IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

IBM Data Science Capstone Project

Florin Neagu

27th of April 2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

- Summary of methodologies

A. Data Collection with RestAPI and Web Scraping
B. Exploratory Data Analysis with Data Visualization
C. Exploratory Data Analysis with SQL
D. Interactive Maps with Folium
E. Interactive Dashboards with Plotly Dash
F. Predictive Analysis

- Summary of all results

1. Exploratory Data Analysis results
2. Interactive Maps and Dashboards results
3. Predictive Analysis results

Introduction

- Project background and context

The aim of this project is to predict if the Falcon 9 first stage will successfully land.

SpaceX advertise on its website that the Falcon 9 rocket launch cost 62 million dollars.

Other providers cost upward of 165 million dollars each.

The price difference is explained by the fact that SpaceX can reuse the first stage.

By determining if the first stage will land, we can determine the cost of a launch.

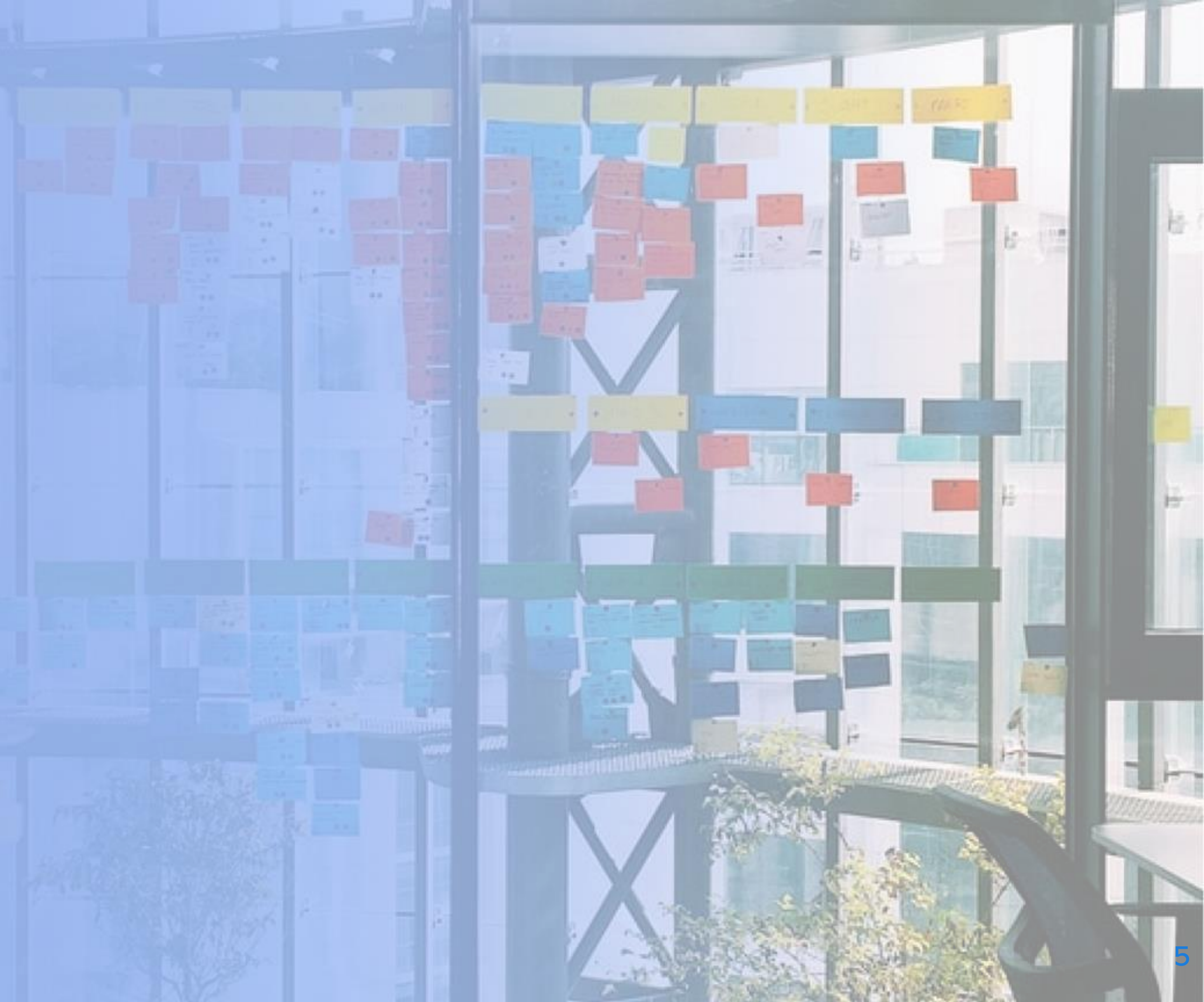
This information is interesting for another company if it wants to compete with SpaceX for a rocket launch.

- Problems you want to find answers

1. What are the main characteristics of a successful or failed landing ?
2. What are the effects of each relationship of the rocket variables on the success or failure of a landing ?
3. What are the conditions which will allow SpaceX to achieve the best landing success rate ?

Section 1

Methodology



Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web Scraping from Wikipedia
- Perform data wrangling
  - Wrangling data using API
  - Sampling [or Filtering] Data
  - Dealing with Missing Values
  - One Hot Encoding for classification models
- 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, tuning and evaluating classification models

Data Collection

Datasets are collected from SpaceX Rest API and Web Scraping Wikipedia using Python

- The information obtained by the API are rocket, launches, payload information.
  - The Space X Rest API URL is: https://api.spacexdata.com/v4/

SpaceX Rest API call → API returns JSON file → Make Dataframe from JSON → Clean Data and export it → dataset_part1.csv

- The information obtained by the Web Scraping of Wikipedia are launches, landing, payload information.
  - Wikipedia URL is: https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

Get HTML response from Wikipedia → Extract data with BeautifulSoup → Make Dataframe → Export Data → spacex_web_scraped.csv

Data Collection – SpaceX Rest API

1. Getting Response from API

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

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

2. Convert Response to JSON file

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

3. Transform data

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
getBoosterVersion(data)
```

4.Create dictionary with 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}
```

5. Create dataframe

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

6. Filter dataframe

```
data_falcon9=df[df['BoosterVersion']!='Falcon 1']
```

7. Export to file

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

GitHub

Data Collection - Web Scraping

1. Getting Response from HTML

```
response=requests.get(static_url).text
```

2. Create BeautifulSoup Object

```
soup=BeautifulSoup(response,"html.parser")
```

3. Find all tables

```
html_tables=soup.find_all('table')
```

4. Get column names

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
first_launch_table.find_all('th')
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
for row in first_launch_table.find_all('th'):
    name=extract_column_from_header(row)
    print(name)

# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
    if name is not None and len(name) > 0:
            column_names.append(name)
```

5. Create dictionary

```
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']=[]
```

6. Add data to keys

See notebook for the rest of code

7. Create dataframe from dictionary

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

8. Export to file

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

GitHub

Data Wrangling

Data Wrangling started already in the process of Collecting Data

1.Wrangling Data using an API

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

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
getBoosterVersion(data)
```

2.Sampling Data

Filtering the data for Falcon 9

```
data_falcon9=df[df['BoosterVersion']!='Falcon 1']
```

