model)
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Building a classification model, picking the best hyperparameters and then tuning it.

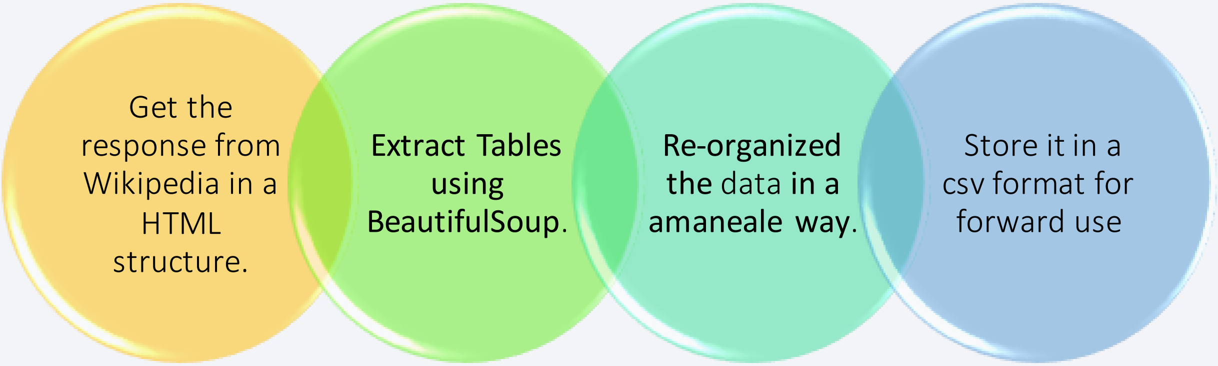
Data Collection

- Describe how data sets were collected.

A) The data was collected firstly via SpaceX REST API.



B) Then via scraping the Wikipedia tables information related to the Falcon 9 launches, all this using BeautifulSoup module.



Data Collection – SpaceX API

1. request the data

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)
```

2. extract the data into JSON format

```
df = pd.read_json(static_json_url)
```

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
df.head()
```

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
jlist = requests.get(static_json_url).json()
df2 = pd.json_normalize(jlist)
df2.head()
```

3. parse de data into PANDAS format

```
# Lets take a subset of our dataframe keeping only the features we want and the flight
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the d
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
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}
```

Data Wrangling

1) We have a dataset containing, among others feautures, one measurement on landing successful **[Outcome].**

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0

2) But this information is in categorical data format **{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}** which for our forward analysis will not be appropriate.

3) For that reason we need to pre-process it. We will do so combining the `apply()` method with `lambda` function.

