IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Marcos Antonio Teixeira

08/08/2022

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- Project background and context
- Problems you want to find answers

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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

Several datasets were used to collect rocket, launchpads, payloads and cores data from api.spacex.com/v4 website.

1. Collecting the data with API call
2. Converting to a dataframe using JSON
3. Preprocessing the data
4. Filtering data to retrieve Falcon 9 launches

Data Collection – SpaceX API

1. Collecting the data with API call

```
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

2. Converting to a dataframe using JSON

```
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

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

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bl

3. Preprocessing the data

```
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 multi
ple payloads in a single rocket.
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)]
```

4. Filtering data to retrieve Falcon 9 launches

```
getBoosterVersion(data)
```

the list has now been update

```
BoosterVersion[0:5]
```

```
['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

[Link to Github]

Data Collection - Scraping

1. Creating the BeautifulSoup

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html5lib')
```

2. Getting 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)
```

3. Creating launch dict

```
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']=[]
```

4. Converting to a dataframe

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

Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time

[Link to Github]()

Data Wrangling

[Link to Github repo]

- Loading the Dataset

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

- Identify and calculate the percentage of the missing values in each attribute

```
df.isnull().sum()/df.count()*100
```

- Identify which columns are numerical and categorical:

```
df.dtypes
```

- Calculate the number of launches on each site

```
df['LaunchSite'].value_counts()
```

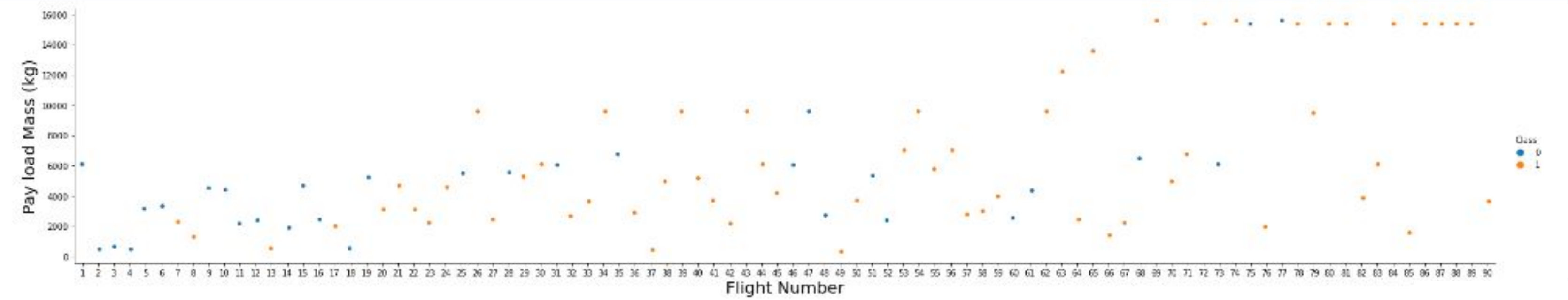
- Calculate the number and occurrence of each orbit

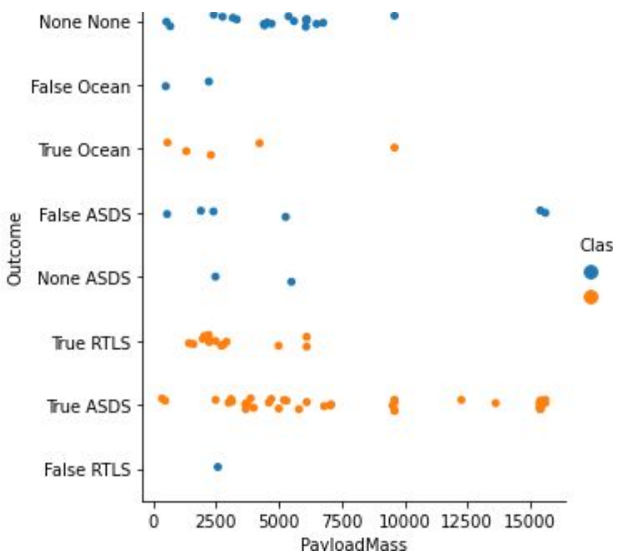
```
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

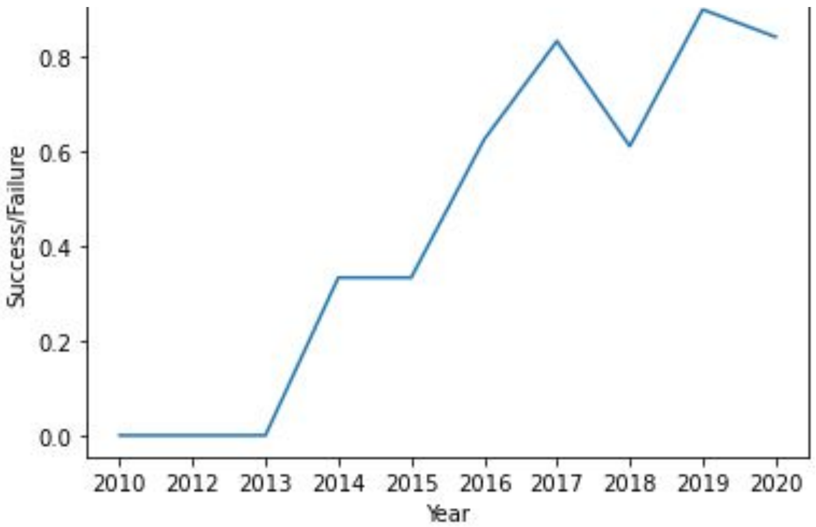
EDA with Data Visualization

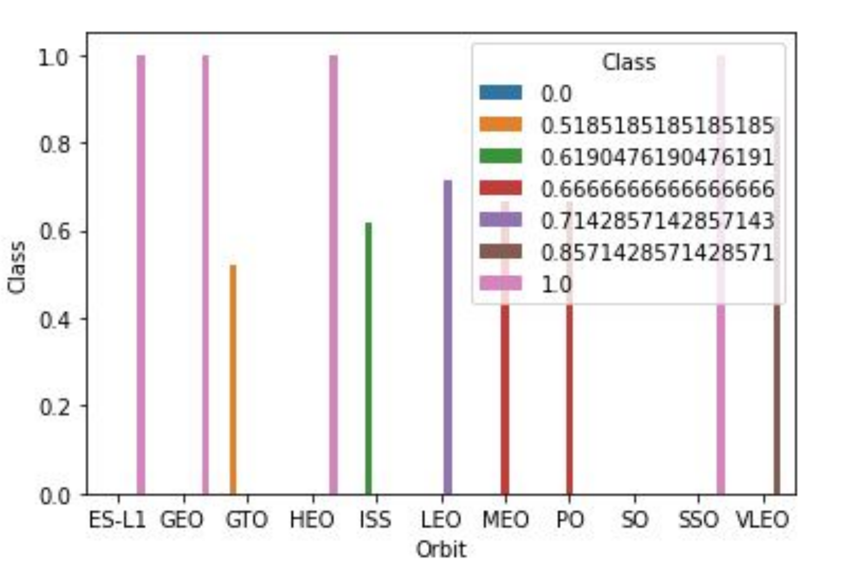
Link to Github repo

- Categorical plot between Flight Number and Payload Mass (Kg)
- Scatter plot between Orbit and Flight Number
- Bar chart between Orbit and Success rate of each orbit
- Line chart between Year and Success rate