3.Dealing with Missing Values

Dealing with Nulls in PayloadMass Column

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

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

4. Launches no. for each site

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

5. No. and occurrence of each orbit

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

6. No. and occurrence of outcome per orbit

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

7.One Hot Encoding for classification models

Create landing outcome label from Outcome column

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for key,value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
# landing_class = 1 otherwise
    else:
        landing_class.append(1)
#Then assign it to the variable landing_class
print(landing_class)
```

8. Export to .csv file

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

GitHub

Exploratory Data Analysis with Data Visualization

Scatter Graphs

Scatter graphs show the relationship [correlation] between variables.

- Flight Number vs. Payload Mass
- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Orbit vs. Flight Number
- Payload vs. Orbit Type
- Orbit vs. Payload Mass

Bar Graph

Bar graphs show the relationship between numeric and categoric variables.

- Success rate vs. Orbit

Line Graph

Line graphs show data variables and their trends. Line graphs help to show global behavior and to make prediction for unseen data.

- Success rate vs. Year

GitHub

Exploratory Data Analysis with SQL

Performed SQL queries to gather and understand data from the dataset [dataset_part_2.csv]

- 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 successful landing outcome in ground pad was achieved.
- 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.
- List the records which will display the month names, failure landing_ouutcomes 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.

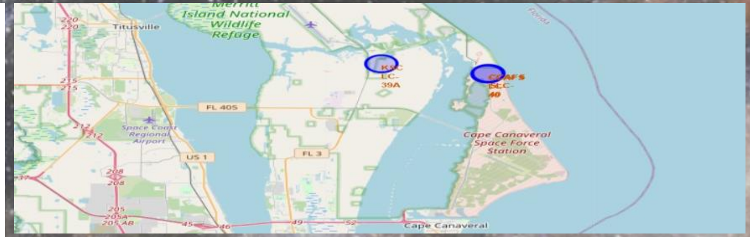
[GitHub]

Build an Interactive Map with Folium

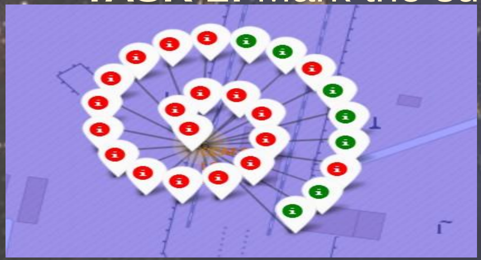
Performed tasks to make the Launch Sites Locations Analysis with Folium

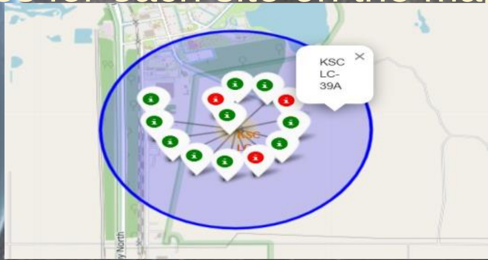
- **TASK 1:** Mark all launch sites on a map
  We used folium.Circle to add a highlighted circle area with a text label on a specific coordinate.

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

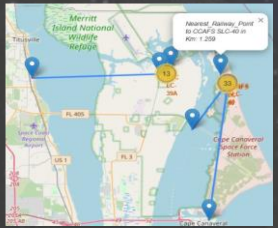


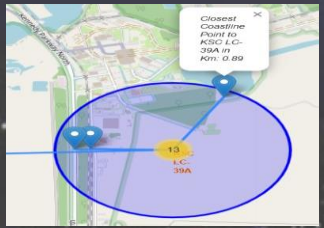
- **TASK 2:** Mark the success/failed launches for each site on the map

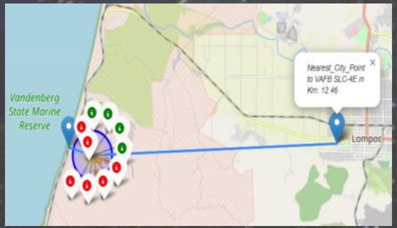




- **TASK 3:** Calculate the distances between a launch site to its proximities







[GitHub]

Build a Dashboard with Plotly Dash

The Dashboard created provides dropdown, pie chart, rangeslider and scatter plot components

- Dropdown allows a user to choose the launch site or all launch sites (dash_core_components.Dropdown).
- Pie chart shows the total success and the total failure for the launch site chosen with the dropdown component (plotly.express.pie).
- Rangeslider allows a user to select a payload mass in a fixed range (dash_core_components.RangeSlider).
- Scatter chart shows the relationship between two variables, in particular Success vs Payload Mass (plotly.express.scatter).

GitHub

Predictive Analysis [Classification]

- Data preparation
  - Load dataset [dataset_part_2.csv]
  - Normalize data
  - Split data into training and test sets
- Model preparation
  - Selection of machine learning algorithms [Logistic Regression, SVM, Decision Tree, KNN]
