IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Szabolcs Vetési

2022.08.28.

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection with webscraping and API
  - EDA (Exploratory Data Analysis in SQL and python)
  - Interactive Visual Analytics with Folium
  - Building a dashboard with plotly and Dash
  - Machine Learning Prediction
- Summary of all results
  - EDA results
  - Interactive Analytics in screenshots
  - Prediction results

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 to be answered
  - What factors determine if the rocket will land successfully?
  - The interactions of features that correspond to the success rate a landing.
  - What conditions need to be satisfied to ensure a successful landing?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping 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 webscraping techniques
  - The core dataset was obtiained by a get request to SpaceX API
  - Next, it was decoded as a json and converted into a pandas dataframe
  - Then we cleaned the data and treated missing values where it was necessary
  - In addition, we performed web scraping from Wikipedia for historical Falcon 9 launch records with BeautifulSoup.
  - In the end the dataframe was exported as a csv

Data Collection – SpaceX API

- API call
- Webscraping more data
- Data cleaning
- Data wrangling

Notebook available at:

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

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)

In [19]: # Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())

In [21]: # Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
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 rocket boosters and rows that have mult
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 the list and replace the feature.
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 date leaving the time
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)]

In [149]: # Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df.loc[df['BoosterVersion']!='Falcon 1', :]

In [152]: # Treating missing values
# Calculate the mean value of PayloadMass column
payloadmass_mean = data_falcon9["PayloadMass"].mean()
# Replace the np.nan values with its mean value

# Using everything mentioned above results in very long code or Warning
if False:
    data_falcon9.loc[np.isnan(data_falcon9["PayloadMass"]), "PayloadMass"] = \
        data_falcon9.loc[np.isnan(data_falcon9["PayloadMass"]), "PayloadMass"].replace(np.NaN, payloadmass_mean)

    data_falcon9.loc[:, "PayloadMass"].replace(np.NaN, payloadmass_mean, inplace=True) # -> Warning

# This creates the same result but without replace
data_falcon9.loc[np.isnan(data_falcon9["PayloadMass"]), "PayloadMass"] = payloadmass_mean
```

Data Collection - Scraping

- We scraped Falcon 9 launch records with BeautifulSoup then parsed the table and converted it into a pandas dataframe
- Available: https://github.com/vetszabolcs/IBM_capstone/blob/main/jupyter-labs-webscraping.ipynb

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

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

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text)
```

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

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

In [8]:
```
# 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")
```

In [15]:
```
for k in list(launch_dict.keys()):
    print(f"{k}: {len(launch_dict[k])}")
    if len(launch_dict[k]) == 0:
        launch_dict.pop(k)
```

```
Flight No.: 121
Launch site: 121
Payload: 121
Payload mass: 0
Orbit: 121
Customer: 121
Launch outcome: 121
Version Booster: 121
Booster landing: 121
Date: 121
Time: 0
```

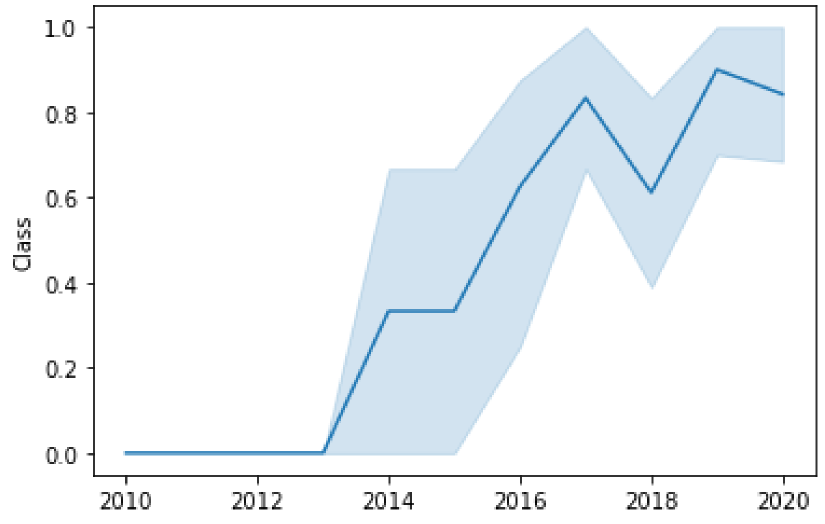
In [16]:
```
df=pd.DataFrame(launch_dict, )
```

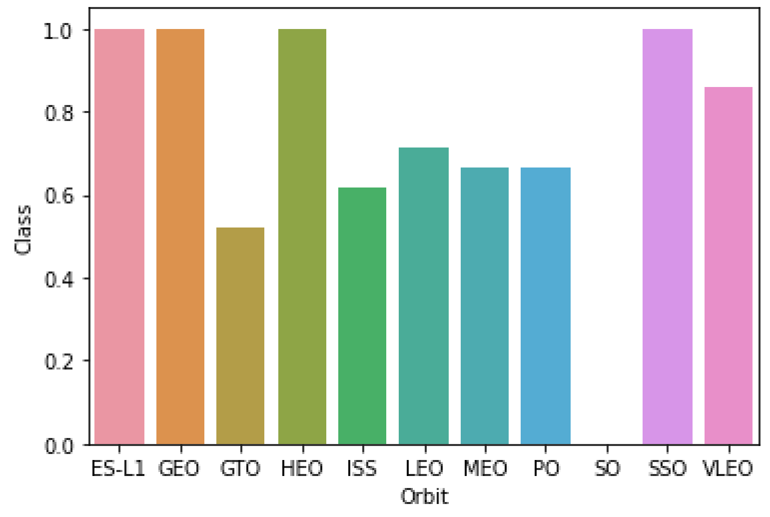
Data Wrangling

- We performed exploratory data analysis and calculated the number of launches at each site, with the number and occurrence of each orbits
- We also created landing outcome label and exported the results to a csv.
- Available: https://github.com/vetszabolcs/IBM_capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- We visualized how Flight Number, Pay Load Mass, Launch Site and Orbit type affects success rate, the annaul trends and these features relationships with each other.
- Available: https://github.com/vetszabolcs/IBM_capstone/blob/main/jupyter-labs-eda-dataviz.ipynb





EDA with SQL

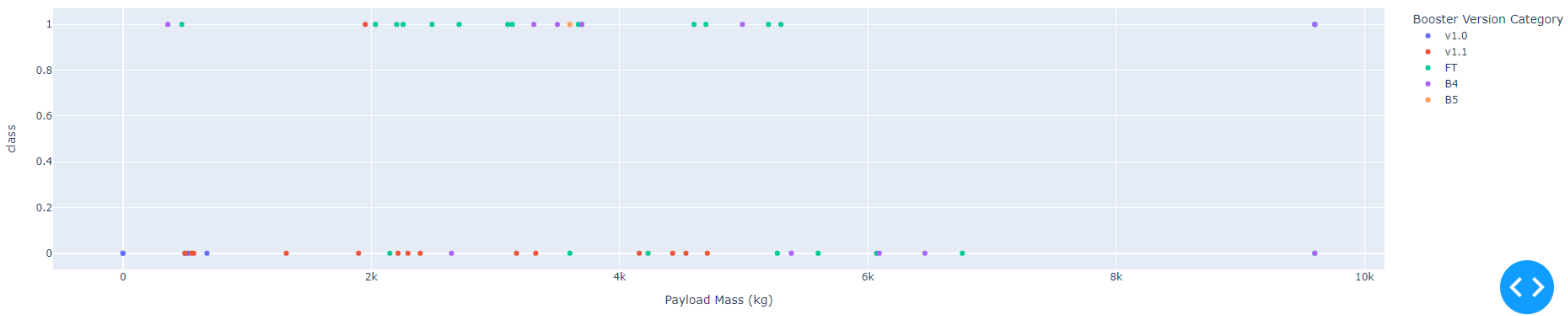
- We created a PostgreSQL database from the SpaceX dataframe with ipython's SQL magic
- We wrote queries to get insights about the payload mass carried by different boosters
  - Number of successful and failure outcomes
  - Average payload mass carried by booster version F9 v1.1
  - Outcomes throught time
  - Etc.
- Available: https://github.com/vetszabolcs/IBM_capstone/blob/main/jupyter-labs-eda-sql-coursera.ipynb