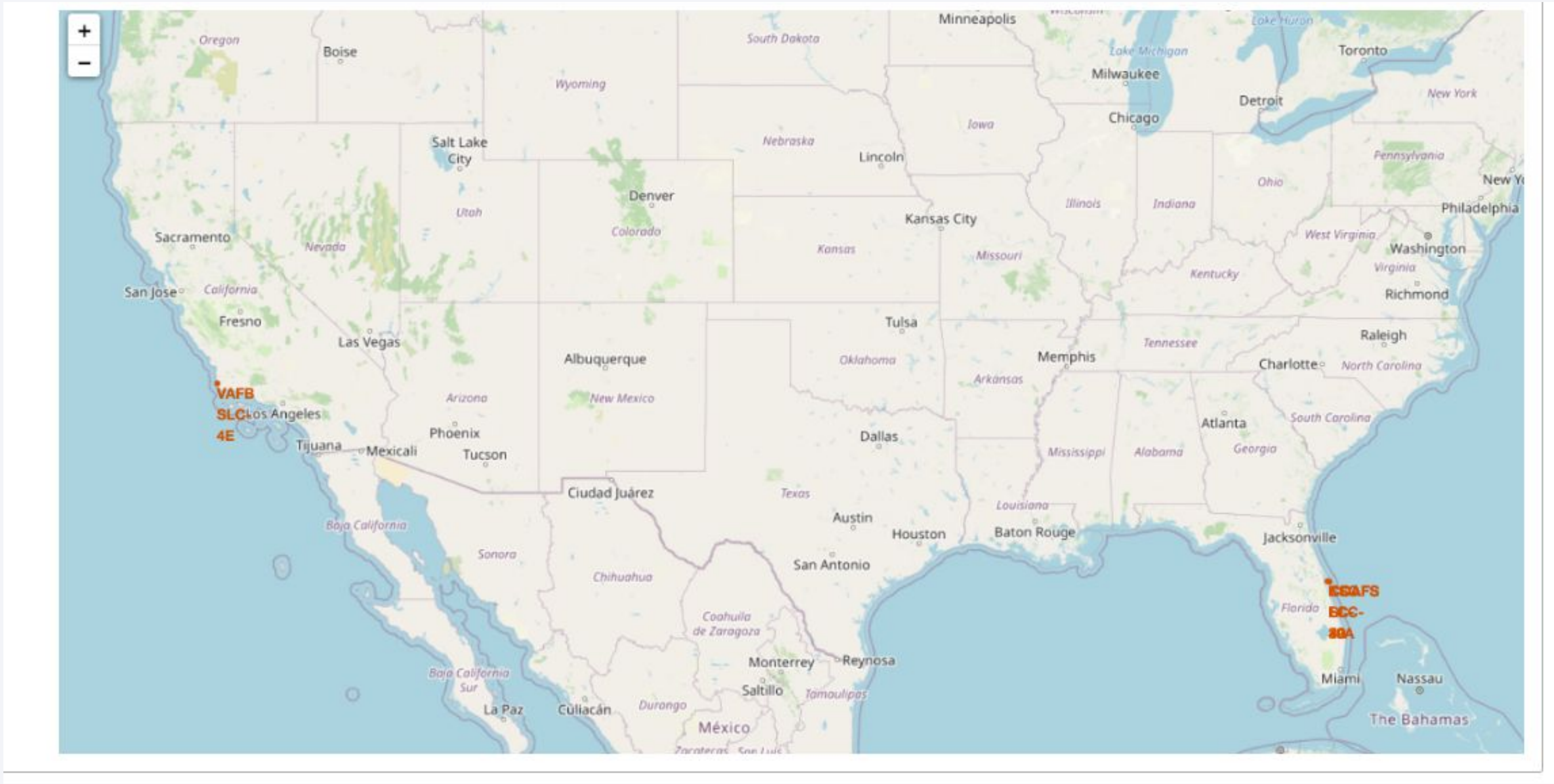
EDA with SQL

- SQL queries performed include:
-
- Displaying the names of the unique launch sites in the space mission
-
- Displaying 5 records where launch sites beginwith the string 'KSC'
-
- Displaying the total payload mass carried by boosters launched by NASA(CRS)

https://github.com/Maxtex20/ibm_final_assigment/blob/master/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

[Link to Github repo]



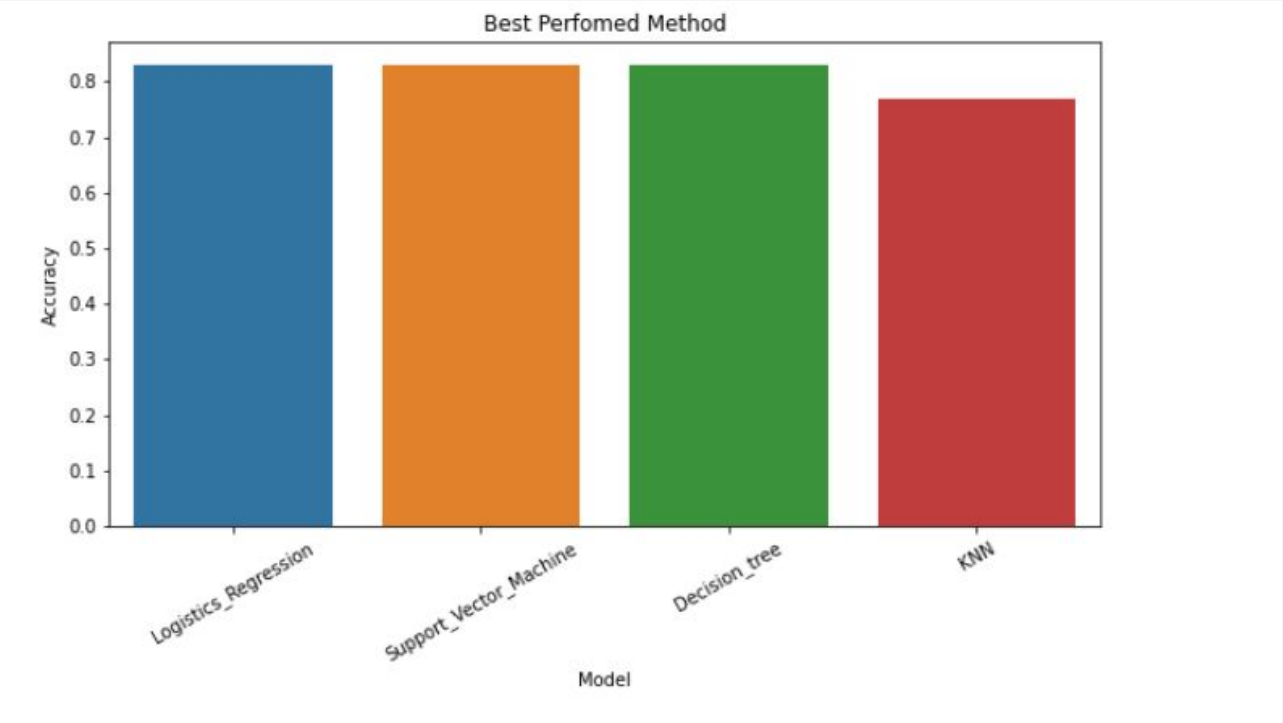
Build a Dashboard with Plotly Dash

Link to Github repo

- Dash and html components were used as they are the most important thing and almost everything depends on them, such as graphs, tables, dropdown and so on.
- To simplify things, I used Pandas to create a dataframe.
- To plot the graphs I used Plotty.
- Scatter and pie chart were used as well.
- Dropdown was used to visualize launch sites.