```
#df['Class']=landing_class
df['Class'] = df['Outcome'].apply(lambda landing_class: 0 if landing_class in bad_outcomes else 1)
df[['Class']].head(8)
```

4) Finally we export our data.

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

EDA with Data Visualization

- For the EDA we used the following charts:

A) **Scatterplots**: In order to look for relationships among variables/features.

[Flight Number] vs **[Payload Mass]**; **[Launch Site]**

[Payload] vs **[Launch Site]**; **[Orbit Type]**

[Orbit] vs **[Flight Number]**; **[Payload Mass]**

B) **Barcharts:** in order to map the categorical variables behavior. In this case we want to visually check if there are any relationship between **[success]** rate and **[orbit type].**

C) **Lineplots:** in order to observe a variable evolution in time.

EDA with SQL

The SQL queries performed were:

- Displaying the names of the unique launch sites in the space mission.
- Displaying 5 records where launch sites begin with the string 'KSC’.
- Displaying the total payload mass carried by boosters launched by NASA (CRS).
- Displaying average payload mass carried by booster version F9 v1.1.
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- 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, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Ranking the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order

Build an Interactive Map with Folium

- In order to visualize the Launch Data into an interactive map we used the **[Latitude]** and **[Longitude]** features, then added a Circle Marker around each launch site.