Build an Interactive Map with Folium

- We marked launch sites, and created several map objects (like markers and circles) to mark the outcome of launches for each sites
- We also used colored markers to show the success reate of sites
- We calculated the distances between a launch site to its proximities
- Available: https://github.com/vetszabolcs/IBM_capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We created an interactive dashboard with dash and plotly
- The dashboard included pie charts showing the launches by certain sites and scatter plots showing the relationship between Outcome and Payload Mass
- Available: https://github.com/vetszabolcs/IBM_capstone/blob/main/main.py



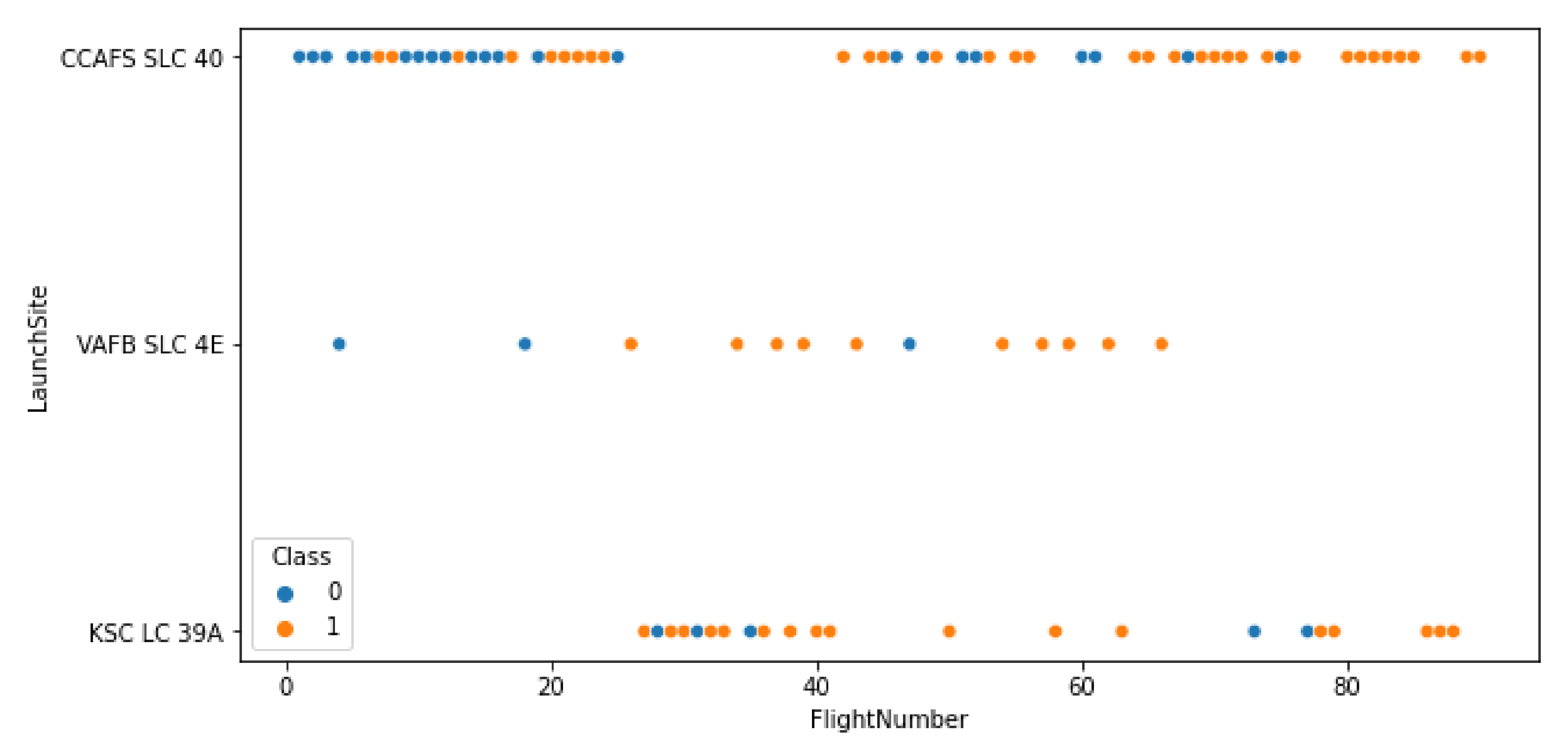
Predictive Analysis (Classification)

- We loaded and preprocessed the data (standardized, splitted)
- We applied GridSearch with several classification models and avaluated which had the best accuracy
- In this case the best performing model was decision tree (83.3% out of sample and 87.7% in sample accuracy)
- Available: https://github.com/vetszabolcs/IBM_capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Section 2