Predictive Analysis (Classification)

- Logistic Regression, SVM and Decision Tree were the best models



https://github.com/Maxtex20/ibm_final_assigment/blob/master/Machine%20Learning%20Prediction.ipynb

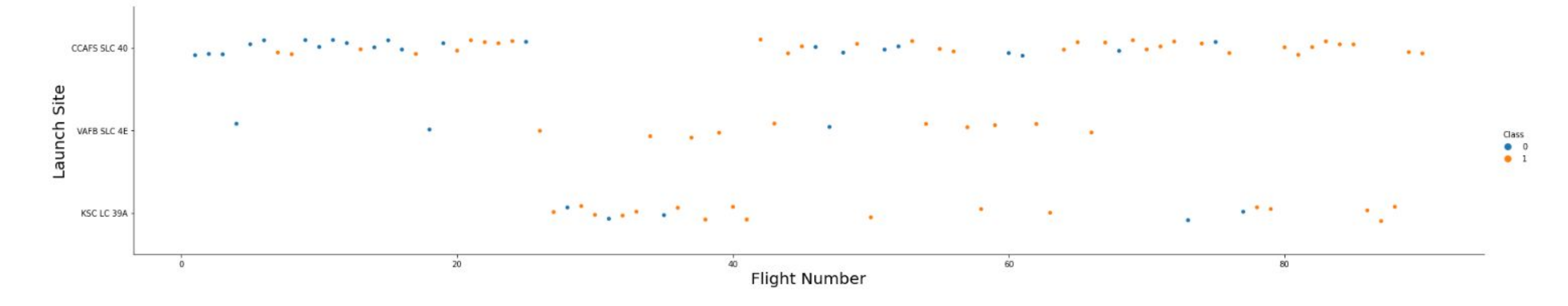
Results

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

Section 2

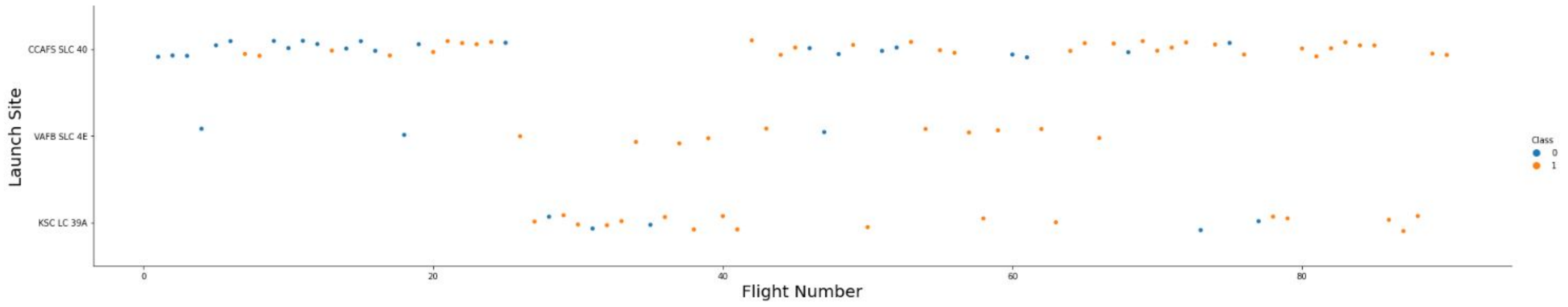
Insights drawn from EDA

Flight Number vs. Launch Site



- As flight numbers were increasing, the success rate also increased as well.

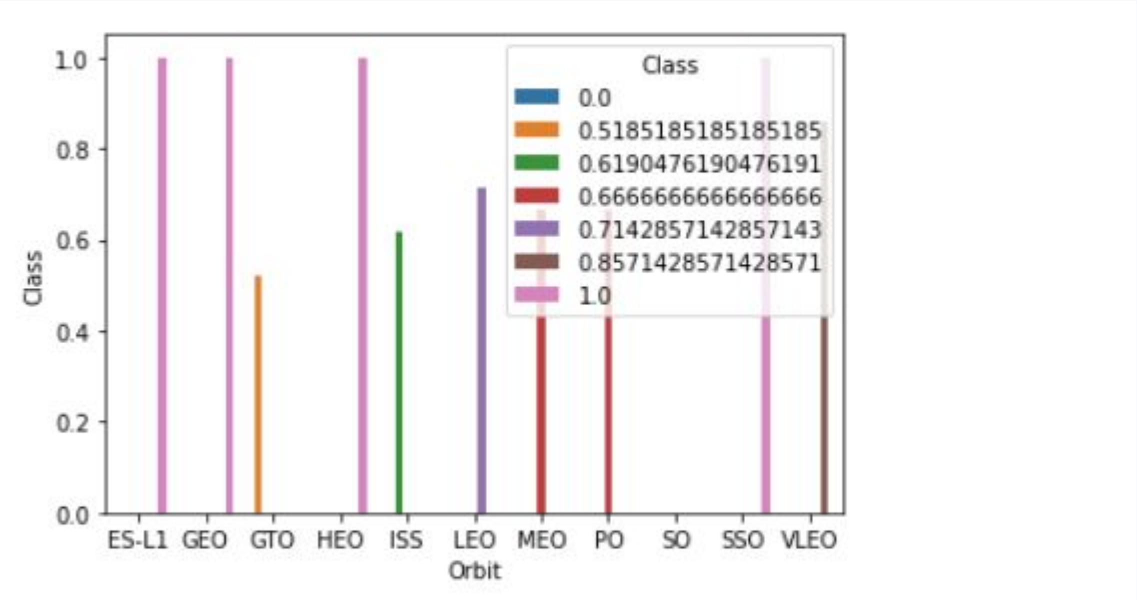
Payload vs. Launch Site



- Success rate were increased after increasing the Payload Mass in the launch sites.