- We assigned the `dataframe.launch_outcomes(failures, successes)` to classes 0 and 1 with green and red markers on the map in a `MarkerCluster()`

- Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns.

Build a Dashboard with Plotly Dash

- We used the following interactive charts:
  - **Pie Chart**: To show the total launches by a certain site.
  - **Scatter Graph:** To show the relationship between [Outcome] and [Payload Mass] .

Dash	Plotly	Charts
• Import dash • FImport dash_html_components as html • Import dash_core__components as dcc • From dash.dependencies import Inpuit, Output	• Import plotly.express as px	• Pie chart: px.pie() • Scatter chart : px.scatter()

Predictive Analysis (Classification)

Build the model

- Load our cleaned data
- Preprocess it according the model to use.
- Split it up
- Fit, and then Train it.

Evaluating the model

- Compute the scores in order to evaluate the model performance

Hyper-parameter Tuning

- Tune the hyper parameters in searchng the best one who deliver a well performer model.

Finding the best model

- Evaluate the diferent scores of the diferent best models and choose the best model

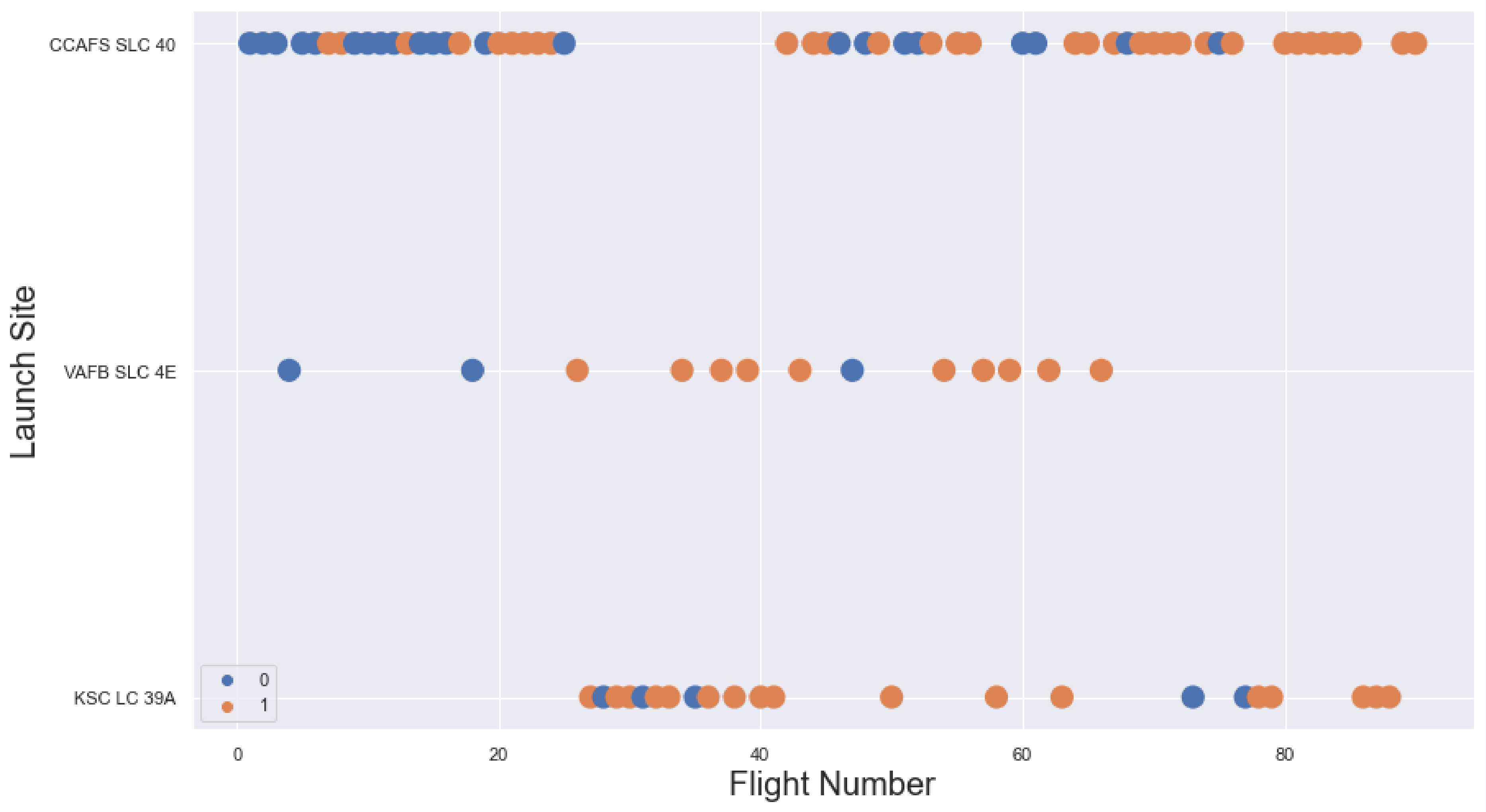
Results

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

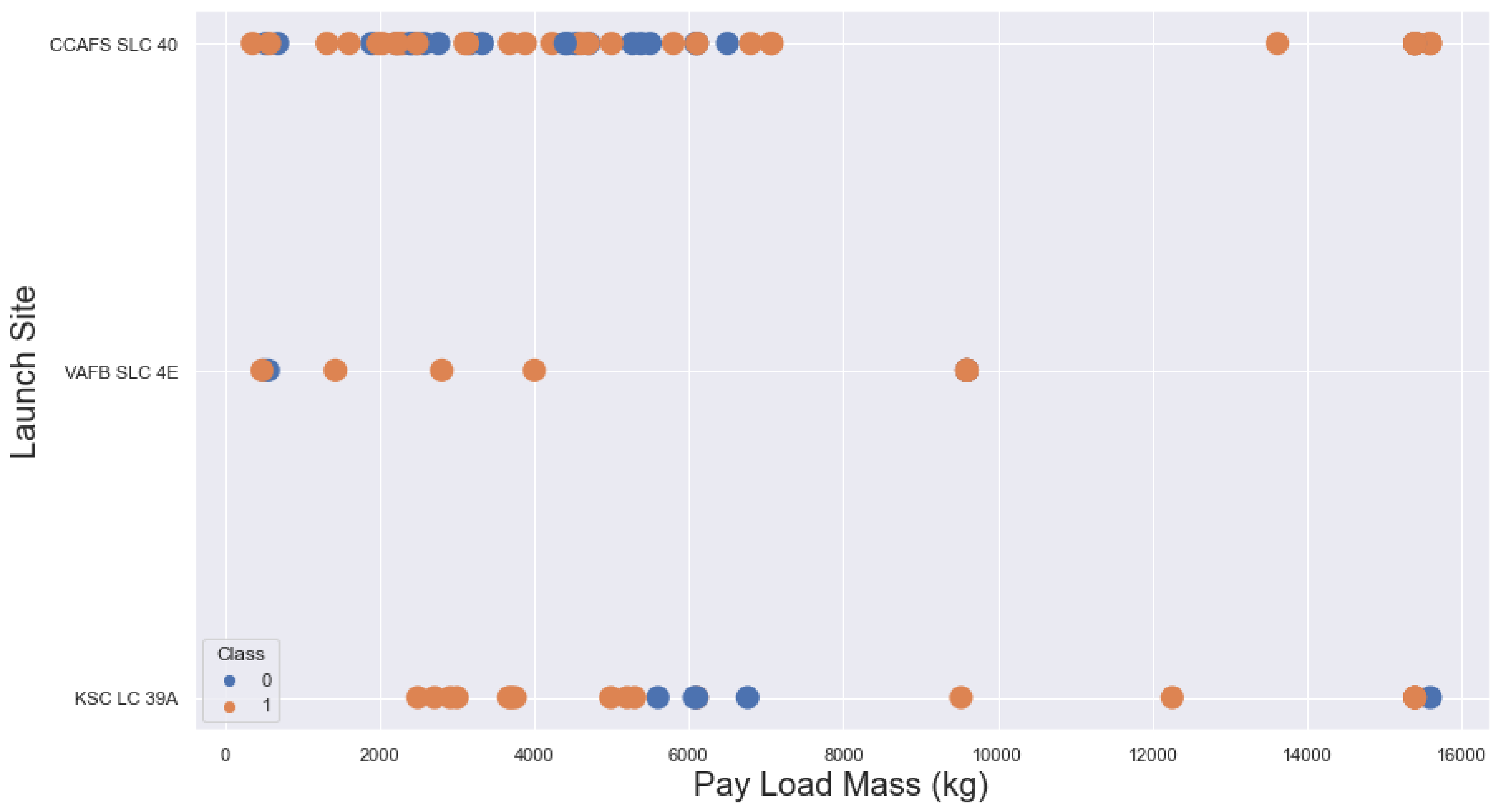
Section 2

Insights drawn from EDA

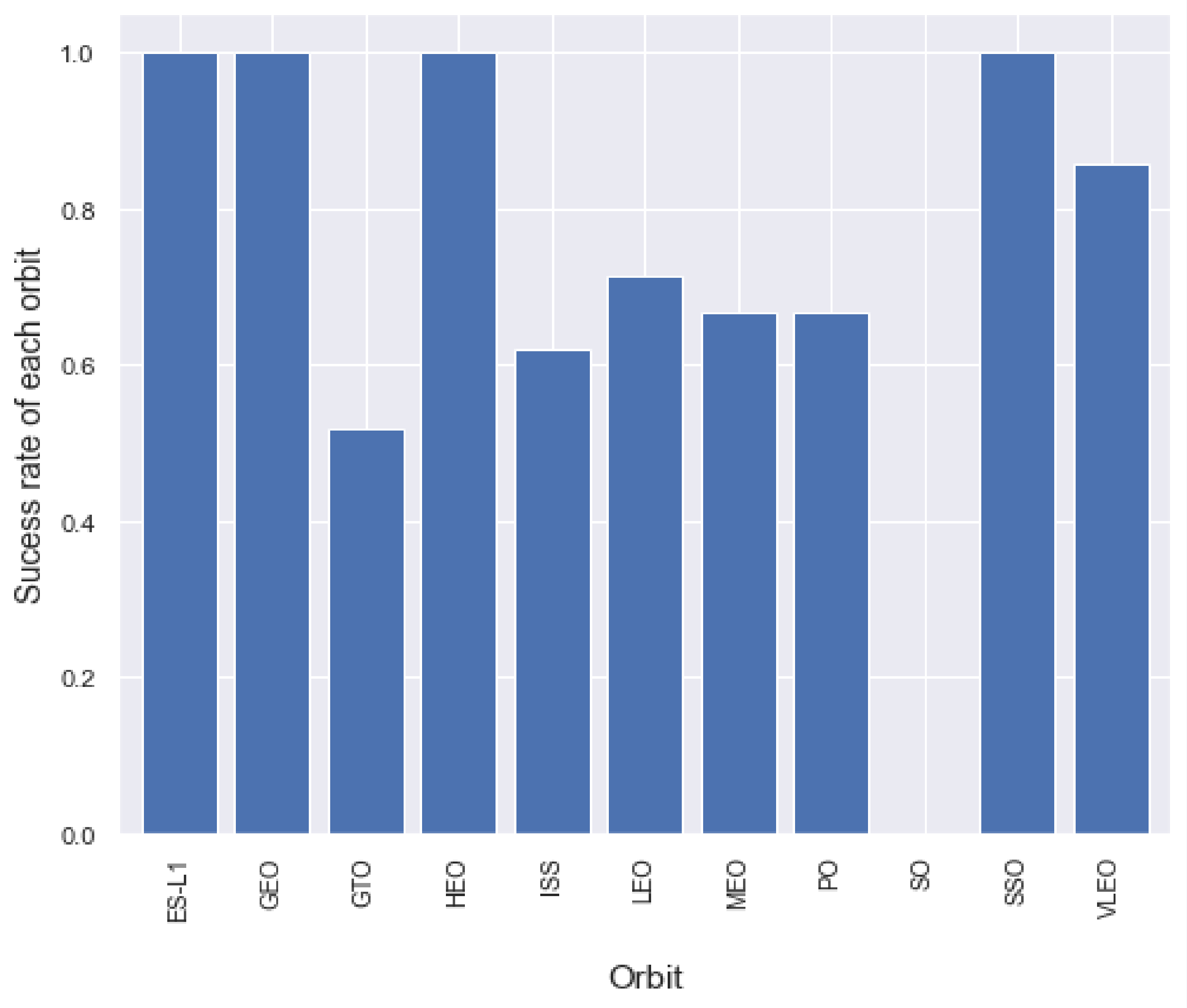
Flight Number vs. Launch Site



Payload vs. Launch Site

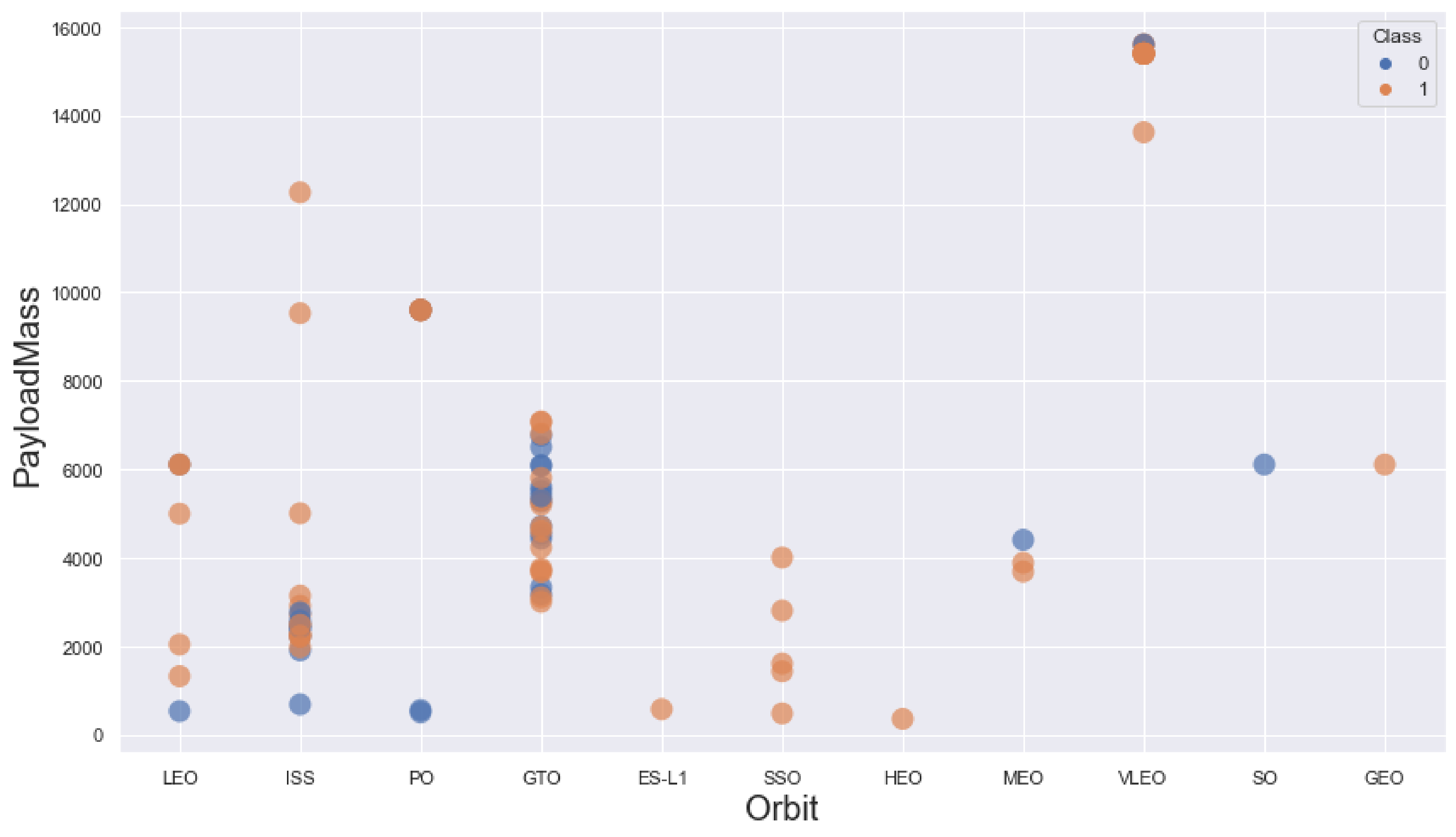


Success Rate vs. Orbit Type

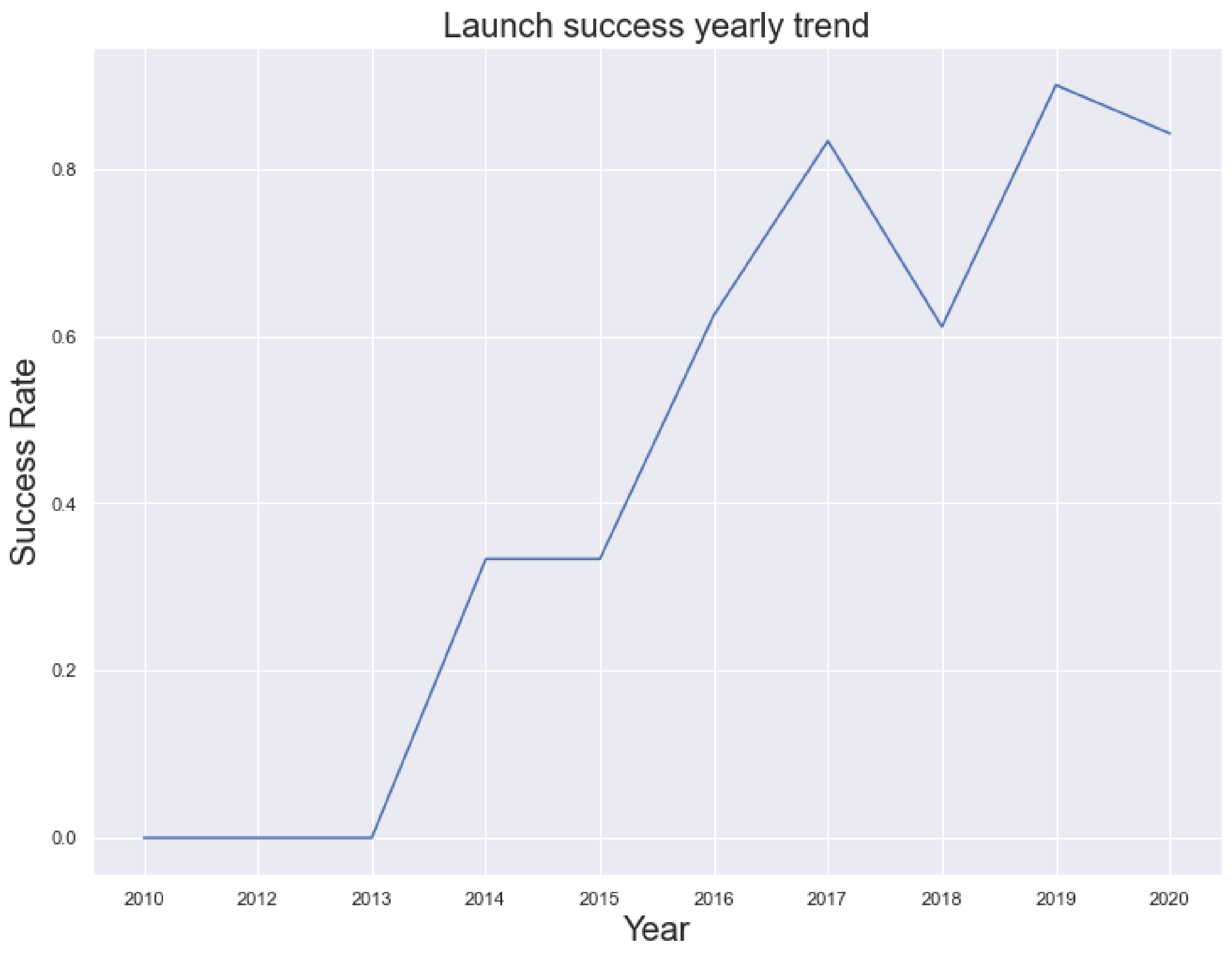


Flight Number vs. Orbit Type

Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

- SQL API python code:

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEX;
```