  - Set parameters for each algorithm to GridSearchCV
  - Training GridSearchModel models with training dataset
- Model evaluation
  - Get best hyperparameters for each type of model
  - Compute accuracy for each model with test dataset
  - Plot Confusion Matrix
- Model comparison
  - Comparison of models according to their accuracy
  - The model with the best accuracy will be chosen

GitHub

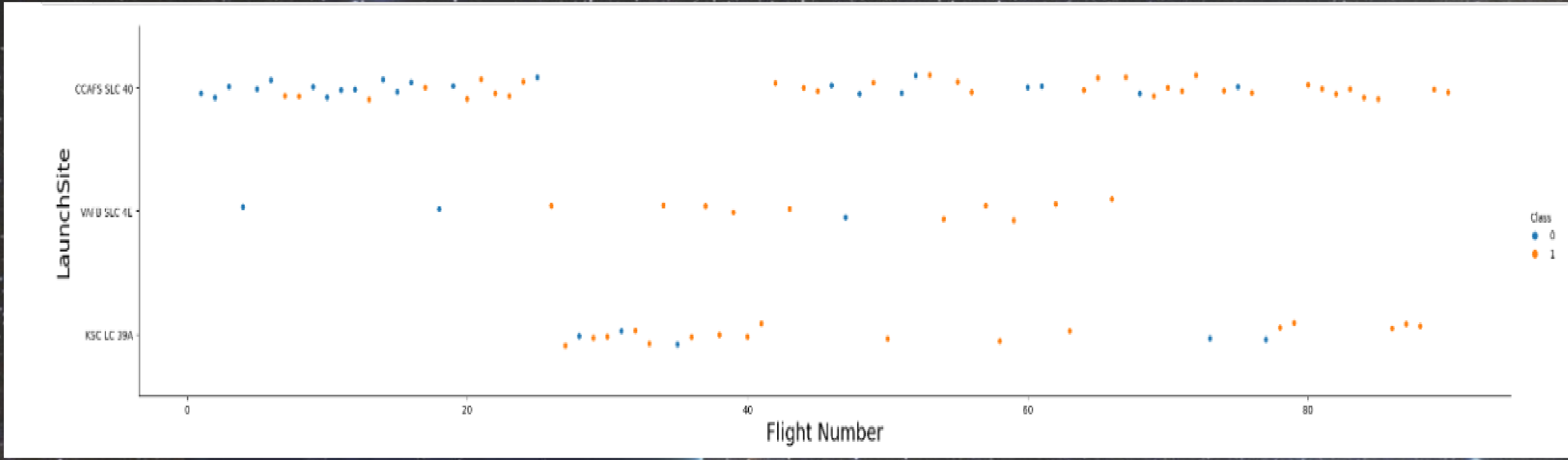
Results

- Exploratory Data Analysis [EDA] results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

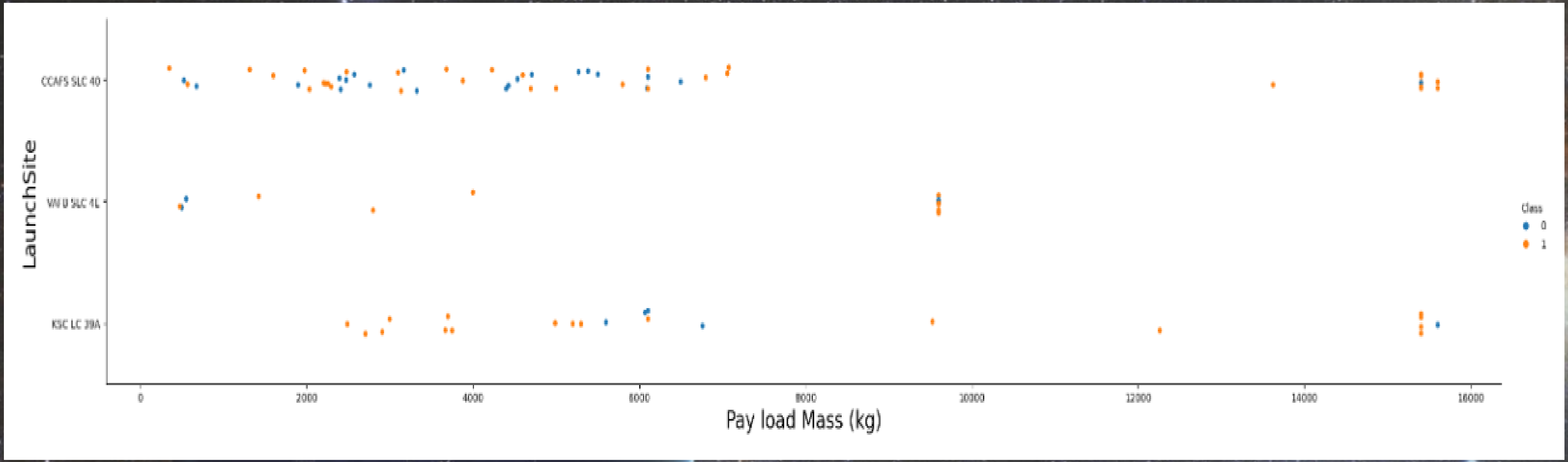
Insights drawn from EDA

Flight Number vs. Launch Site



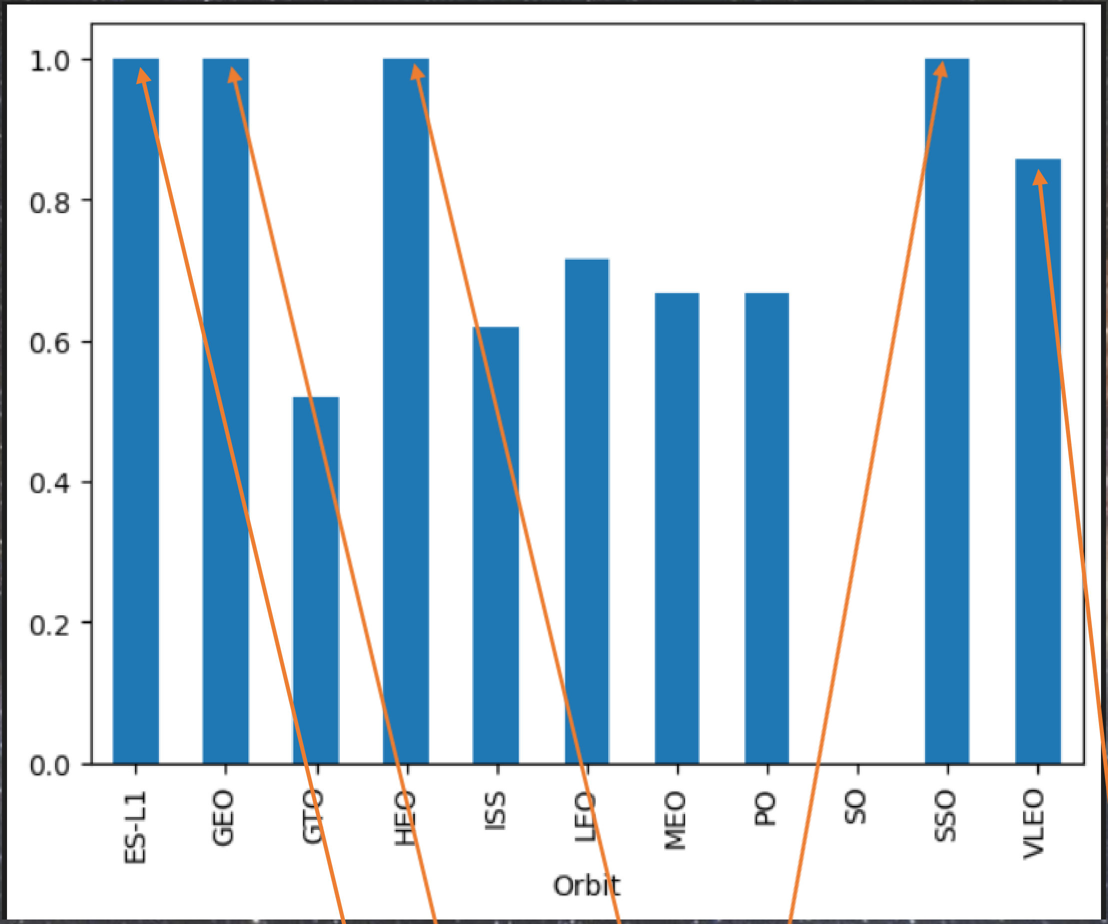
For each Launch Site with higher Flight Number [greater than 30] the success rate was increased

Payload vs. Launch Site



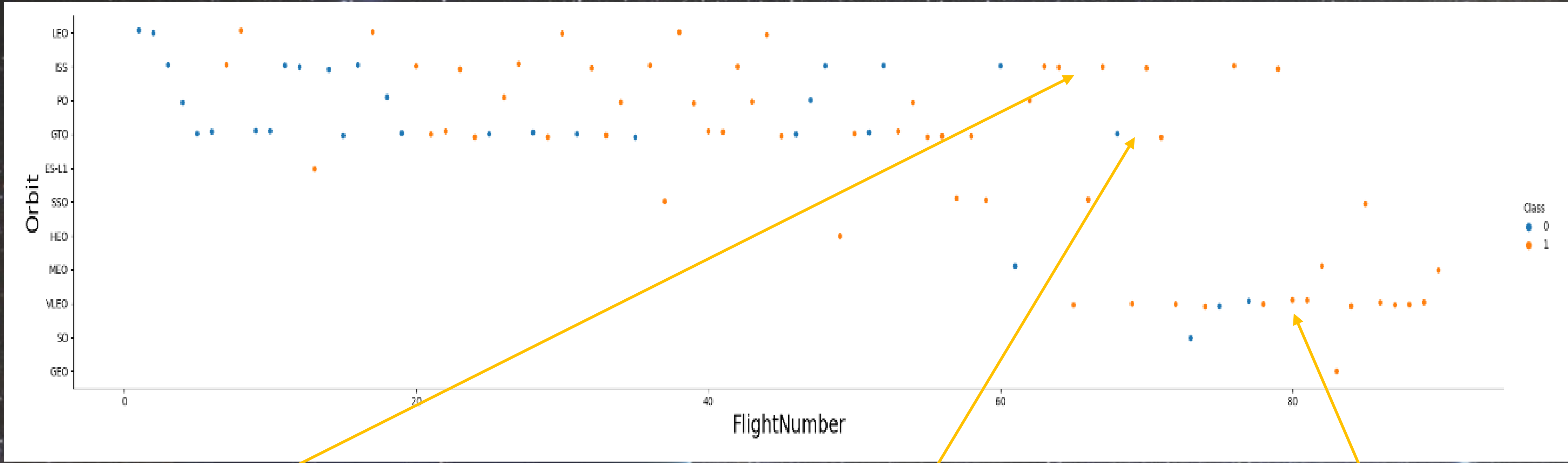
For each Launch Site with higher Payload [greater than 7000 Kg] the success rate was increased, but too heavy payloads could make a landing fail.

Success Rate vs. Orbit Type



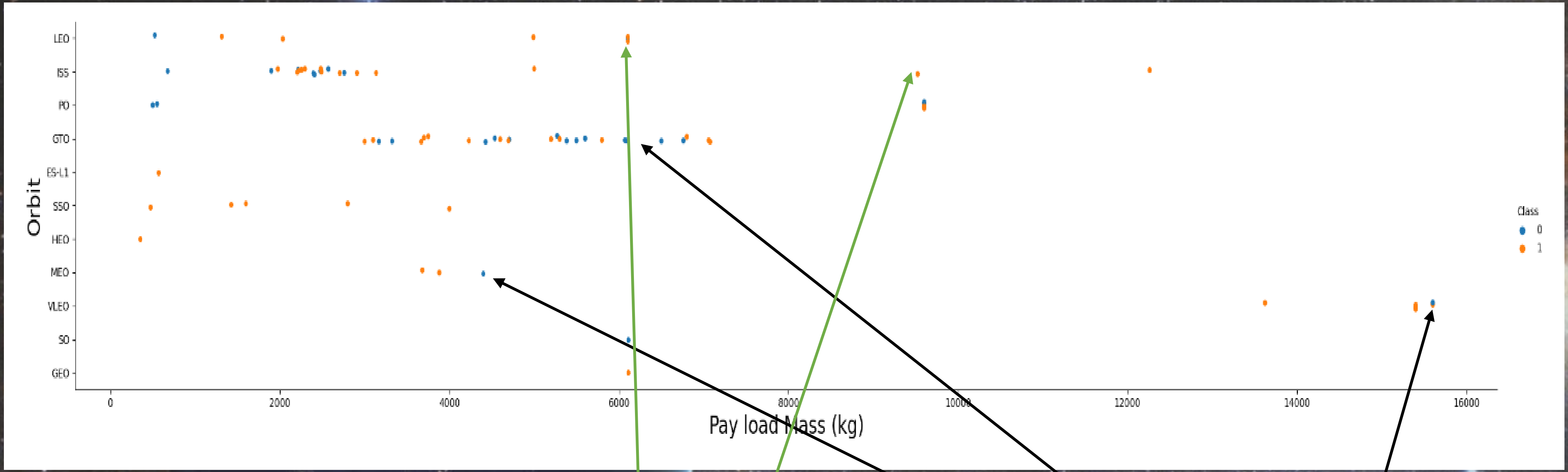
This Bar Chart shows that Orbit Types with highest Success Rate are: ES-L1, GEO, HEO and SSO followed by VLEO.

Flight Number vs. Orbit Type



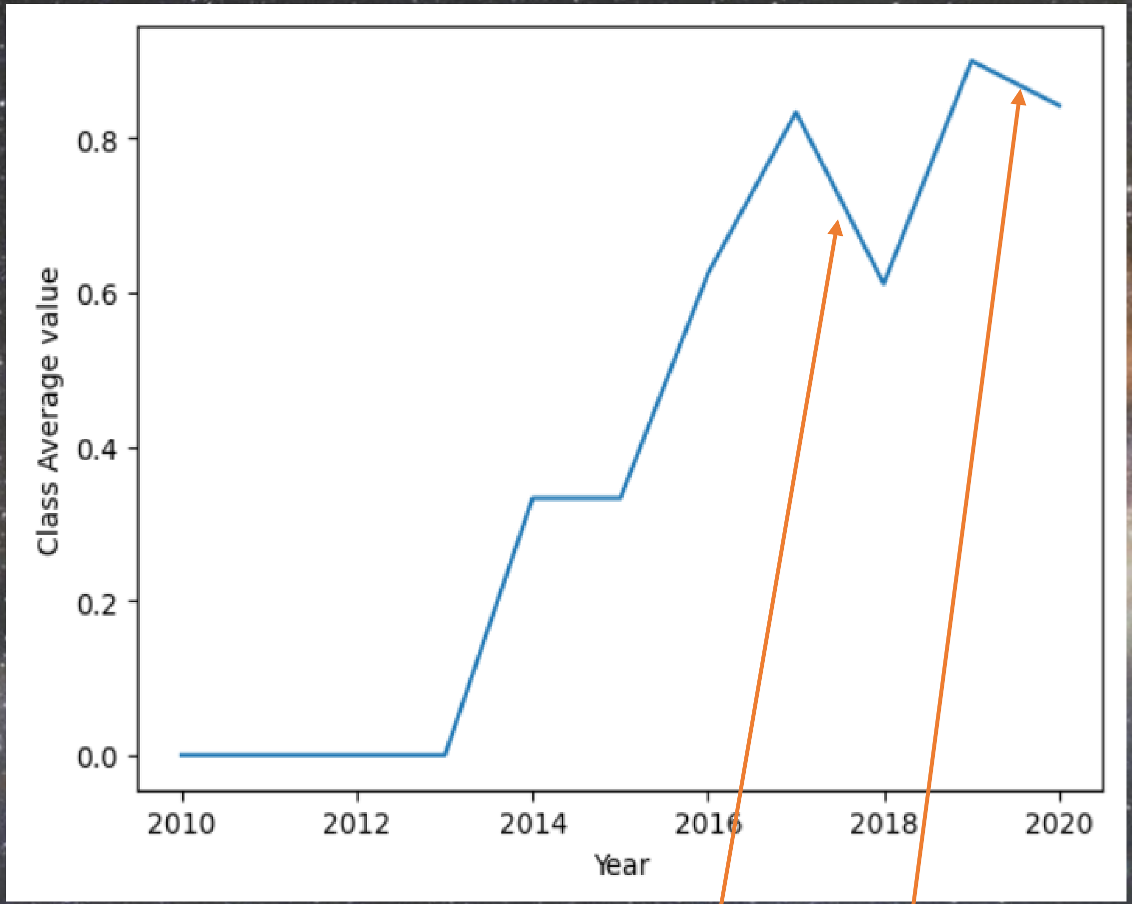
For LEO the success increases with the number of flights, for GTO seems to be no relationship between number of flights and success rate. Also a higher rate of success that increases with the number of flights has VLEO.

Payload vs. Orbit Type



Heavy payloads have a positive influence on LEO and ISS and a negative one on MEO, GTO and VLEO.

Launch Success Yearly Trend



The success rate started to increase since 2013 and kept an ascending trend, despite 2017 and 2019 small losses.

All Launch Site Names

SQL Query