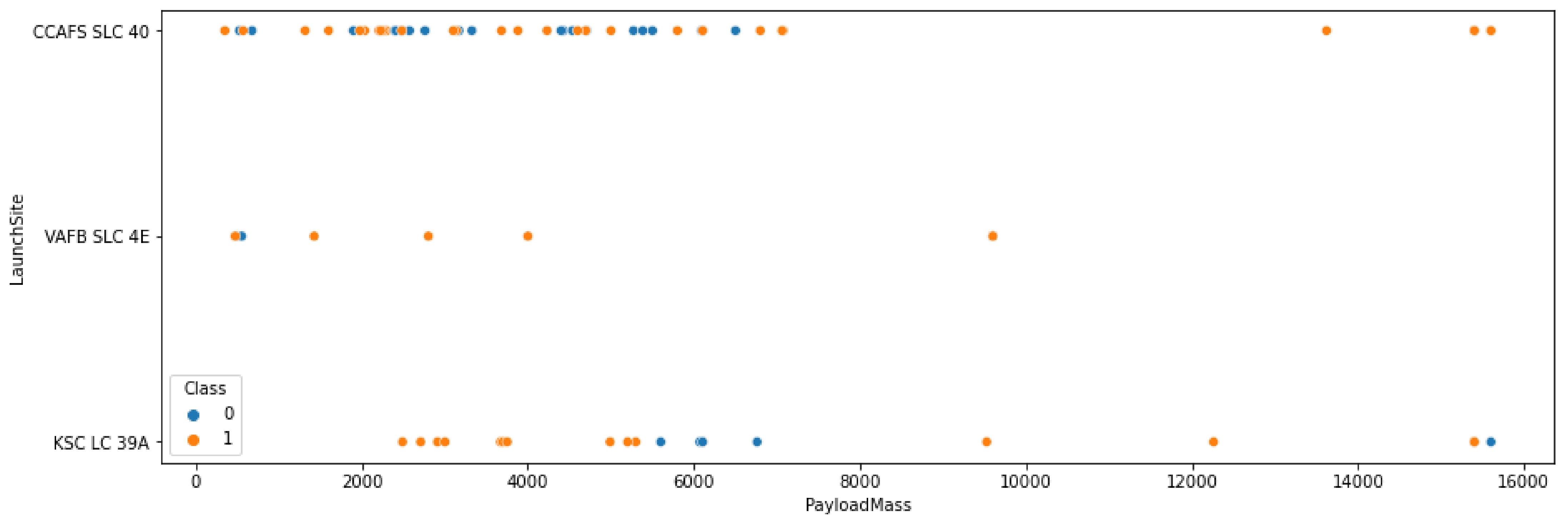
Insights drawn from EDA

Flight Number vs. Launch Site



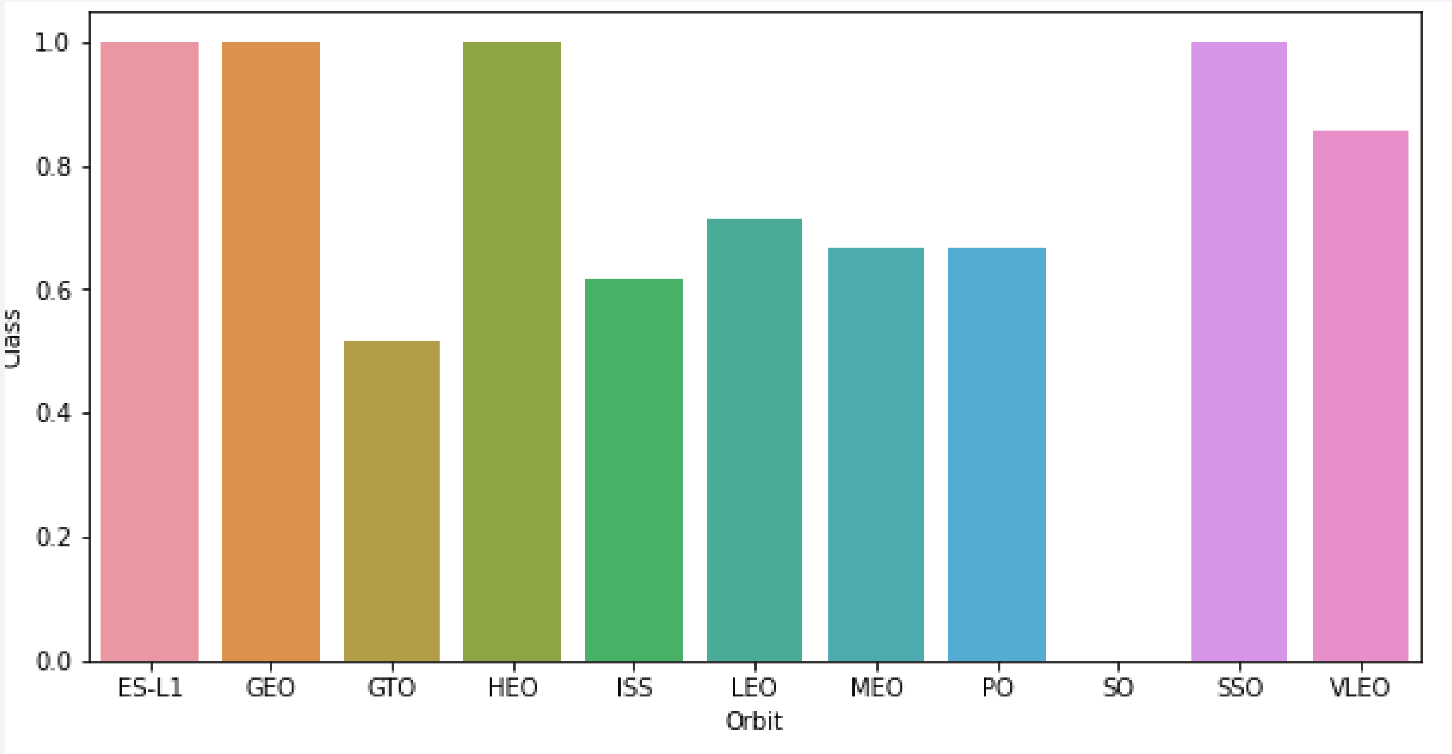
As flight number increases success rate also increases

Payload vs. Launch Site



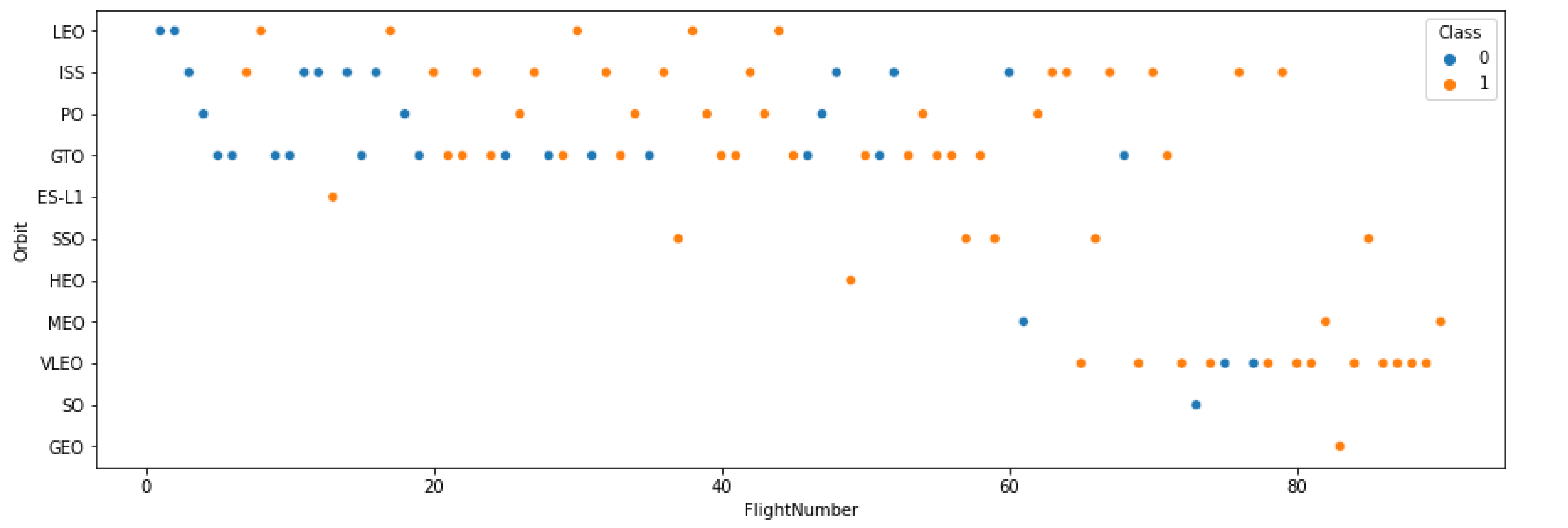
Heavy payload (>8000 kg) is rare and heavier payload mass positively correlates with success rate.