- Code meaning

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

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

Launch Site Names Begin with 'CCA'

- SQL API python code:

```
%sql SELECT * FROM SPACEX WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

- Code meaning

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

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

- SQL API python code:

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)" FROM SPACEX WHERE CUSTOMER = 'NASA (CRS)';
```

- Code meaning

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

Total Payload Mass by NASA (CRS)
45596

Average Payload Mass by F9 v1.1

- SQL API python code:

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster Version F9 v1.1" FROM SPACEX
WHERE BOOSTER_VERSION = 'F9 v1.1';
```

- Code meaning

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

Average Payload Mass by Booster Version F9 v1.1
2928

First Successful Ground Landing Date

- SQL API python code:

```
%sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad" FROM SPACEX \
WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

- Code meaning

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

First Succesful Landing Outcome in Ground Pad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- SQL API python code:

```
%sql SELECT BOOSTER_VERSION FROM SPACEX WHERE LANDING__OUTCOME = 'Success (drone ship)'
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

- Code meaning

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

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 API python code:

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Success%';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Successful Mission
100

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Failure Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Failure%';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Failure Mission
1

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Total Number of Successful and Failure Mission" FROM SPACEX \
WHERE MISSION_OUTCOME LIKE 'Success%' OR MISSION_OUTCOME LIKE 'Failure%';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Total Number of Successful and Failure Mission
101

- Code meaning

List the total number of successful and failure mission outcomes

Boosters Carried Maximum Payload

- SQL API python code:

- Code meaning

List the names of the **[booster_versions]** which have carried the maximum payload Mass.

```
%sql
SELECT DISTINCT BOOSTER_VERSION
AS "Booster Versions which carried the Maximum Payload Mass"
FROM SPACEX
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_)
                           FROM SPACEX);
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.
Done.
```

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

2015 Launch Records

- SQL API python code:

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE DATE LIKE '2015-%' AND
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde0
Done.
```

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

- Code meaning

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

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

- SQL API python code:

```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEX \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.clo
Done.
```