```
%%sql
Select DISTINCT Launch_Site from SPACEXTBL
```

Output

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

Explanation: DISTINCT function is used to specify that the statement returns unique values in the specified column Launch_Site.

Launch Site Names Begin with 'CCA' [5 records]

SQL Query

```
%%sql
Select * from SPACEXTBL
Where Launch_site like 'CCA%' Limit 5
```

Output

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Explanation: Like operator is used in a Where clause to search for a specified pattern in the column. The wild-card % is used in conjunction with Like operator and represent the unknown rest of the expression searched. Limit 5 is a clause to specify the maximum number of rows the output must have.

Total Payload Mass [carried by boosters from NASA]

SQL Query

```
%%sql
Select SUM(PAYLOAD_MASS__KG_) from SPACEXTBL
Where Customer = 'NASA (CRS)'
```

Output

SUM(PAYLOAD_MASS__KG_)
45596

Explanation: SUM function calculates the total in the PAYLOAD_MASS__KG_ column and Where clause filter data to choose only Customer with name “NASA (CRS)”.

Average Payload Mass carried by booster version F9 v1.1

SQL Query

```
%%sql
Select AVG(PAYLOAD_MASS__KG_) from SPACEXTBL
Where Booster_Version = 'F9 v1.1'
```

Output

AVG(PAYLOAD_MASS__KG_)
2928.4

Explanation: AVG function calculates the average in the PAYLOAD_MASS__KG_ column and Where clause filter data to only perform calculation for Booster_Version with name “F9 v1.1”.

First Successful Ground Landing Date

SQL Query

```
%%sql
SELECT MIN(Date) as First_succesful_landing_outcome_in_ground_pad ,"Landing _Outcome" from SPACEXTBL
Where "Landing _Outcome"='Success (ground pad)'
```

Output

First_succesful_landing_outcome_in_ground_pad	Landing _Outcome
01-05-2017	Success (ground pad)

Explanation:The WHERE clause filters dataset in order to keep only records where landing was successful.
The MIN function, select the record with the oldest date.

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query

```
%%sql
SELECT booster_version,PAYLOAD_MASS__KG_ ,"Landing _Outcome" from SPACEXTBL
where  "Landing _Outcome"='Success (drone ship)' and PAYLOAD_MASS__KG_ >4000 and PAYLOAD_MASS__KG_ < 6000
```

Output

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)

Explanation: The query returns the booster version where landing was successful and payload mass was between 4000 and 6000 kg. The where clause together with and operators filter the dataset.

Total Number of Successful and Failure Mission Outcomes

SQL Query

```
%%sql
Update SPACEXTBL
SET "Mission_Outcome" = 'Success'
Where "Mission_Outcome" like 'Success%';
Update SPACEXTBL
SET "Mission_Outcome" = 'Failure'
Where "Mission_Outcome" like 'Failure%';
Select "Mission_Outcome",Count("Mission_Outcome") as 'Total_Number_of_Mission_Outcomes' from SPACEXTBL
Group by "Mission_Outcome"
Order by Count("Mission_Outcome") Desc
```

Output

Mission_Outcome	Total_Number_of_Mission_Outcomes
Success	100
Failure	1

Explanation: Update command updates the table SPACEXTBL with the results of SET function, which have modified the records in “Mission_Outcome” column, so that, no matters what appears after the string “Success”, all the record being considered “Success” and similar for the records that start with the string “Failure”.
The Count function make the total for the unique values that Group by function output in column .
Order by function together with Desc operator ensured that the output was given in descending order.

Boosters Carried Maximum Payload

SQL Query

```
%%sql
SELECT booster_version,PAYLOAD_MASS__KG_ as maximum_payload_mass from SPACEXTBL
Where PAYLOAD_MASS__KG_ = (Select MAX(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

Output

Booster_Version	maximum_payload_mass
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

Explanation: Sub query was used in Where clause to choose in the “PAYLOAD_MASS__KG_” column only the records with maxim weight.

2015 Launch Records

Listed Records which will display the **month names**, failure landing_outcomes in drone ship, booster versions, launch_site for the months in year 2015.

SQL Query

```
%%sql
DROP TABLE LookupMonth;
CREATE TABLE LookupMonth(month_name CHAR(20), month_num INT(2) PRIMARY KEY NOT NULL);
INSERT INTO LookupMonth(month_name,month_num)VALUES("January","01");
INSERT INTO LookupMonth(month_name,month_num)VALUES("February","02");
INSERT INTO LookupMonth(month_name,month_num)VALUES("March","03");
INSERT INTO LookupMonth(month_name,month_num)VALUES("April","04");
INSERT INTO LookupMonth(month_name,month_num)VALUES("May","05");
INSERT INTO LookupMonth(month_name,month_num)VALUES("June","06");
INSERT INTO LookupMonth(month_name,month_num)VALUES("July","07");
INSERT INTO LookupMonth(month_name,month_num)VALUES("August","08");
INSERT INTO LookupMonth(month_name,month_num)VALUES("September","09");
INSERT INTO LookupMonth(month_name,month_num)VALUES("October","10");
INSERT INTO LookupMonth(month_name,month_num)VALUES("November","11");
INSERT INTO LookupMonth(month_name,month_num)VALUES("December","12");
SELECT * FROM LookupMonth;
```

```
%%sql
Select substr(Date, 4, 2) as 'Month_number',month_name as 'Month_Name',substr(Date,7,4) as 'Year',
"Booster_Version","Launch_Site","Landing _Outcome"
from SPACEXTBL Left Outer Join LookupMonth
on substr(Date, 4, 2)=month_num
Where substr(Date,7,4)='2015' and "Landing _Outcome"='Failure (drone ship)';
```

Output

month_name	month_num
January	1
February	2
March	3
April	4
May	5
June	6
July	7
August	8
September	9
October	10
November	11
December	12

Month_number	Month_Name	Year	Booster_Version	Launch_Site	Landing _Outcome
01	January	2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	April	2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

Explanation: As SQLLite does not support month names I needed to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.
I created LookupMonth table with 2 columns “month_name” and “month-num”.
I used Left Outer Join method to obtain the “Month_Name” column in the output of the query.

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

SQL Query

```
%%sql
Select "Date","Landing _Outcome",COUNT("Landing _Outcome") as LANDING_OUTCOME_COUNT from SPACEXTBL
Where "Landing _Outcome" like 'Success%' and substr(Date,7,4) || substr(Date,4,2) || substr(Date,1,2) between '20100604' and '20170320'
Group by "Landing _Outcome" Order by count("Landing _Outcome") Desc
```

Output

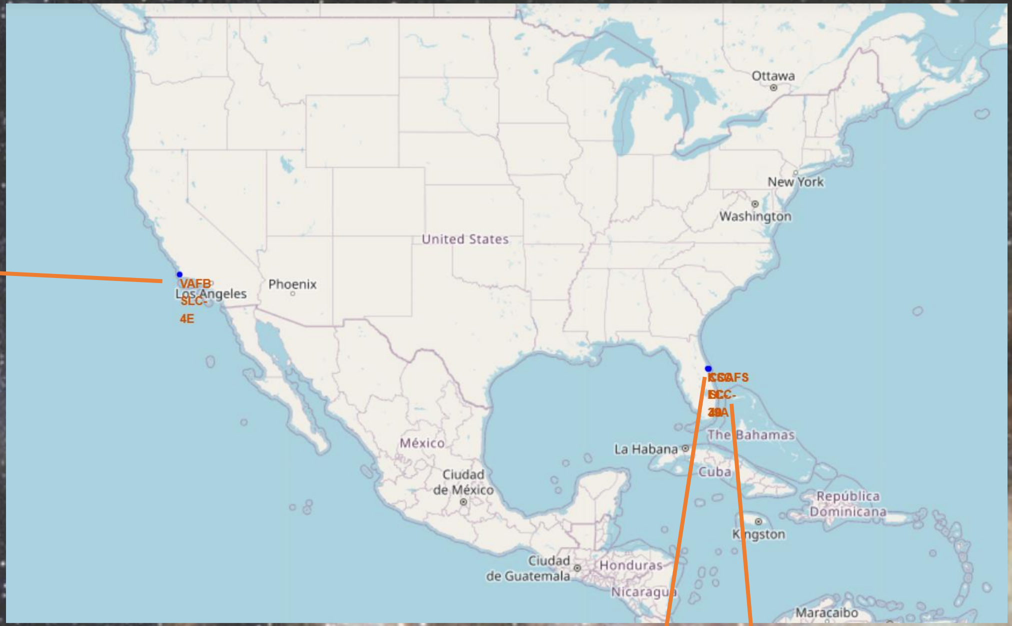
Date	Landing _Outcome	LANDING_OUTCOME_COUNT
08-04-2016	Success (drone ship)	5
22-12-2015	Success (ground pad)	3

Explanation: As SQLLite does not support days names, months names I needed to use substr(Date,7,4) for year, substr(Date, 4, 2) for month's number and substr(Date,1,2) method for the day's number.