Success Rate vs. Orbit Type



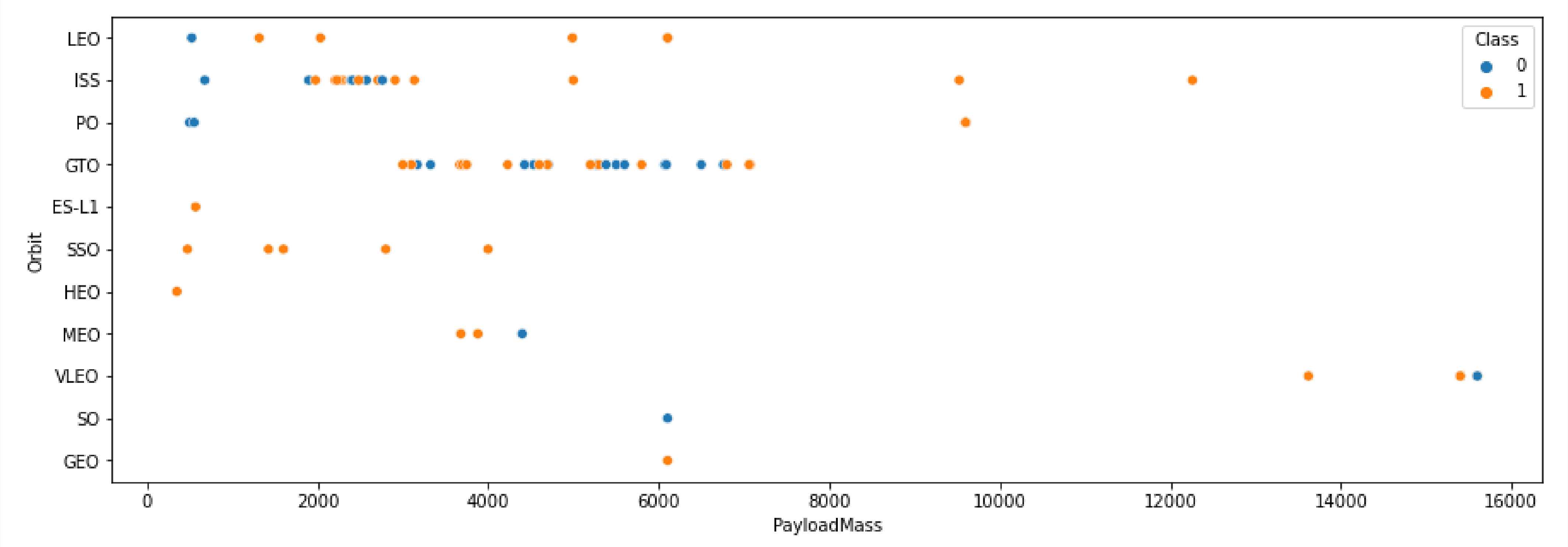
ES-L1, GEO, HEO and SSO had the highest success rate. SO had only failures.

Flight Number vs. Orbit Type



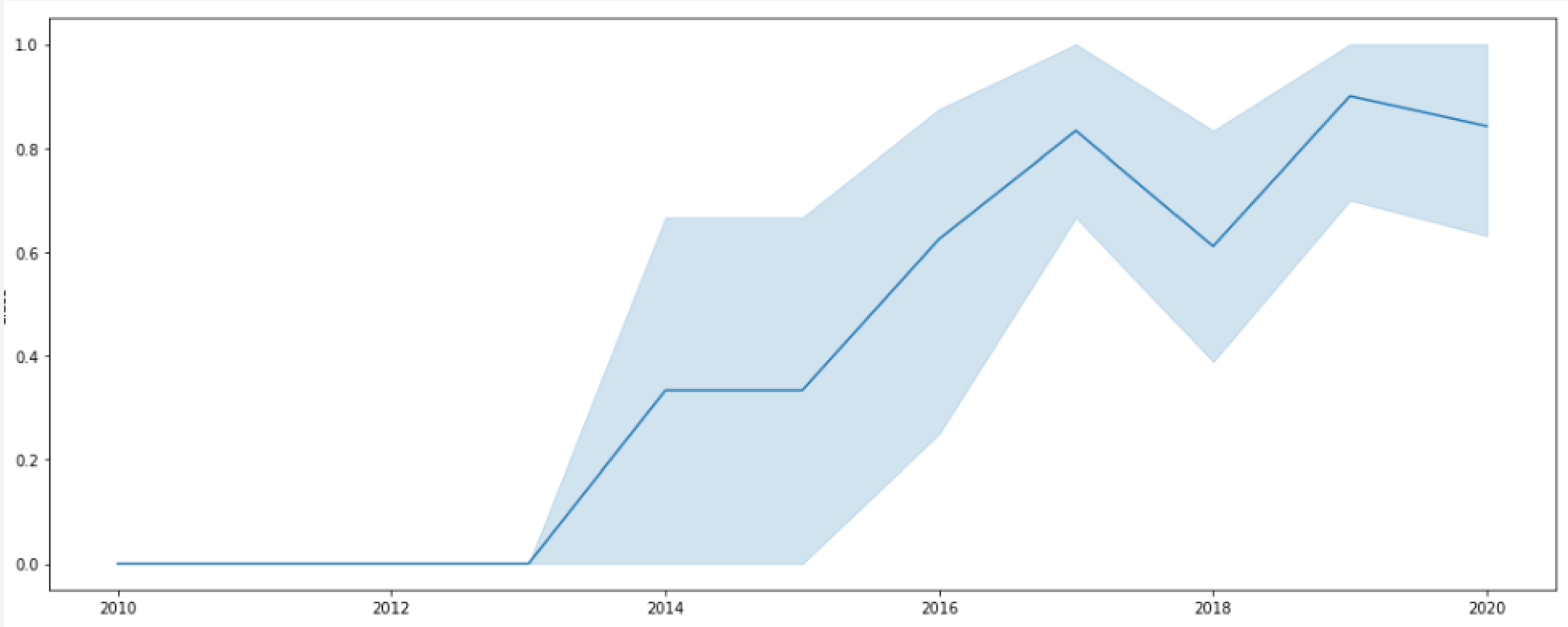
In the LEO orbit the Success appears related to the number of flights although, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for PO, 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



The sucess rate since 2013 kept increasing till 2020

All Launch Site Names

We used the DISTINCT statement to return only the unique sites.

```
%%sql
select distinct "Launch_Site" from ibm.spacex
```

```
 * postgresql://postgres:***@localhost:5432/postgres
4 rows affected.
```

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

Launch Site Names Begin with 'CCA'

```
%%sql
select * from ibm.spacex
where "Launch_Site" like 'CCA%'
limit 5
```

```
 * postgresql://postgres:***@localhost:5432/postgres
5 rows affected.
```

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

We used the LIKE operator to find strings containging CCA and limited our results to 5 with limit

Total Payload Mass

```
%%sql
select sum("PAYLOAD_MASS__KG_") as "TPM by NASA (CRS)" from ibm.spacex
where "Customer" = 'NASA (CRS)'
```

```
 * postgresql://postgres:***@localhost:5432/postgres
1 rows affected.
```

TPM by NASA (CRS)
45596

We calculated the TPM with above query
We also renamed our resulting column

First Successful Ground Landing Date

```
%%sql
select "Mission_Outcome", min("Date") from ibm.spacex
where "Mission_Outcome" = 'Success'
group by "Mission_Outcome"
limit 1
```

```
 * postgresql://postgres:***@localhost:5432/postgres
1 rows affected.
```

Mission_Outcome	min
Success	2010-06-04

We parsed successful outcomes and selected the first minimum value of Date column for it

Average Payload Mass by F9 v1.1

```
%%sql
select round(avg("PAYLOAD_MASS__KG_"), 3) from ibm.spacex
where "Booster_Version" like '%F9 v1.1%'
```

```
 * postgresql://postgres:***@localhost:5432/postgres
1 rows affected.
```

round
2534.667

We calculated and rounded the average of paload mass where Booster Version Included F9 v1.1

Successful Drone Ship Landing with Payload between 4000 and 6000

We selected Payload column where Landing Outcome met multiple conditions

```
%%sql
select "Payload" from ibm.spacex
where "Landing _Outcome" like '%uccess%'
    and "PAYLOAD_MASS__KG_" > 4000
    and "PAYLOAD_MASS__KG_" < 6000
```

```
 * postgresql://postgres:***@localhost:5432/postgres
14 rows affected.
```