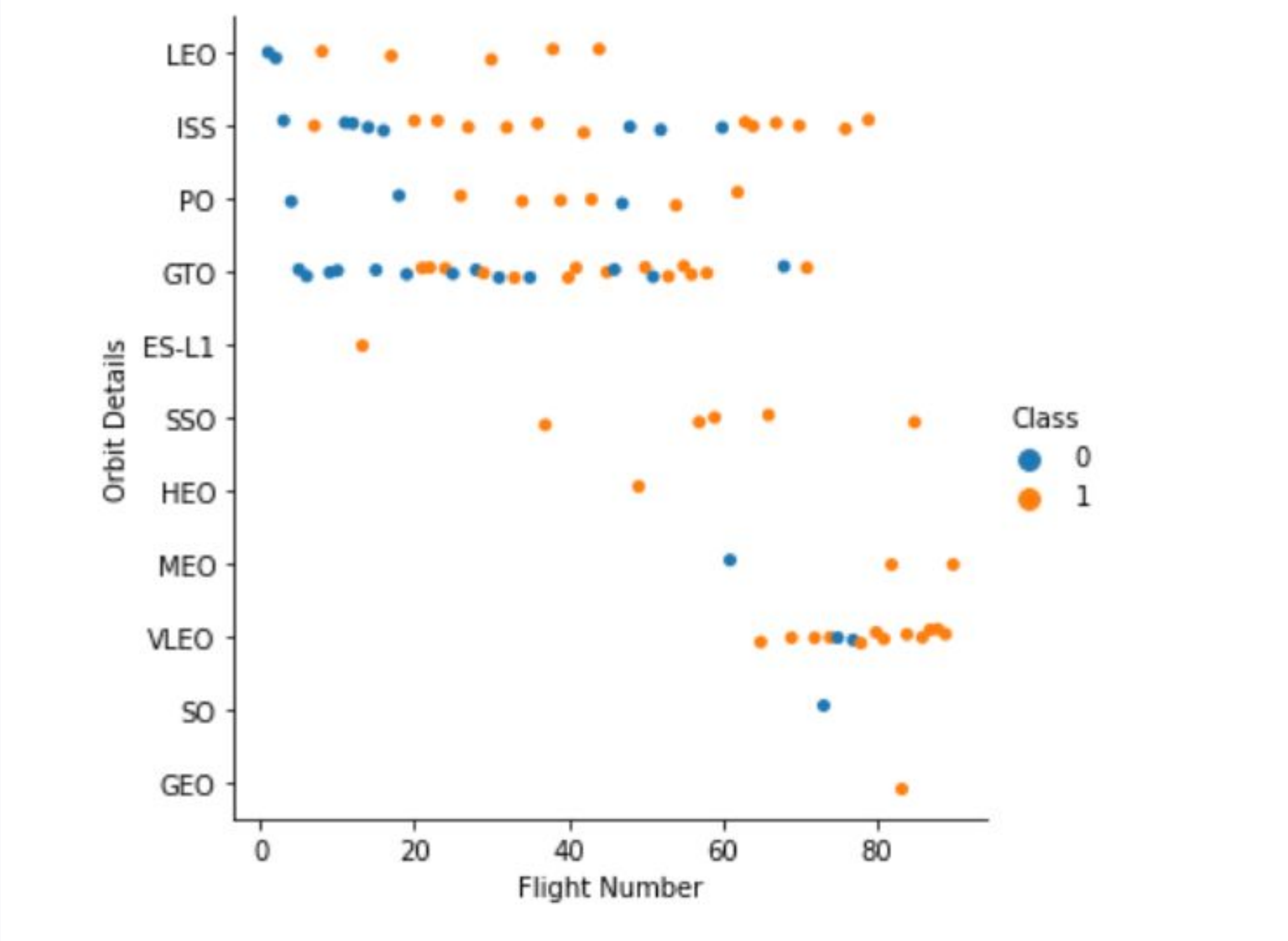
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, and SSO have a success rate of 100%.
- SO has a success rate of 0%.
-



Flight Number vs. Orbit Type

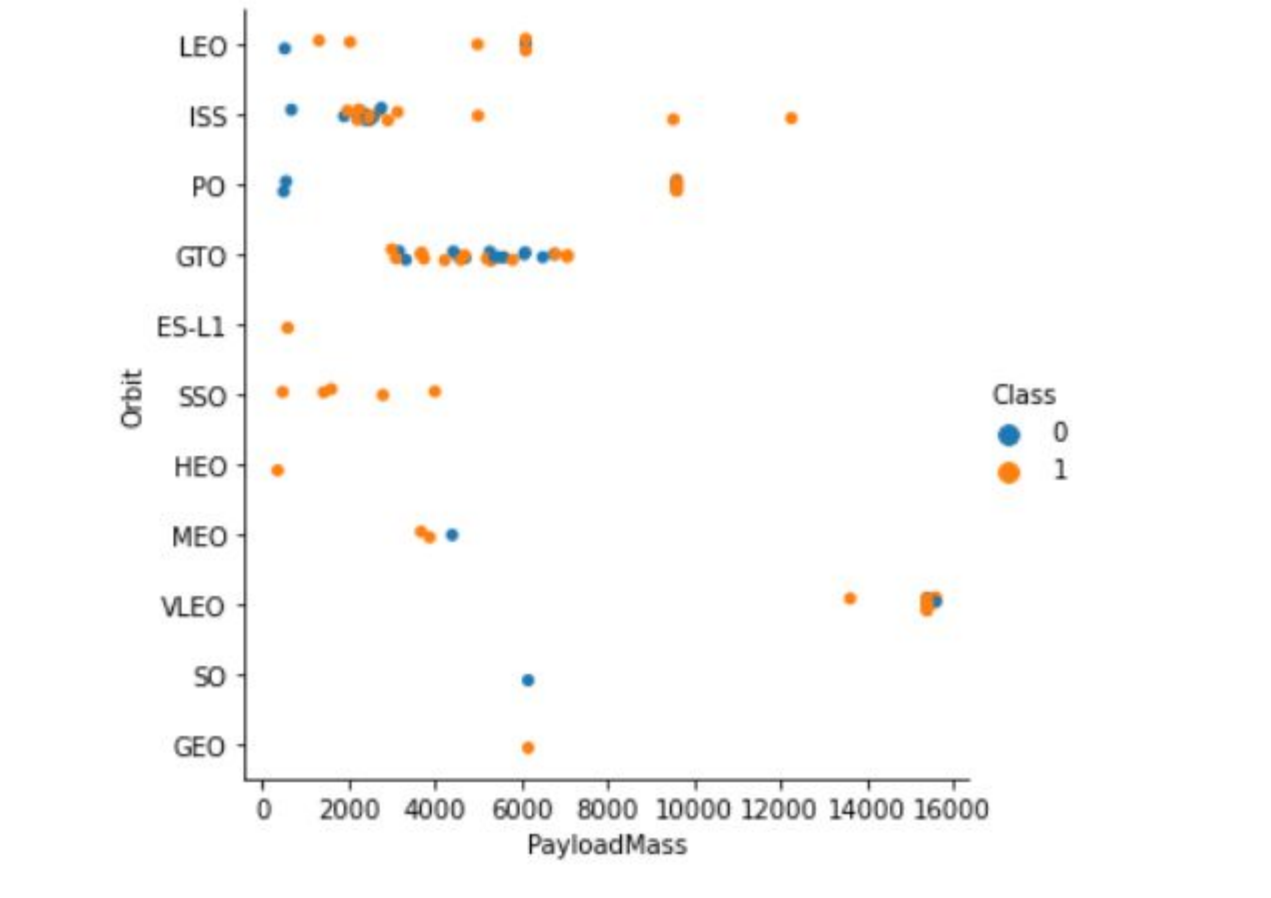
- LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