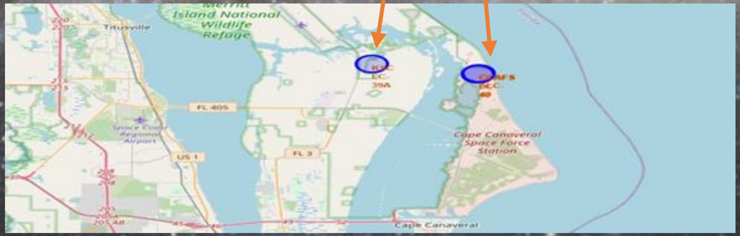
Section 3

Launch Sites Proximities Analysis

Launch Sites Locations on Folium Map

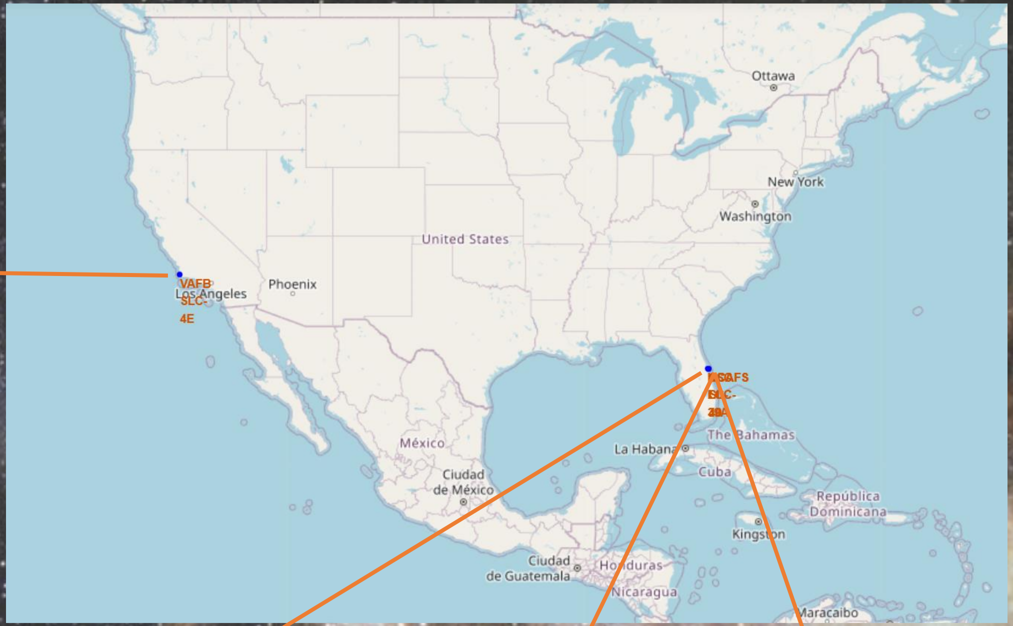


	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745



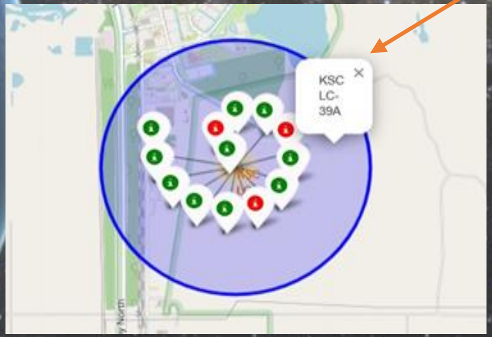
Explanation: The SpaceX Launch Sites are situated in very close proximity of the US coasts and in proximity of Equator line.

Color Marked Records on Folium Map



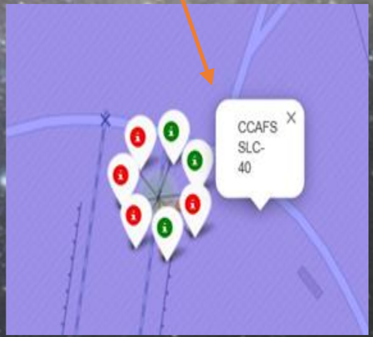
VAFB SLC-4E

KSC LC-39A



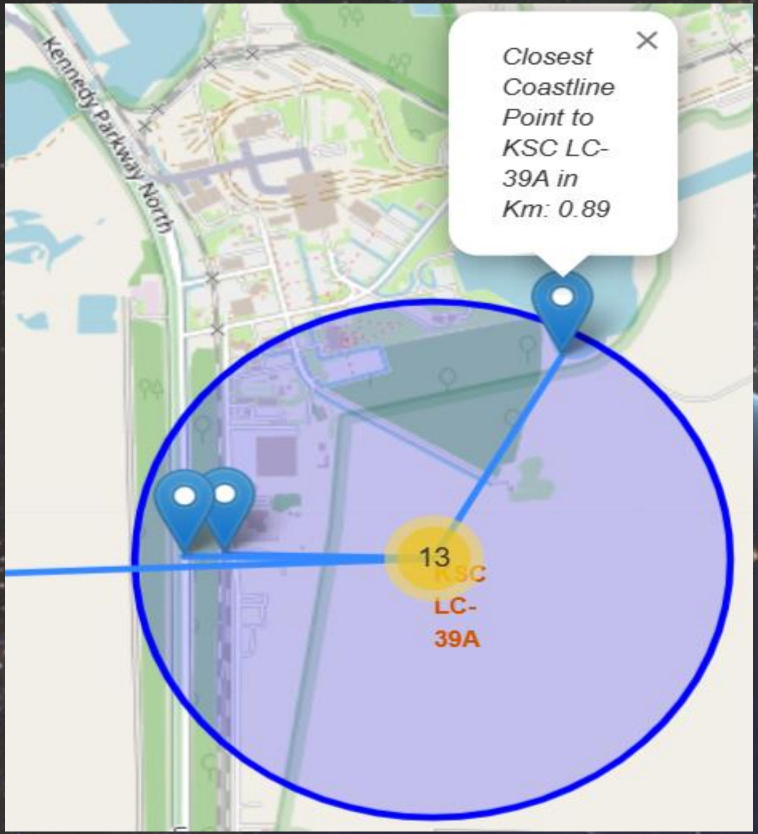
CCAFS LC-40

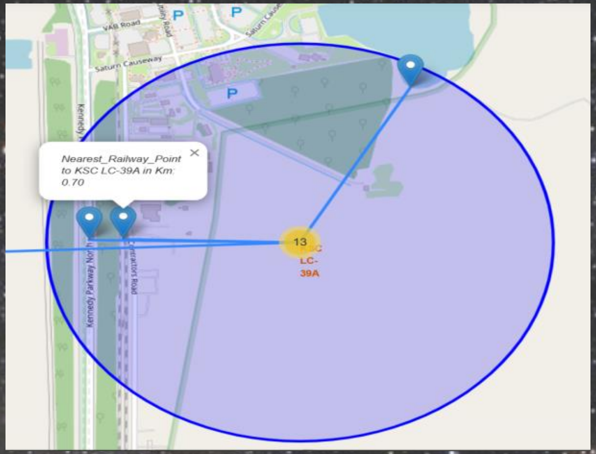
CCAFS SLC-40

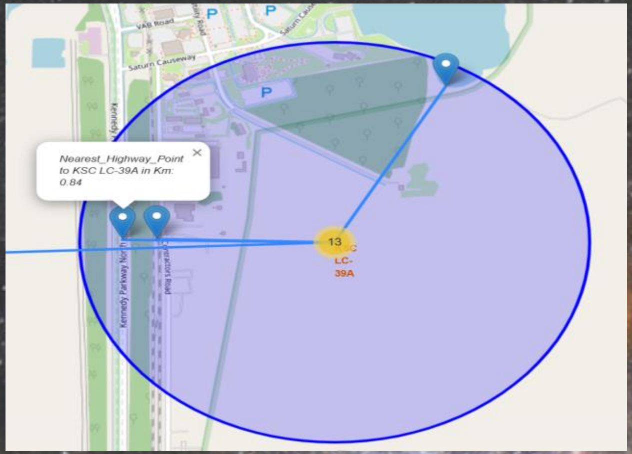


Explanation: **Green** marker shows success. **Red** marker shows failure.
KSC LC-39A has the higher success rate.