Payload
JCSAT-14
JCSAT-16
SES-10
NROL-76
Boeing X-37B OTV-5
SES-11 / EchoStar 105
Zuma
Merah Putih
Es hail 2
Nusantara Satu, Beresheet Moon lander, S5
RADARSAT Constellation, SpaceX CRS-18
GPS III-03, ANASIS-II
ANASIS-II, Starlink 9 v1.0
GPS III-04 , Crew-1

Total Number of Successful and Failure Mission Outcomes

```
%%sql
select
    count(*) filter (where "Mission_Outcome" like '%uccess%') as "Success Count",
    count(*) filter (where "Mission_Outcome" like '%ail%') as "Failure Count"
from ibm.spacex
```

```
 * postgresql://postgres:***@localhost:5432/postgres
1 rows affected.
```

Success Count	Failure Count
100	1

We combined the SELECT statement with multiple filter functions and like operators to filter counts which are needed for us

Boosters Carried Maximum Payload

- We used a subquery to select Booster Version with maximum payload mass

```
%%sql
select "Booster_Version" from ibm.spacex
where "PAYLOAD_MASS__KG_" = (select max("PAYLOAD_MASS__KG_") from ibm.spacex)
```

```
 * postgresql://postgres:***@localhost:5432/postgres
12 rows affected.
```

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

- Using simple mathematical operators is quite handy when working with dates.

```
%%sql
select "Booster_Version", "Launch_Site" from ibm.spacex
where "Date" >= '2015-01-01'
    and "Date" <= '2015-12-31'
    and "Landing _Outcome" like '%ail%'
```

```
 * postgresql://postgres:***@localhost:5432/postgres
2 rows affected.
```

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

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

- Between operator is also useful in these situations. We also used group by for aggregation and order by to show our results in descending order.

```
%%sql
select "Landing _Outcome", count("Landing _Outcome") as "count" from ibm.spacex
where "Date" between '2010-06-04' and '2017-03-20'
group by "Landing _Outcome"
order by "count" desc
```

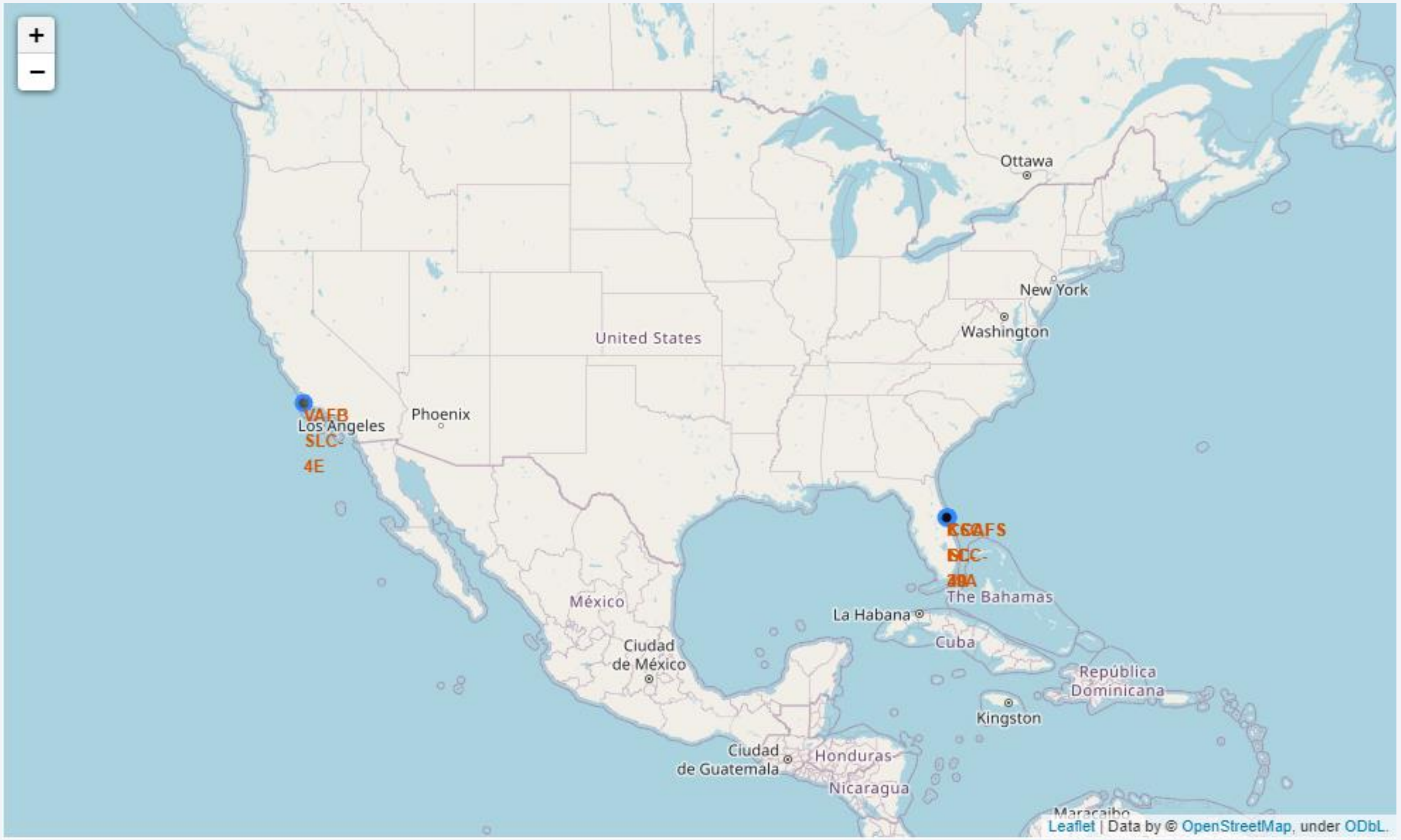
```
 * postgresql://postgres:***@localhost:5432/postgres
8 rows affected.
```