Payload vs. Orbit Type

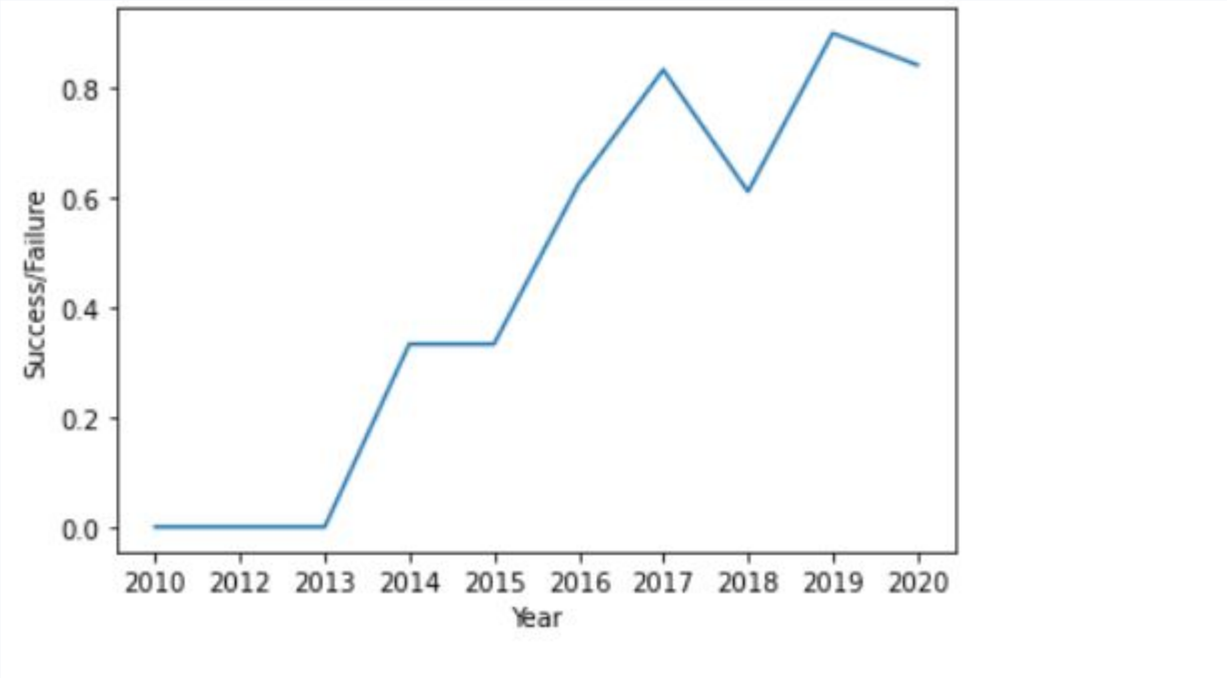
- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

- 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

- We can observe that the success rate since 2013 kept increasing till 2020.



All Launch Site Names

- Unique launch sites in the space mission

```
%sql select distinct LAUNCH_SITE from SPACEX;
```

```
 * ibm_db_sa://spw36786:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

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

Launch Site Names Begin with 'CCA'

```
%sql SELECT LAUNCH_SITE from SPACEX where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://spw36786:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

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

```
%sql select SUM(PAYLOAD_MASS__KG_) from SPACEX
```

```
 * ibm_db_sa://spw36786:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

1
619967

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

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

```
%sql select AVG(PAYLOAD_MASS__KG_) from SPACEX where BOOSTER_VERSION = 'F9 v1.1';
```

```
 * ibm_db_sa://spw36786:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

1
2928

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

```
1 %sql SELECT MIN(DATE) FROM SPACEX WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

```
 * ibm_db_sa://spw36786:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

1
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

- Payload mass data taken between 4000 and 6000 only.

```
1  %sql select distinct BOOSTER_VERSION from SPACEX where PAYLOAD_MASS__KG_ between 4000 and 6000 and LANDING__OUTCOME = 'Succe
```

```
 * ibm_db_sa://spw36786:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

- 99 successful missions and 1 failed mission.

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

```
 * ibm_db_sa://spw36786:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

missionoutcomes
1
99
1

Boosters Carried Maximum Payload

```
%sql select booster_version, max(payload_mass__kg_) as"maximum payload mass" from (select booster_version , payload
```

booster_version	maximum payload mass
F9 B4 B1039.2	2647
F9 B4 B1040.2	5384
F9 B4 B1041.2	9600
F9 B4 B1043.2	6460
F9 B4 B1039.1	3310
F9 B4 B1040.1	4990
F9 B4 B1041.1	9600
F9 B4 B1042.1	3500
F9 B4 B1043.1	5000
F9 B4 B1044	6092
F9 B4 B1045.1	362
F9 B4 B1045.2	2697
F9 B5 B1046.1	3600
F9 B5 B1046.2	5800
F9 B5 B1046.3	4000
F9 B5 B1046.4	12050

2015 Launch Records

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)

```
%sql select date, booster_version, launch_site, landing__outcome from spacex where landing__outcome ='Failure (dron
```

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

```
%sql select count(*) as "count of landing outcomes", landing__outcome from spacex where date between '2010-06-04' a
```