Distances from KSC LC-39A to its proximities on Folium Map







Explanation:

Is KSC LC-39Ain close proximity to railways ? Yes

Is KSC LC-39A in close proximity to highways ? Yes

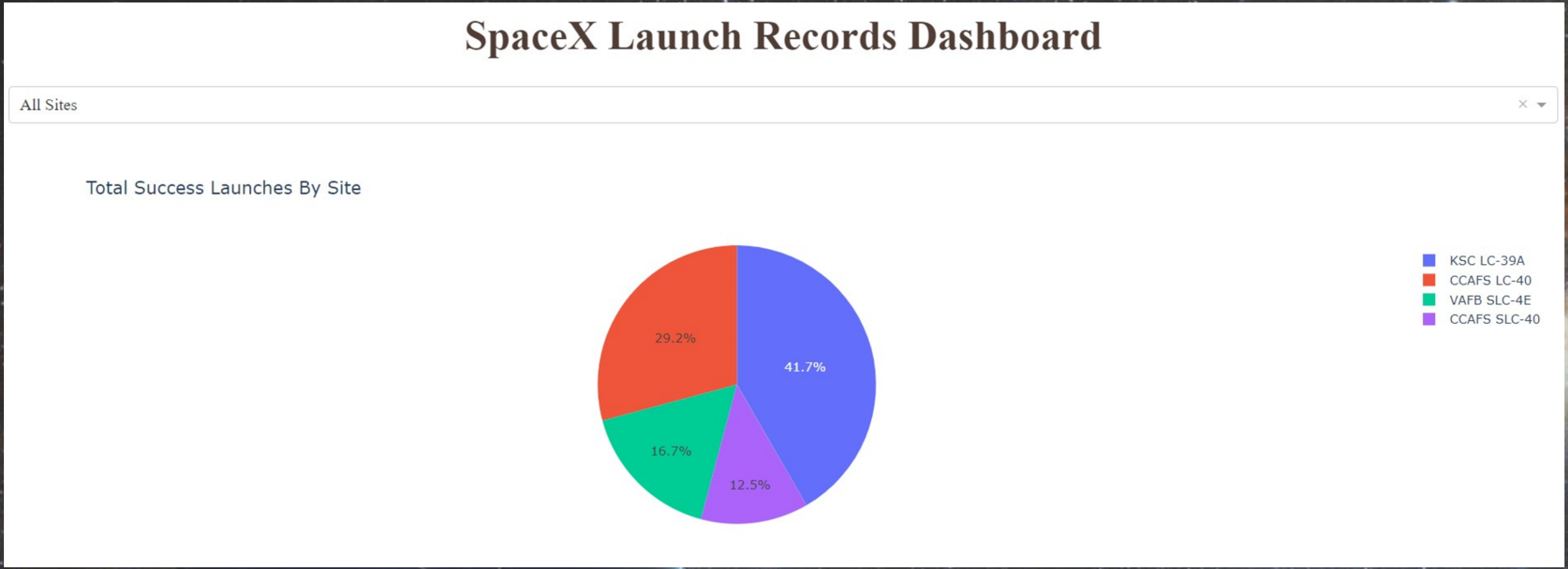
Is KSC LC-39A in close proximity to coastline ? Yes

Do KSC LC-39A keeps certain distance away from cities ? No

Section 4

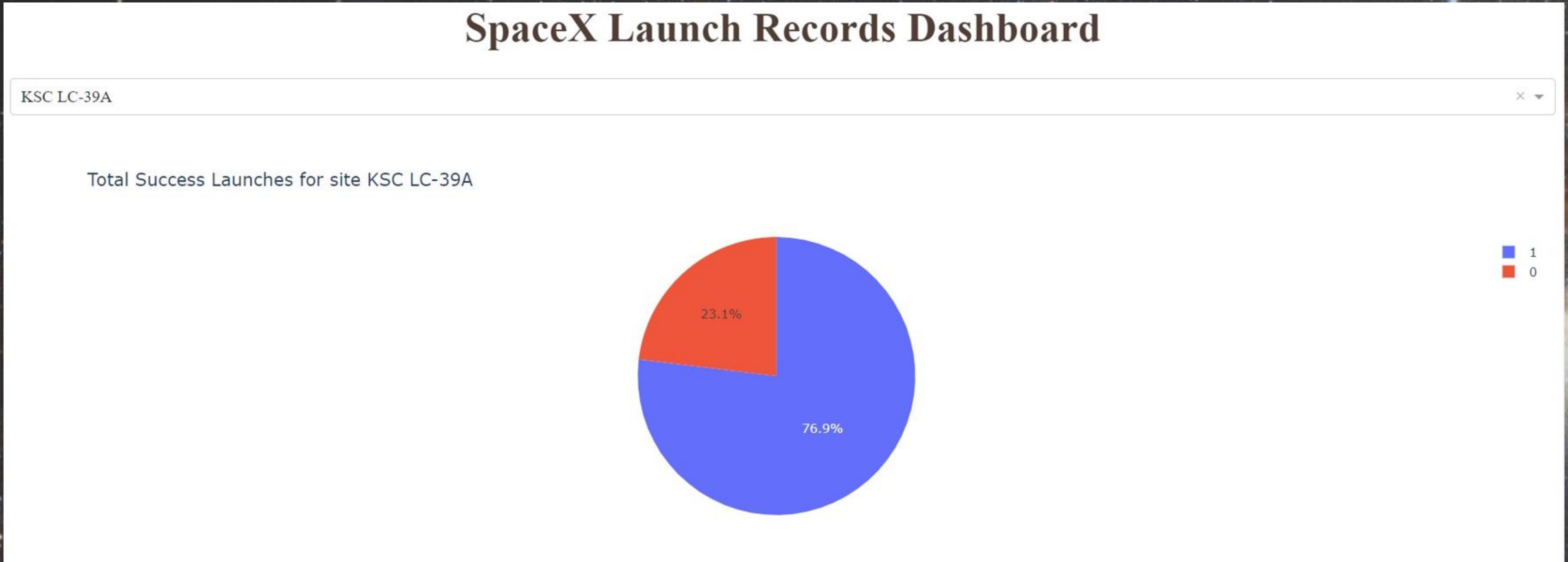
Build a Dashboard with Plotly Dash

Dashboard - Total Success by Launch Site



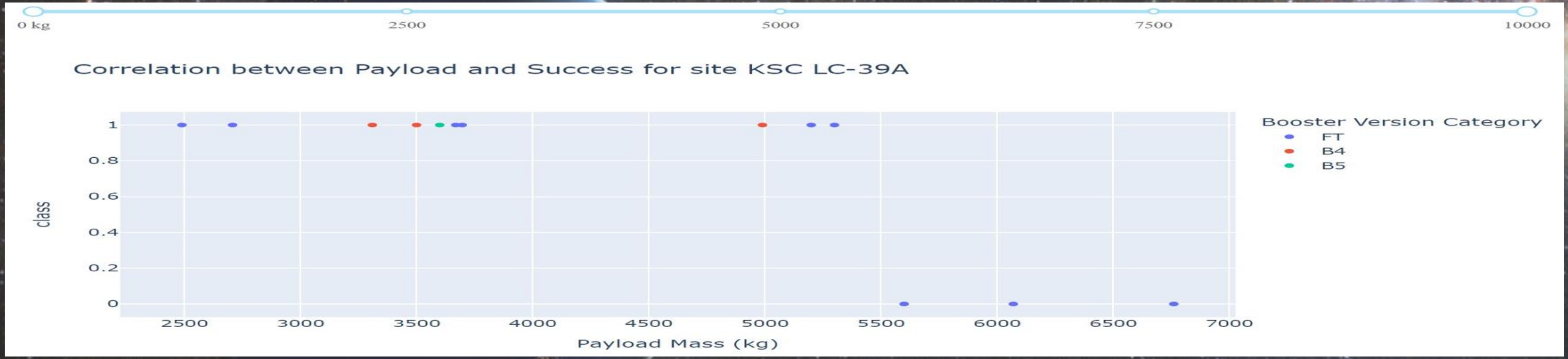
KSC LC-39A has the highest rate of success.