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

Section 3

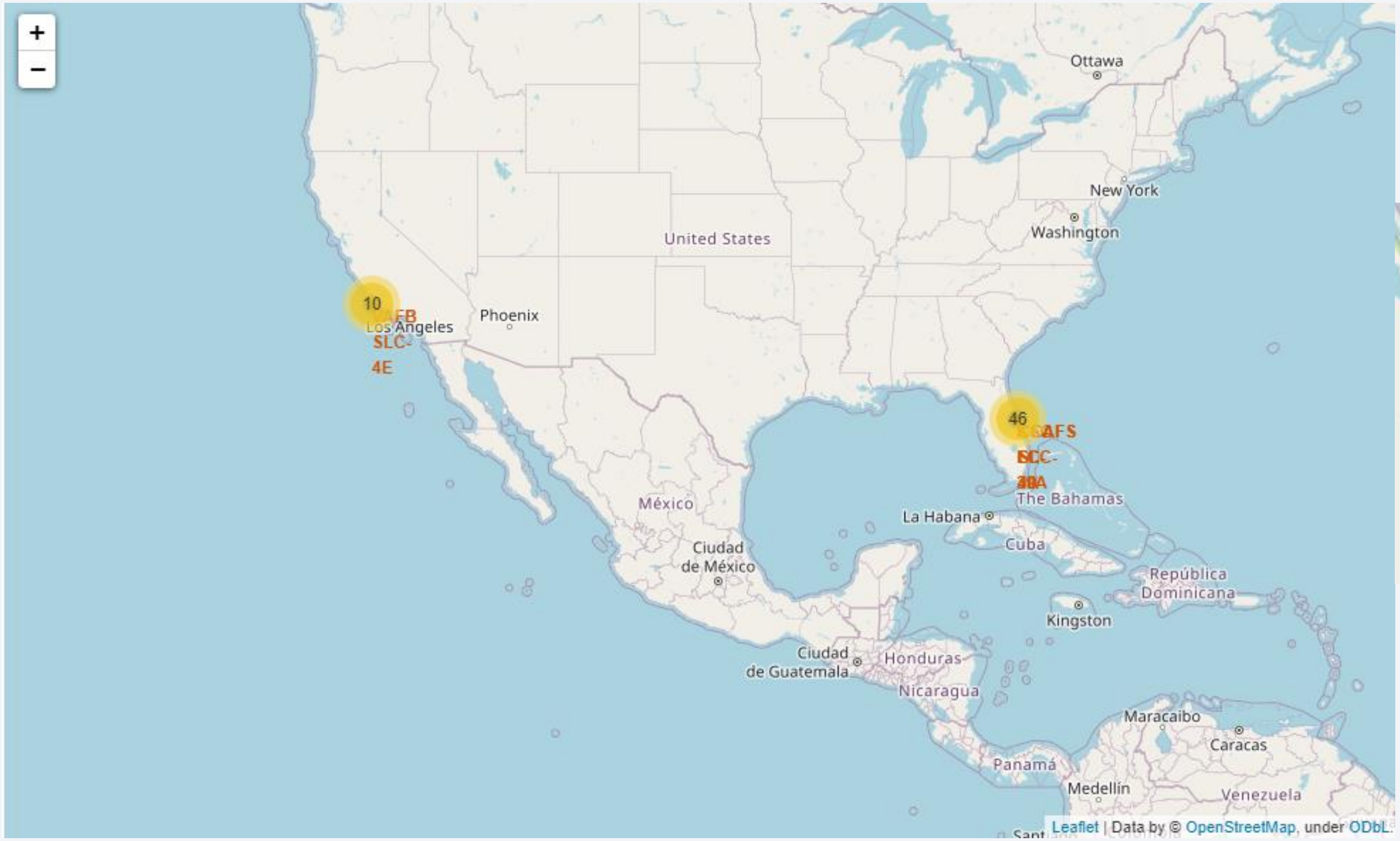
Launch Sites Proximities Analysis

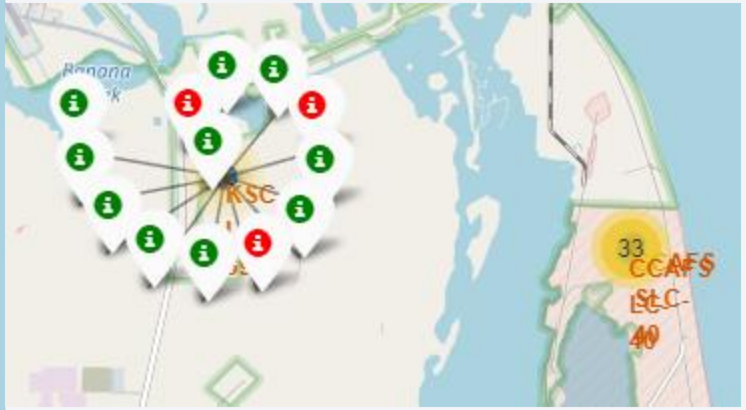
Launch Sites



Displaying all Launch Sites used used for the analysis

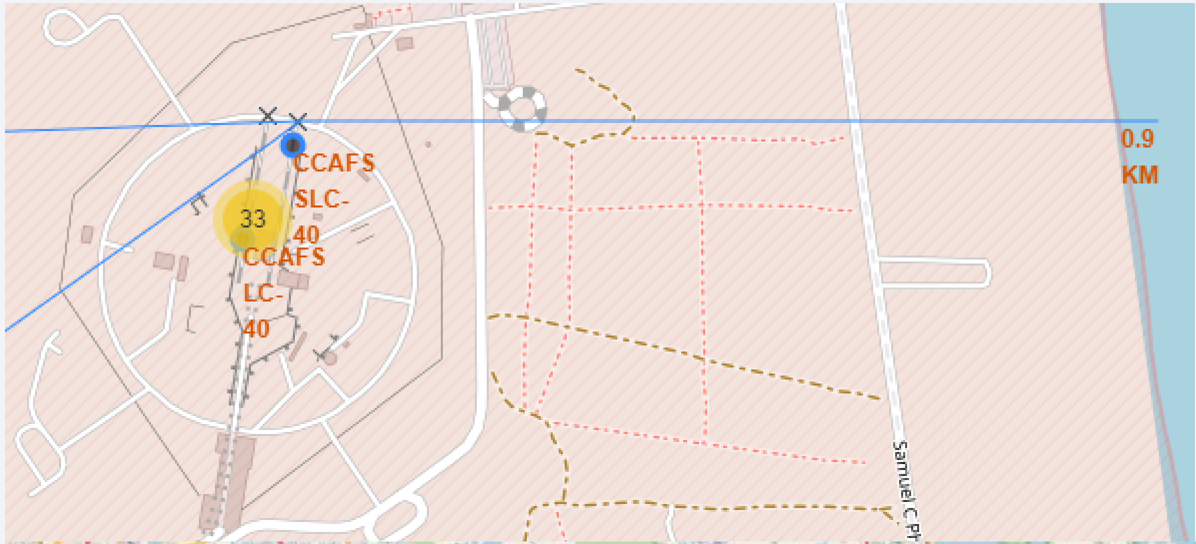
Success/failed launches

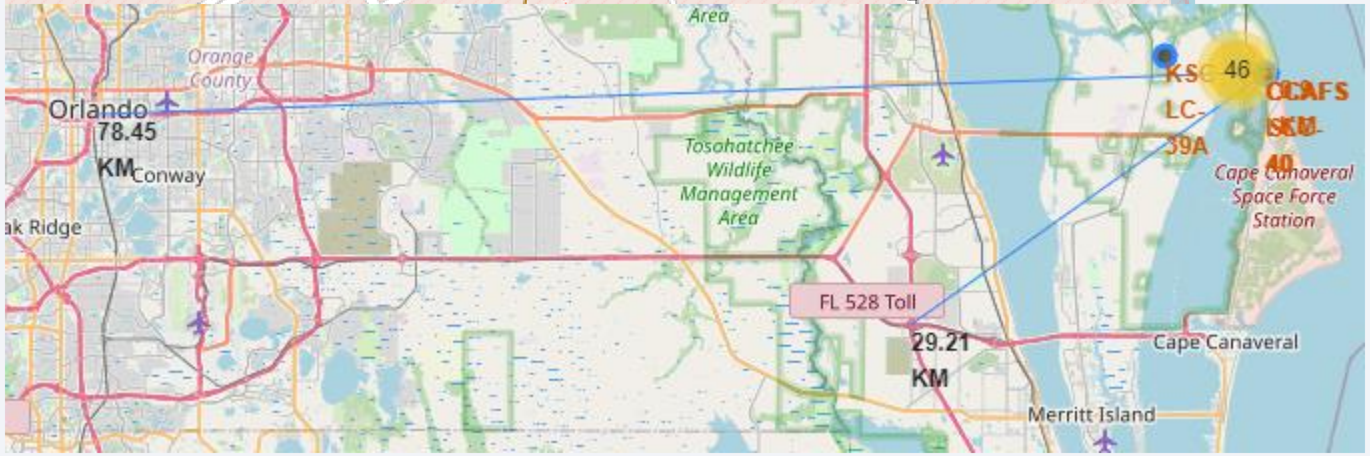




Marked successful and failed launches