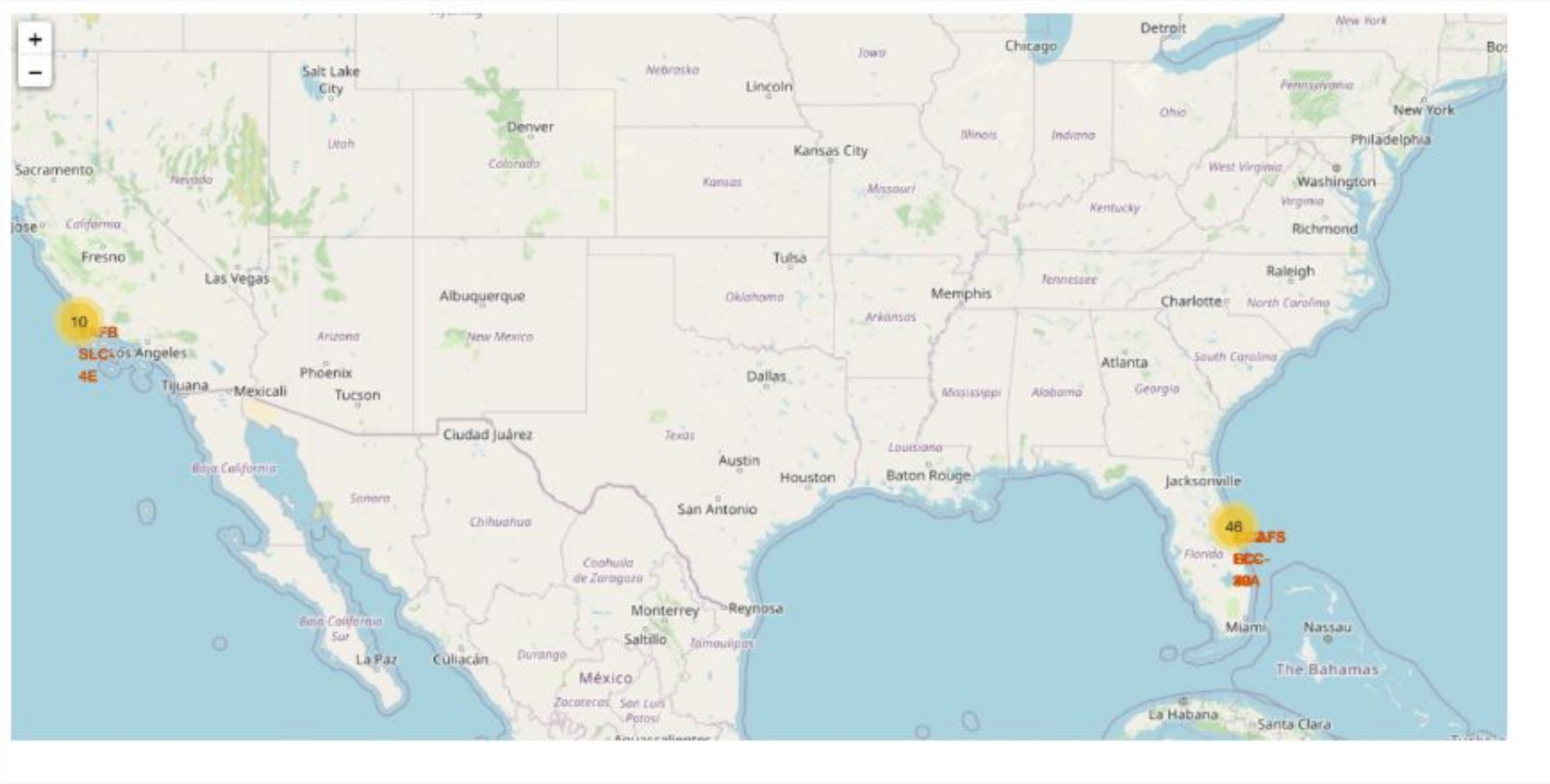
count of landing outcomes	landing__outcome
10	No attempt
5	Failure (drone ship)
5	Success (drone ship)
3	Controlled (ocean)
3	Success (ground pad)
2	Failure (parachute)
2	Uncontrolled (ocean)
1	Precluded (drone ship)

Section 3

Launch Sites Proximities Analysis

Folium Map Screenshot

- Lauchs sites are in Florida and California, USA.



Launch Sites Outcome

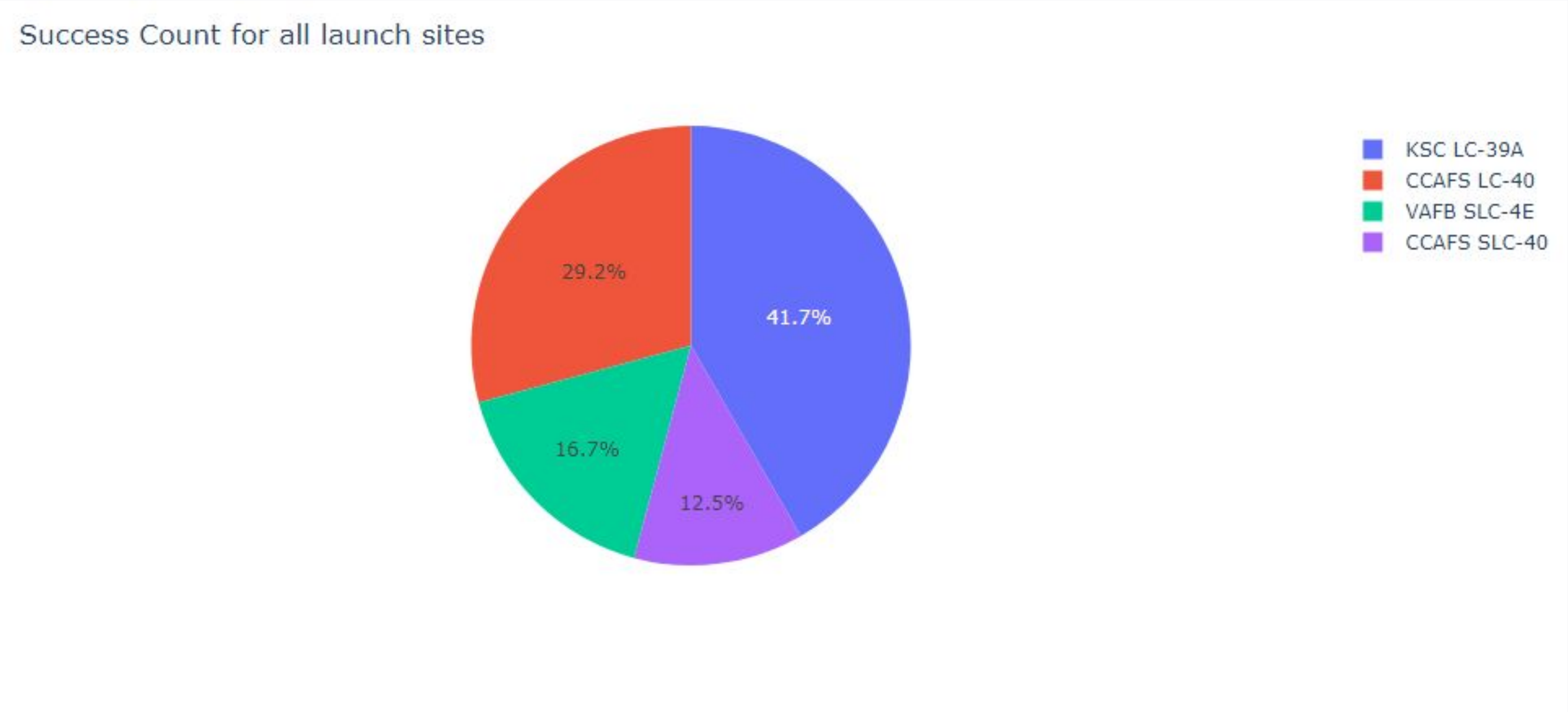
- Green means successful
- Red means Failure