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

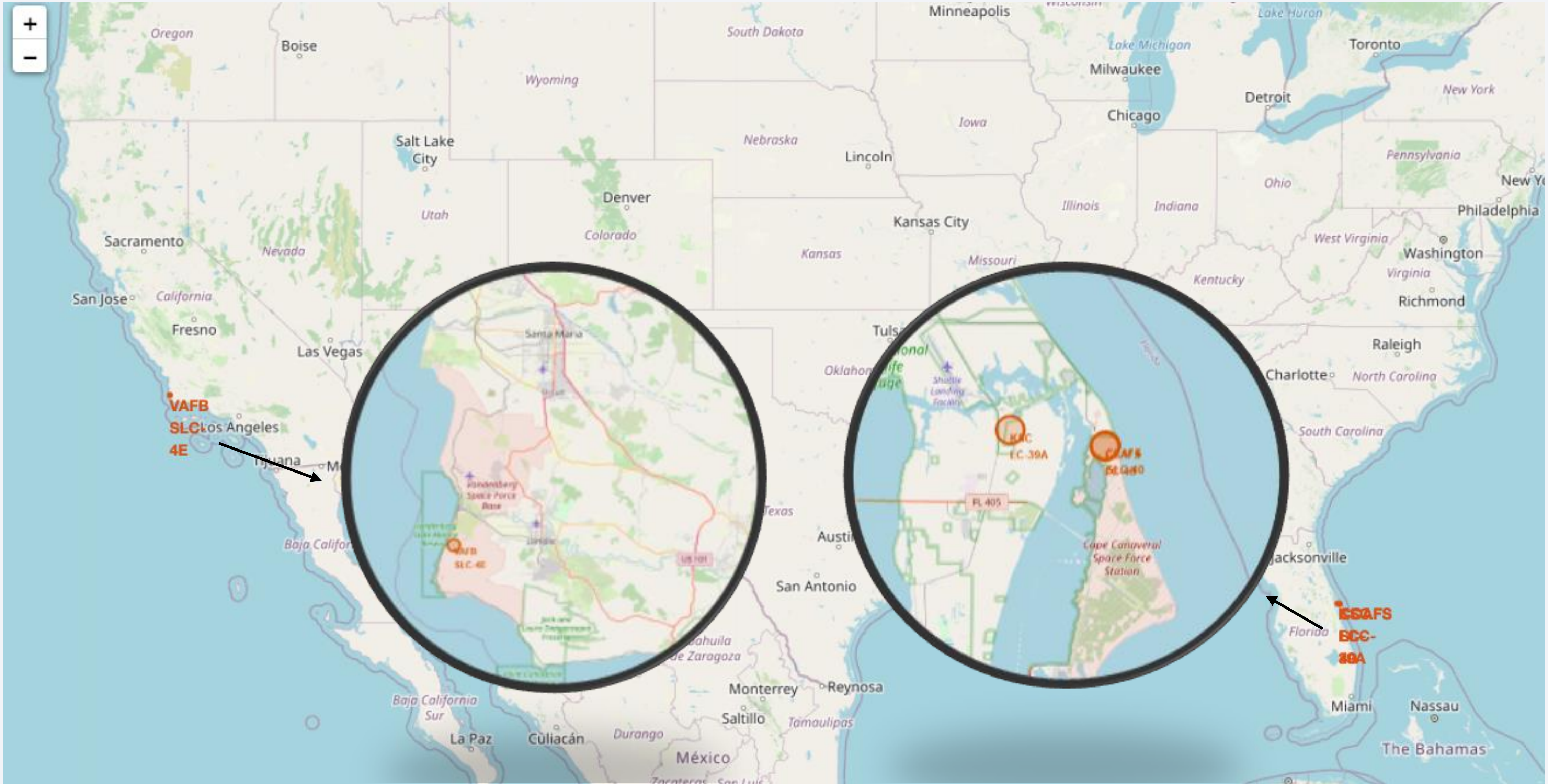
- Code meaning

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

Section 3

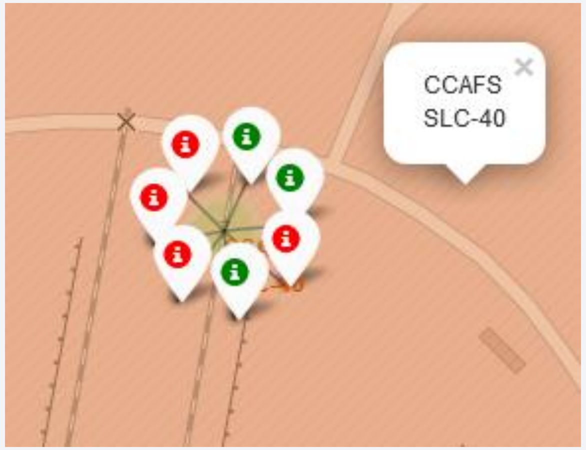
Launch Sites Proximities Analysis

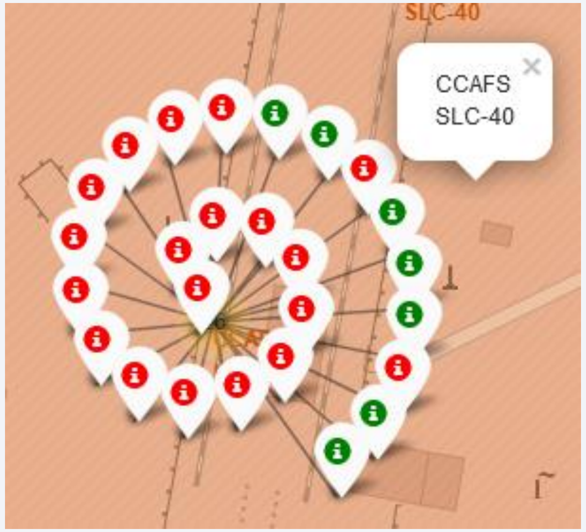
All launch sites

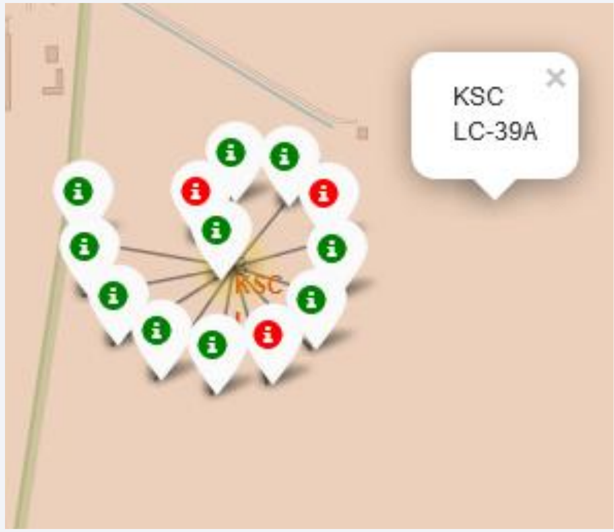


Launch records labeled by color

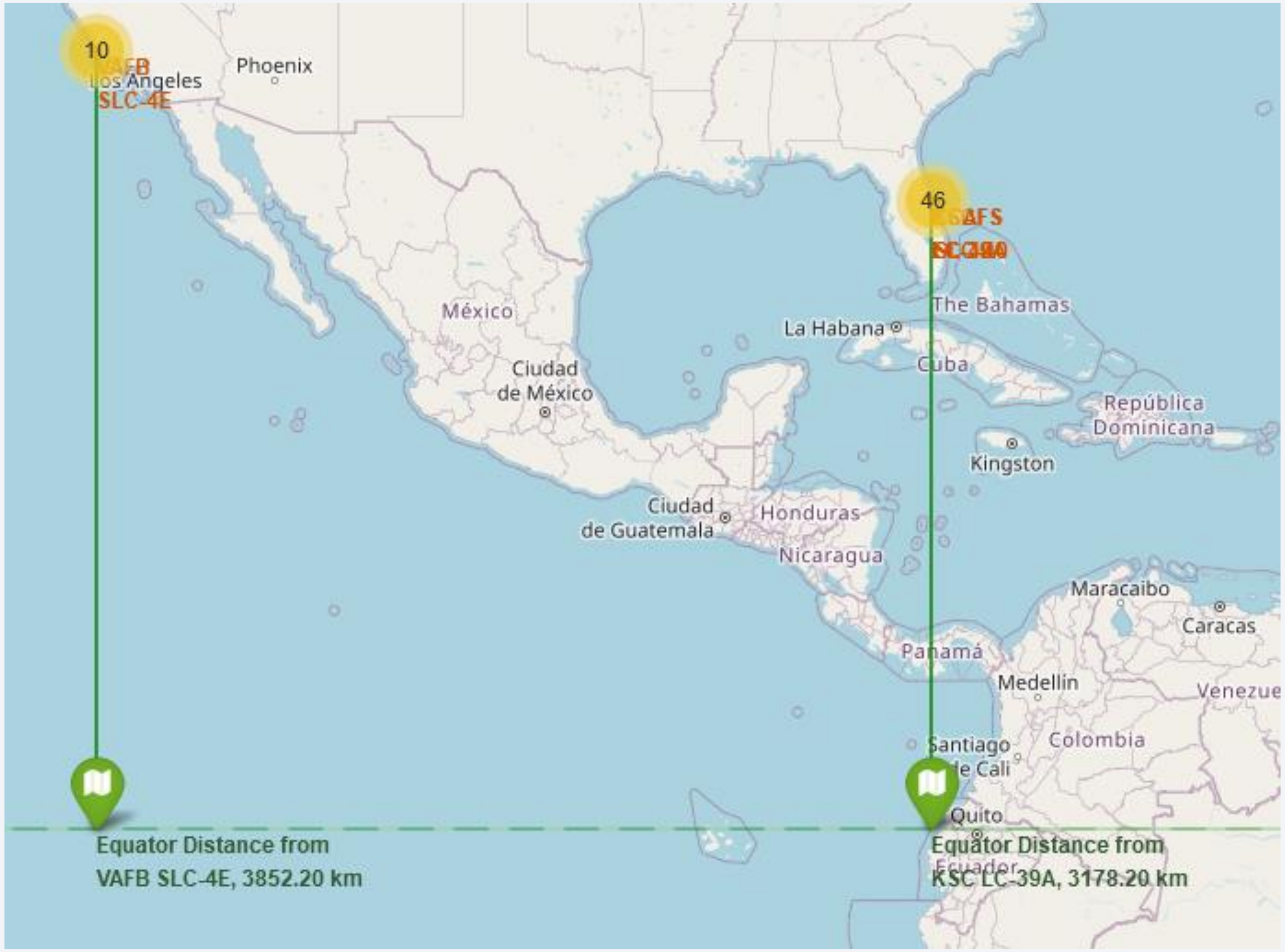
- Red -> failed launches | Green -> Succesful launches