<Folium Map Screenshot 3>



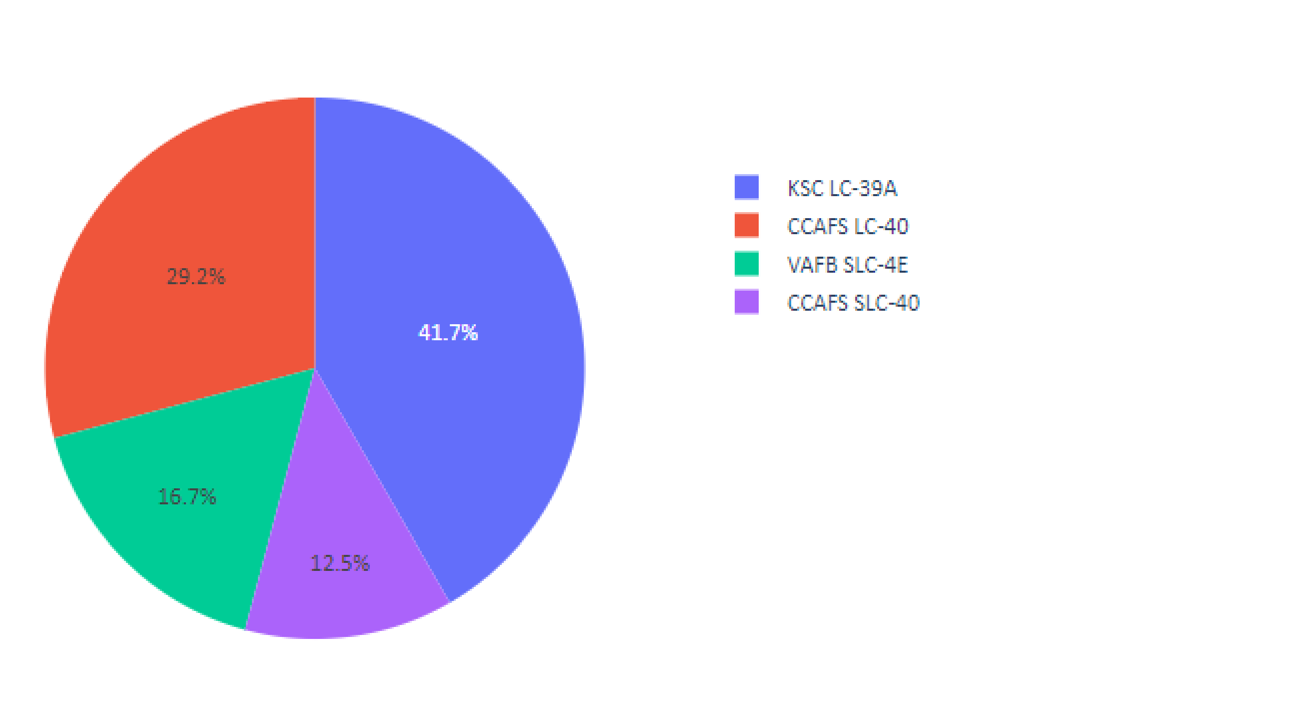


Launch sites are relaitvely far away from railways, highways and cities although they are close proximity to coastlines.

Section 4

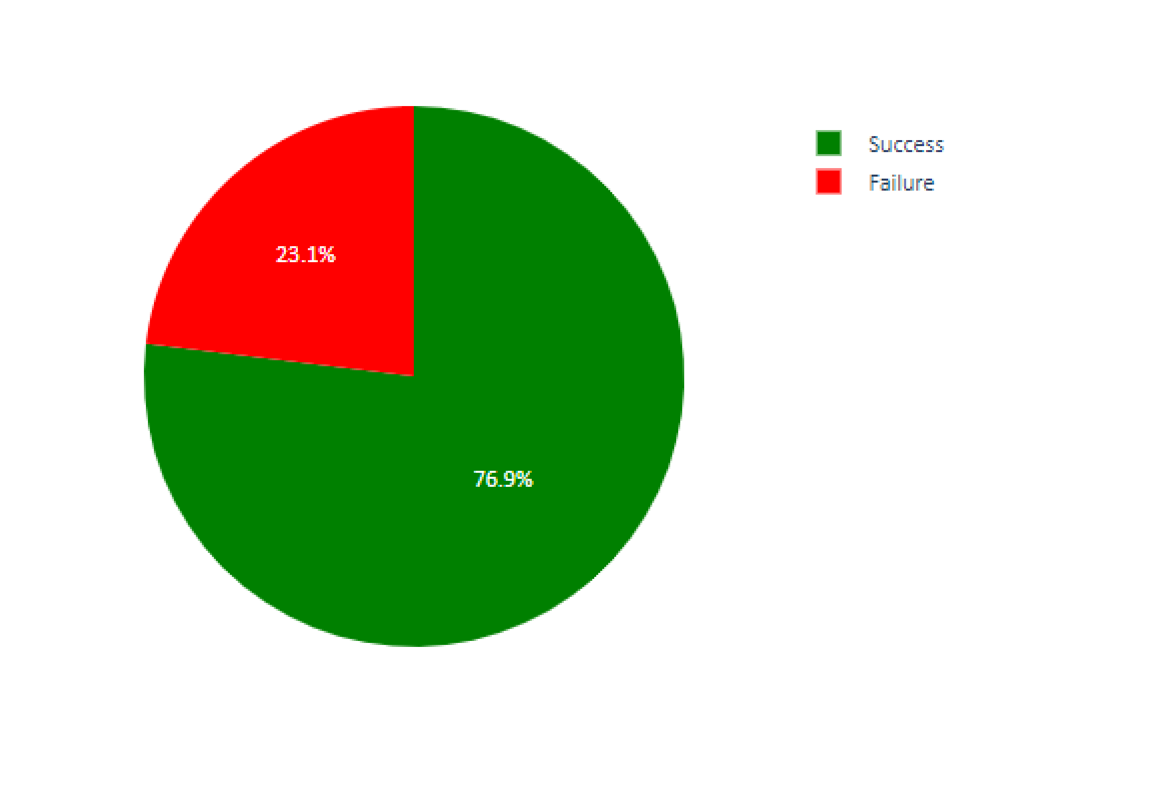
Build a Dashboard with Plotly Dash

Success rate of all sites



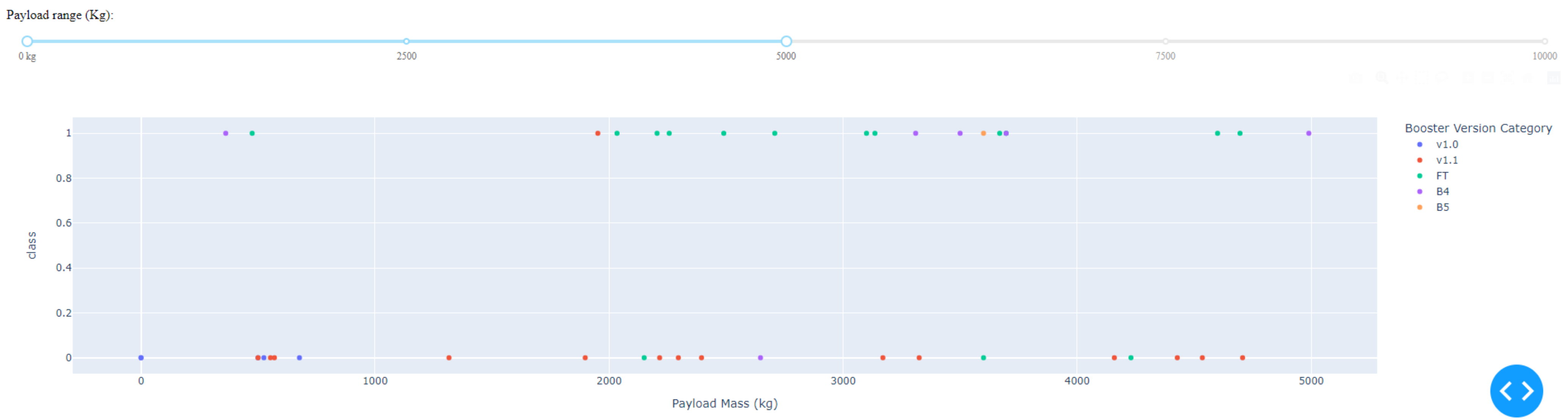
- KSC LC had the highest success rate
- CCAFS SLC had the lowest one