Launch Site Vicinity

Section 4

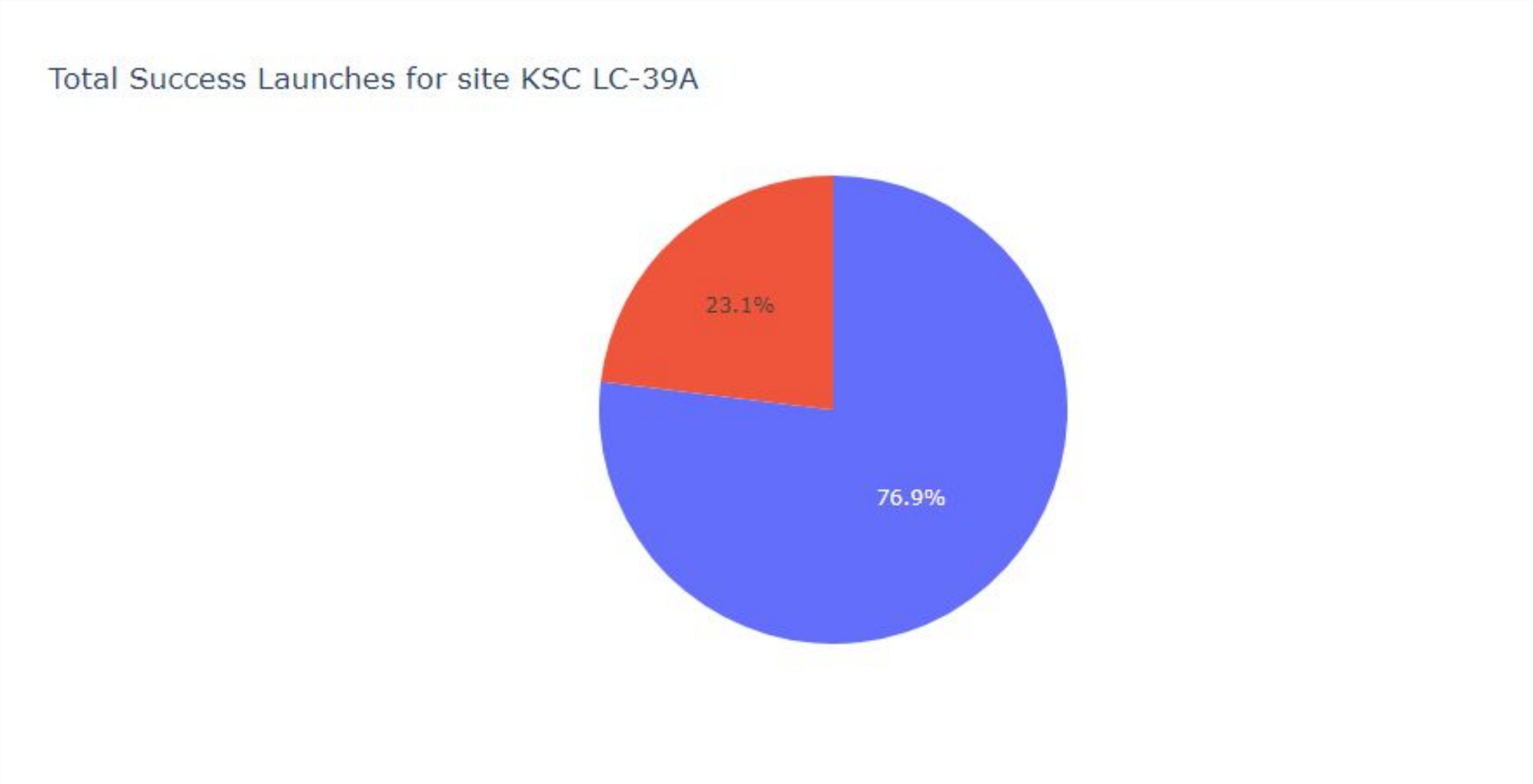
Build a Dashboard with Plotly Dash

Total success launches by all sites

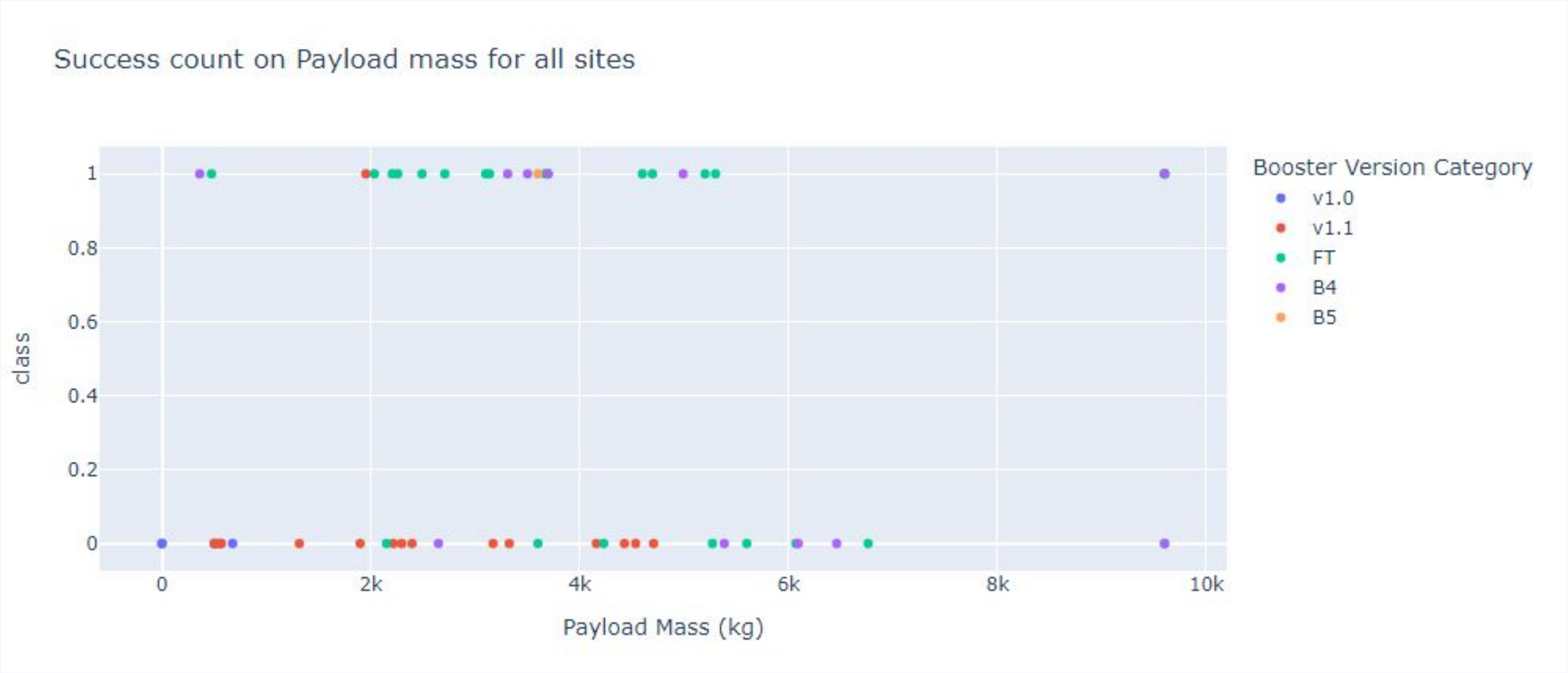


Sucess Rate by site

KSC LC-39A achieved a success rate of 76.95%



Payload vs Launch outcome

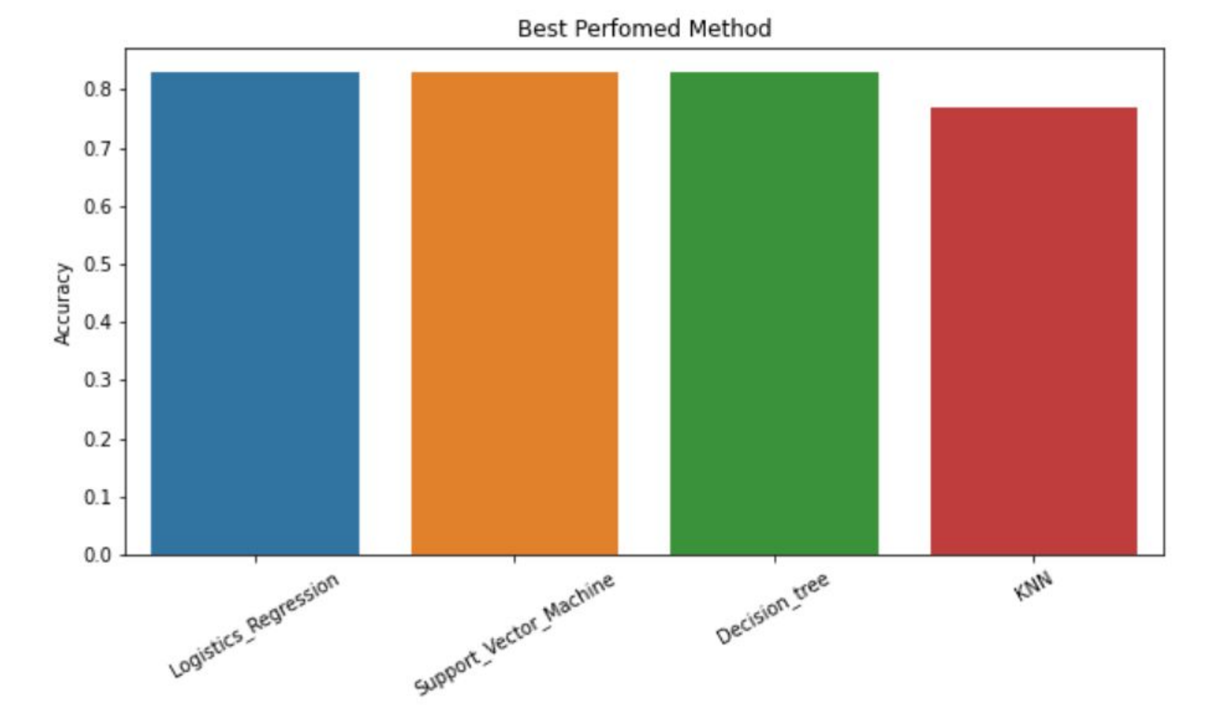


Section 5

Predictive Analysis (Classification)

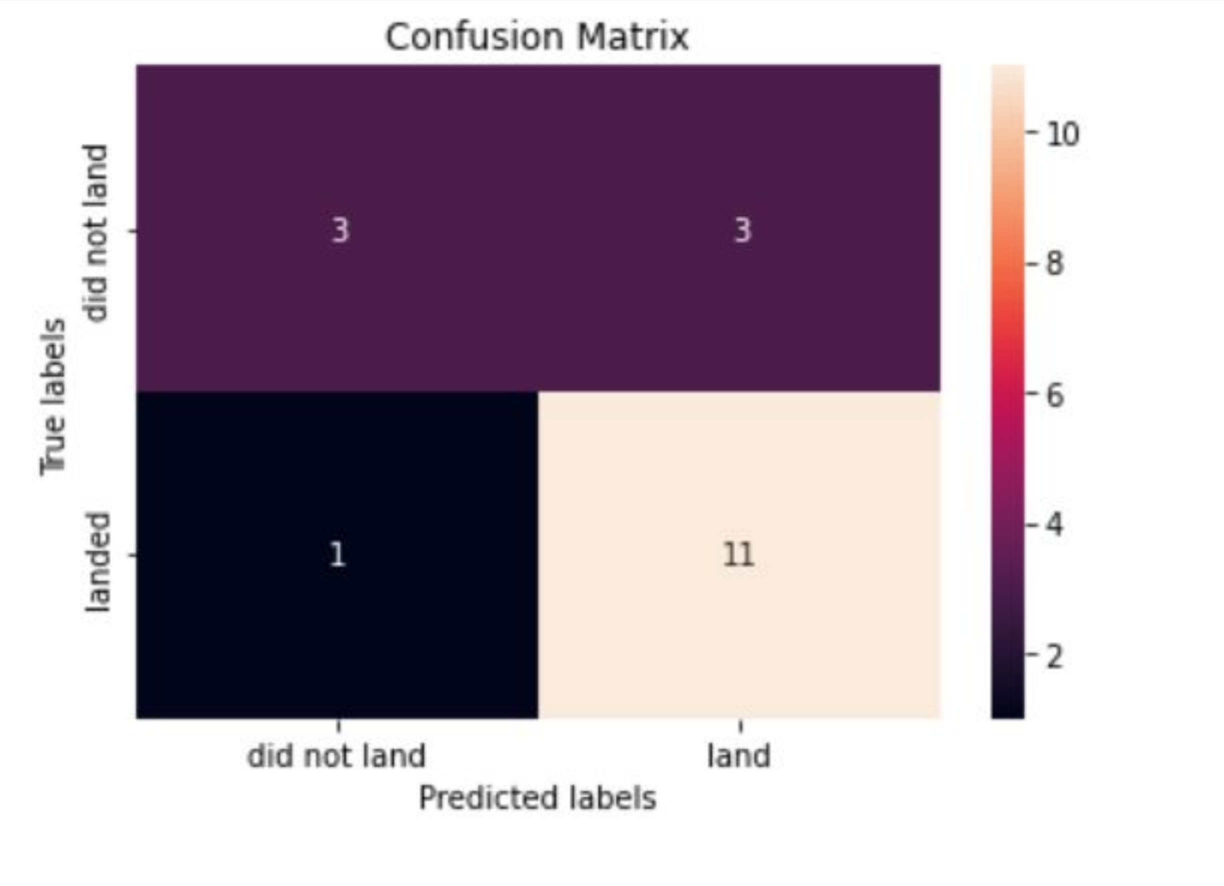
Classification Accuracy

- Logistic Regression, SVM and Decision Tree were the best models with accuracy around 0.83



Confusion Matrix

- Logistic Regression, SV and Decision Tree got similar confusion matrizes



Conclusions

- Launch site with highest score was KSC LC-39A
- The payload of 0 to 5000 Kg was more diverse than 6000 to 10000 kg.
- Logistic Regression, SVM and Decision Tree were very similar with accuracy around 0.83

Appendix

Thank you!