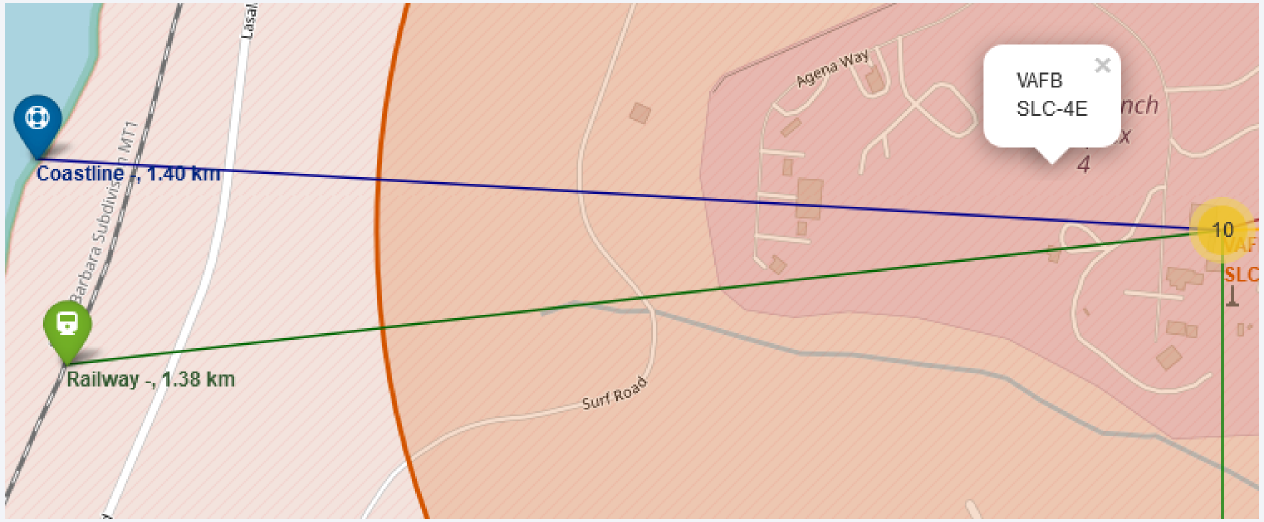


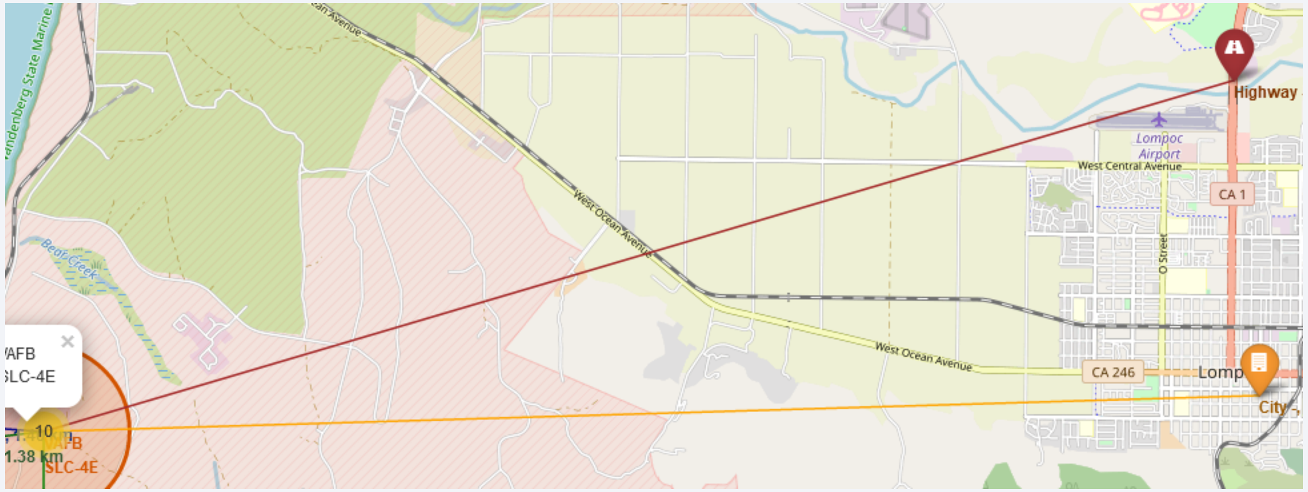


Launch site distance from the Equator



Launch site distance from coastline, railway, highway and city.