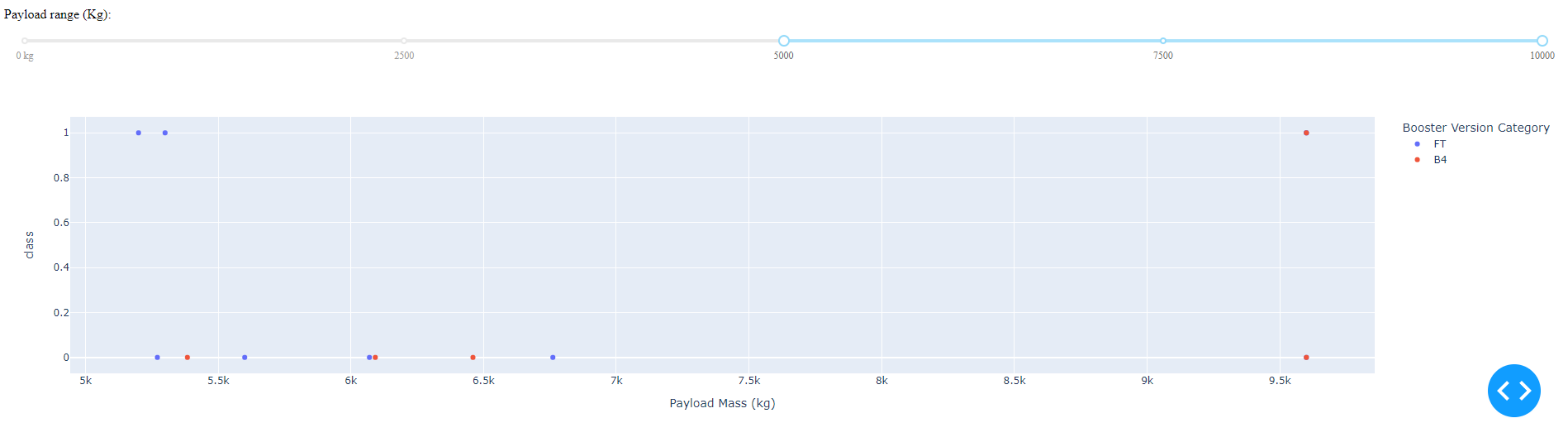
Success rate of KSC LC-39A



KSC LC-39A had the highest, almost 77% success rate

Success rate and Payload range



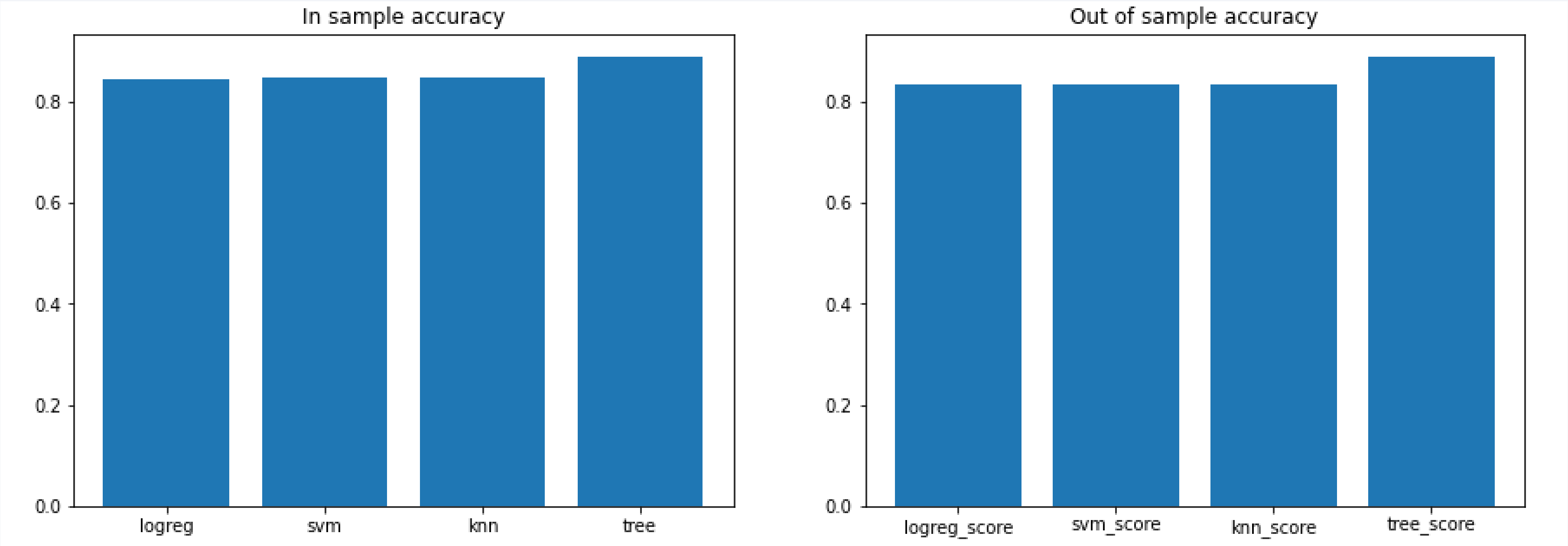


Higher payload resulted in lower succcess rate

Section 5

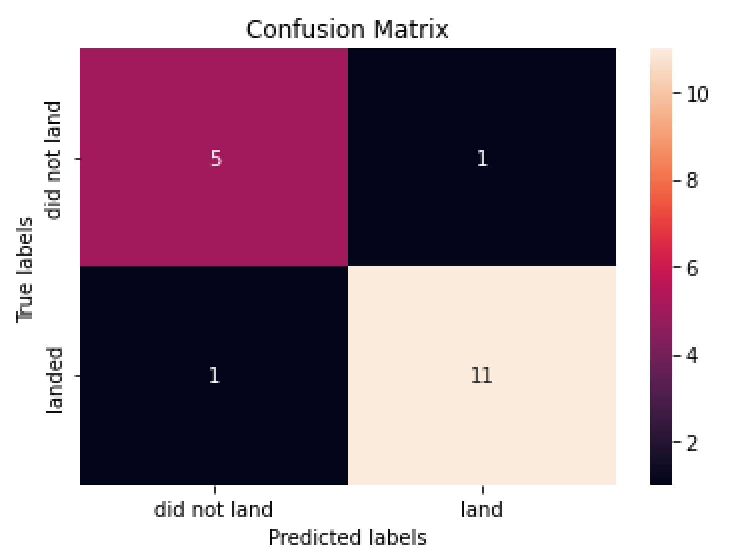
Predictive Analysis (Classification)

Classification Accuracy



Decision tree had the highest accuracy in both in sample and out of sample cases

Confusion Matrix



The confusion matrix shows how accurate was the model's prediction. In this case the Decision Tree model could distinguish between the different classes and had only 2 incorrect predictions (black boxes) 1 false positive and 1 false negative.

Conclusions

- Heavy payload positively correlates with success rate.
- KSC LC had the highest success rate
- The sucess rate since 2013 kept increasing till 202
- In this particular case Decision Tree Clasifier was the best predictive model

Thank you!