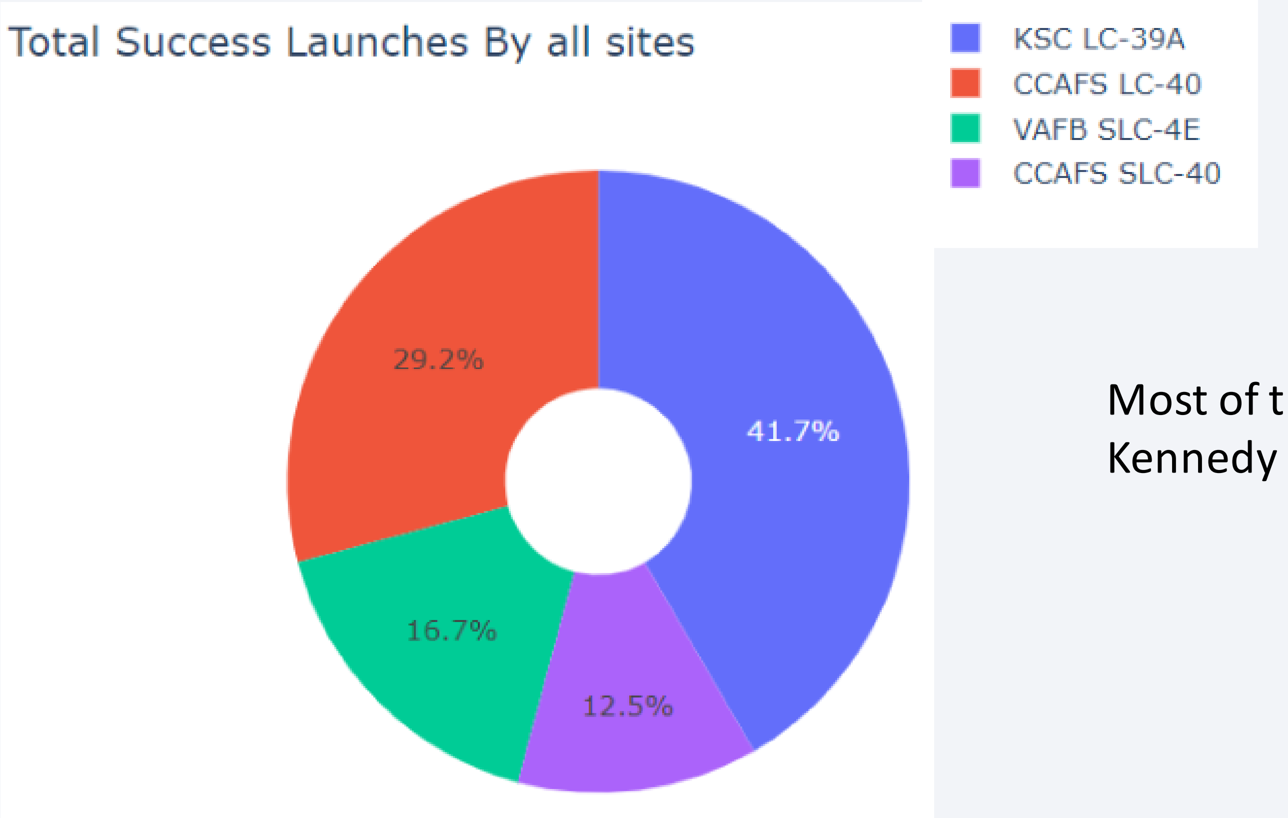




Section 4

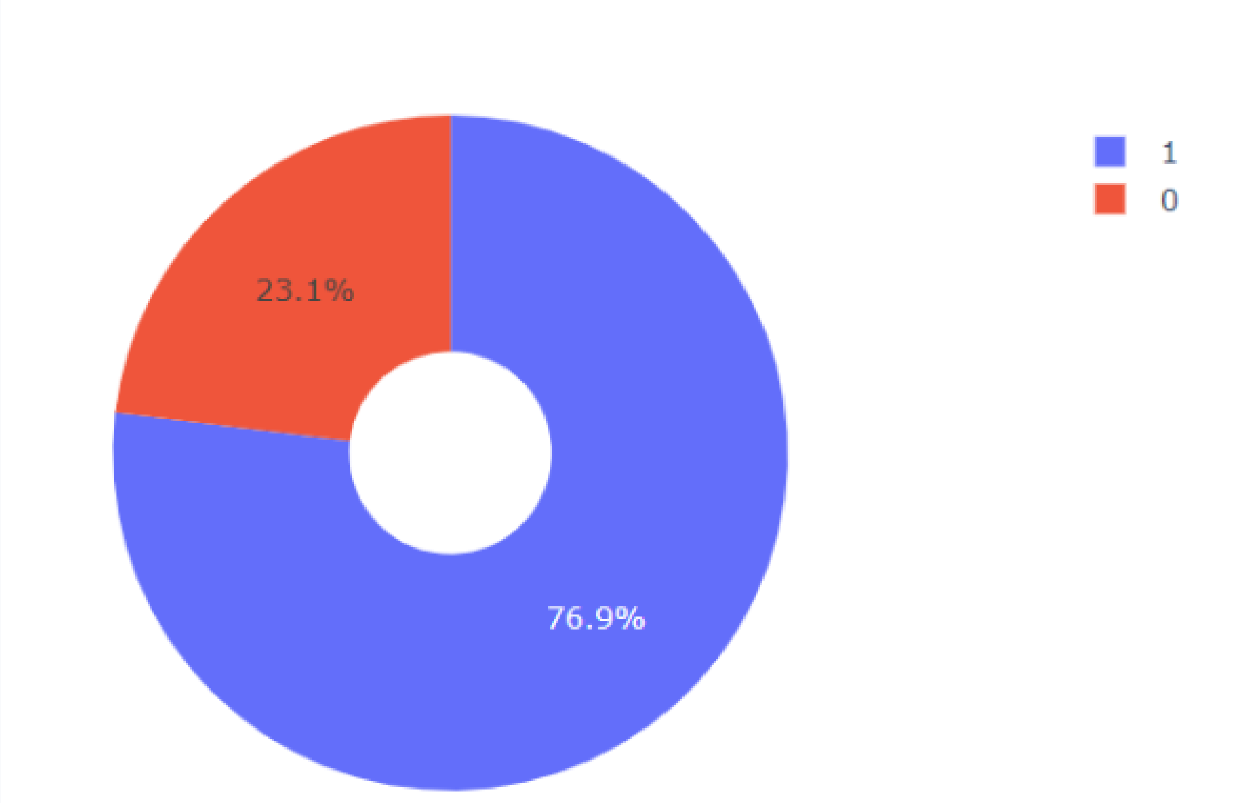
Build a Dashboard with Plotly Dash

Pie Chart of launch success count for all sites.



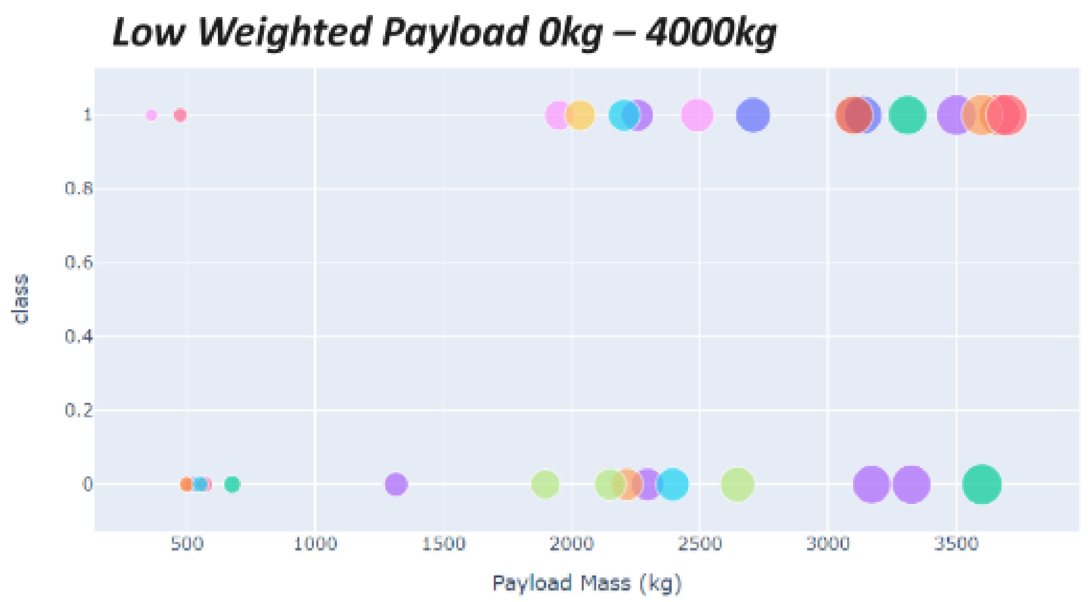
Most of the succeful launches come from the Kennedy Space Center (42%).

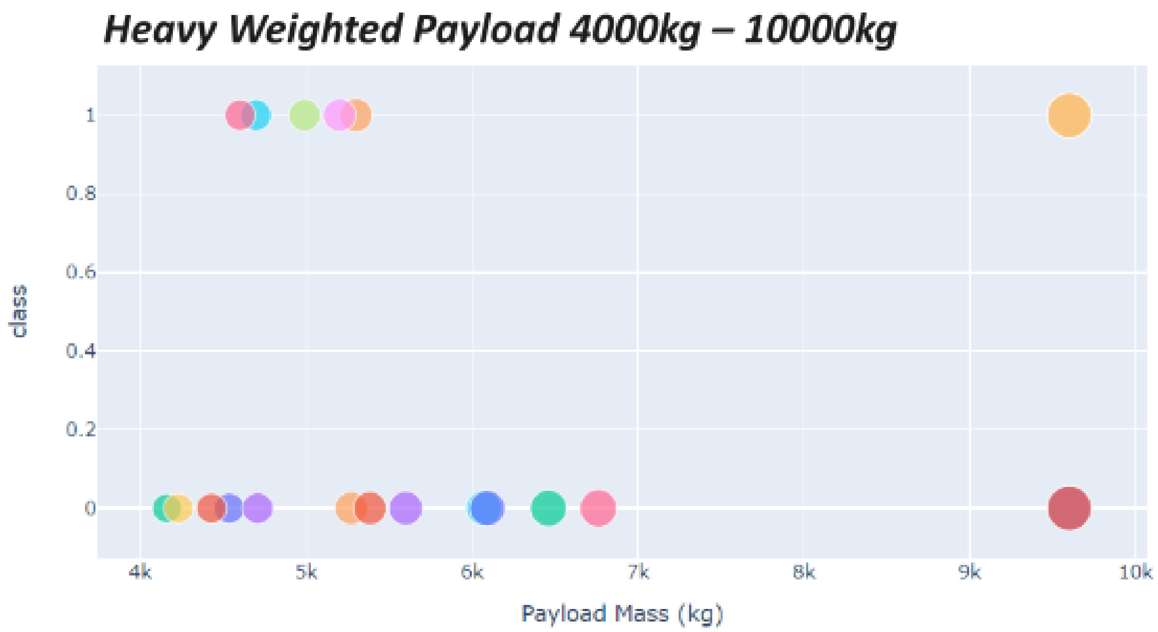
Pie Chart of the launch sit with highest launch success ratio.



The Kennedy space center KSC LC-39A has a 77% success rate.

Scatter plot for Payload vs Launch Outcome





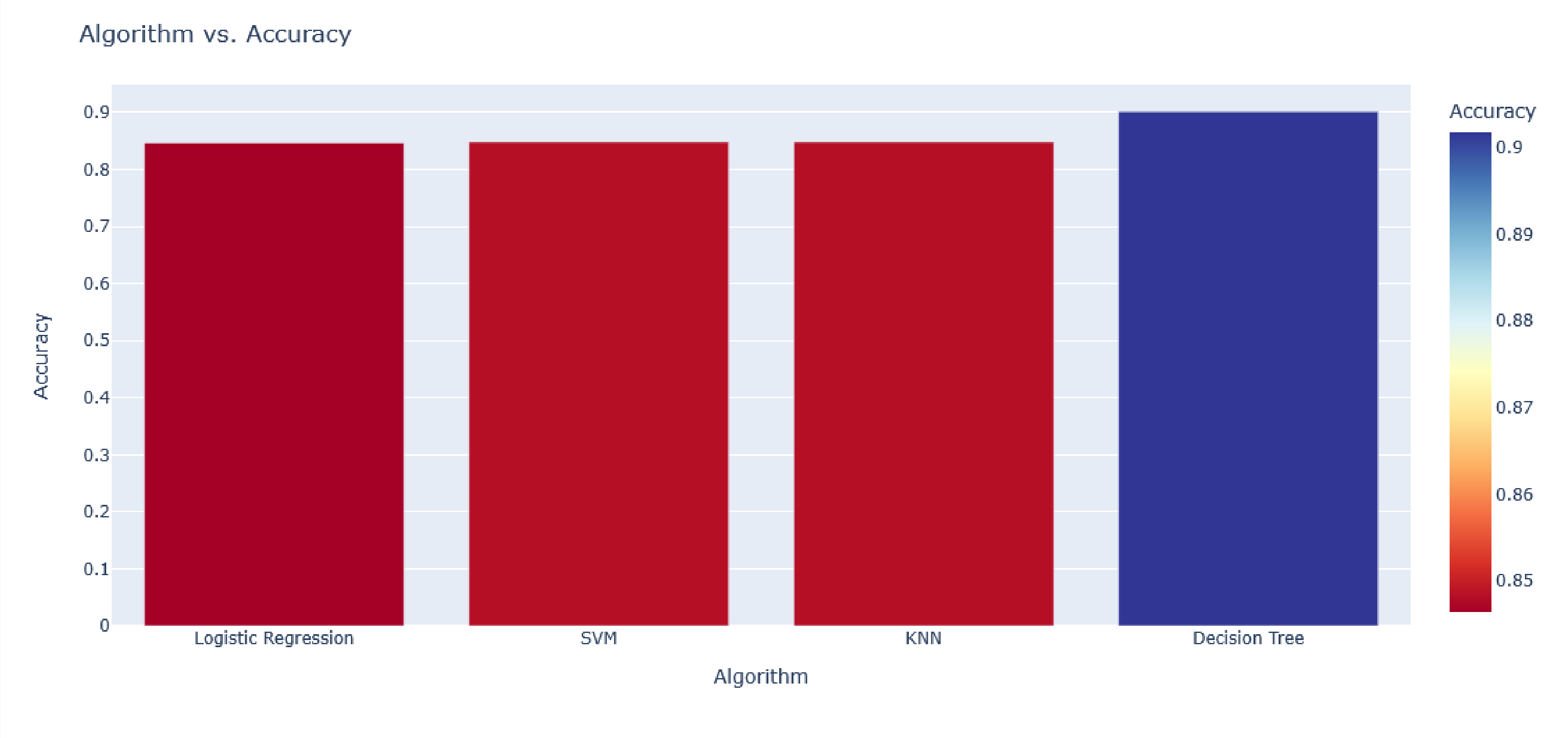
We notice that the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

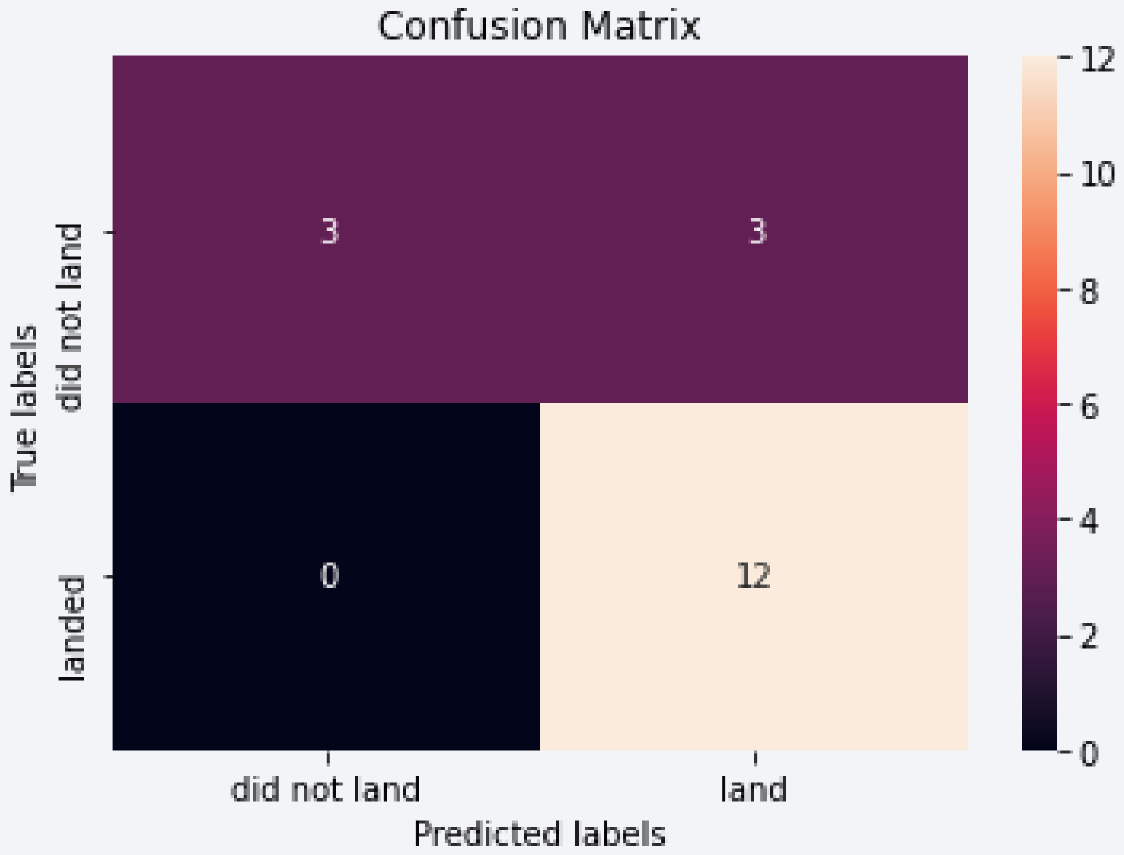
All models show roughly the same performance, but the Decision Tree show a slightly better accuracy score, which tempt us to choose it as best model.



Confusion Matrix

The model show a good performance classifying the "landed" label, but a poor performance classifying the "no land" label.

All this means that even we tuned our model, maybe we should consider employ another models.



Conclusions

- Most of the features we considered in the model are meaningful.
- Our data suffer from imbalance class, we need to treat it.
- Despite the preprocess and data wrangling we performed, the model we bulit have not a very good performance, which gives a room for improvement.
- From the models built, the Decision tree shows the best performance.

Thank you!