Dashboard – Launch Site with Highest Launch Success Ratio



KSC LC-39A has achieved 76.9% success rate and only 23.1% failure rate.

Dashboard – Payload vs. Launch Outcome Scatter Plot for All Sites



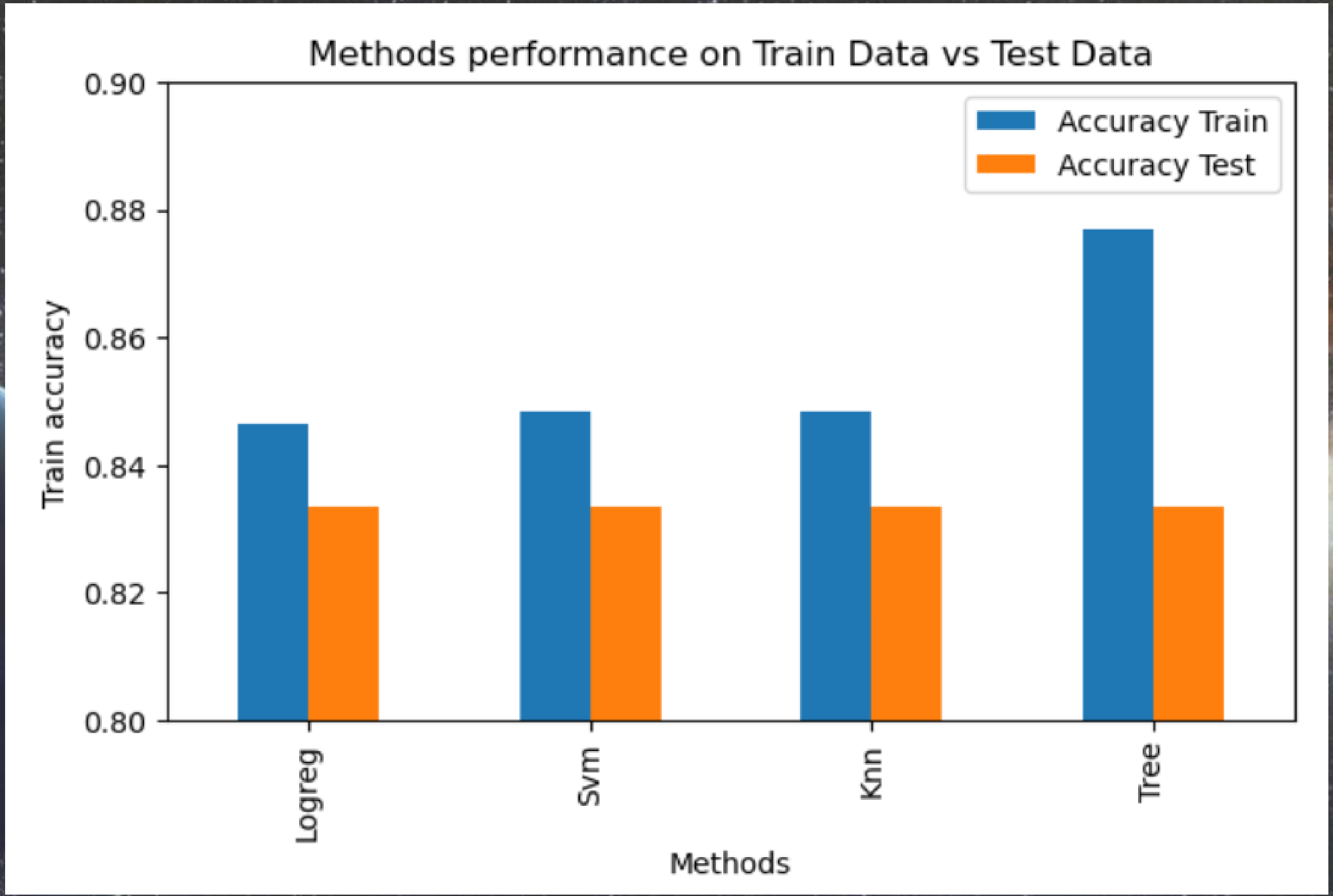
Low weighted payloads have a better success rate than the heavy weighted payloads for all Launch Sites as well as for KSC L-39A.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

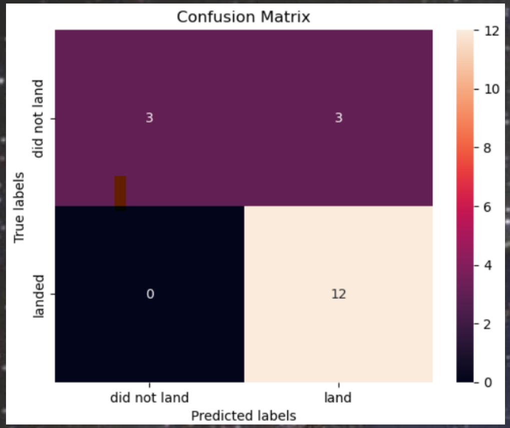
	Accuracy Train	Accuracy Test
Logreg	0.846429	0.833333
Svm	0.848214	0.833333
Knn	0.848214	0.833333
Tree	0.876786	0.833333



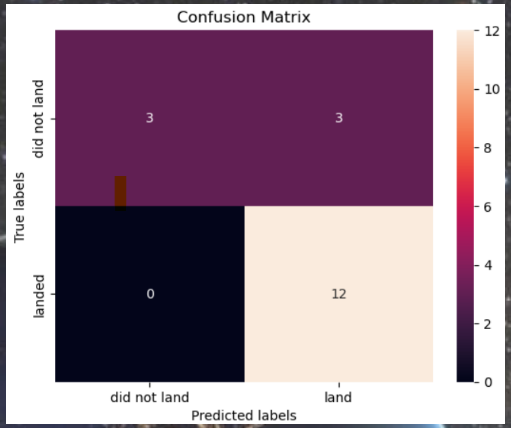
The model that has the highest classification accuracy is the one built with the method Decision Tree.

Confusion Matrix

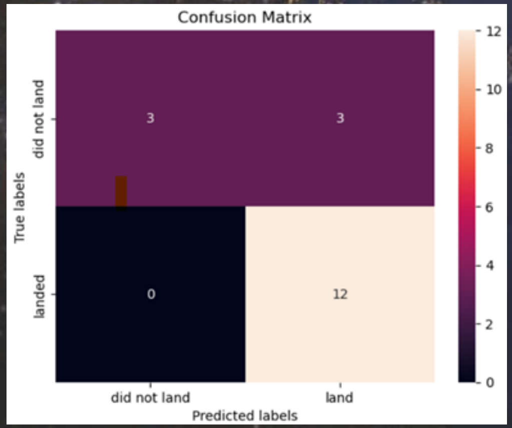
Logistic Regression



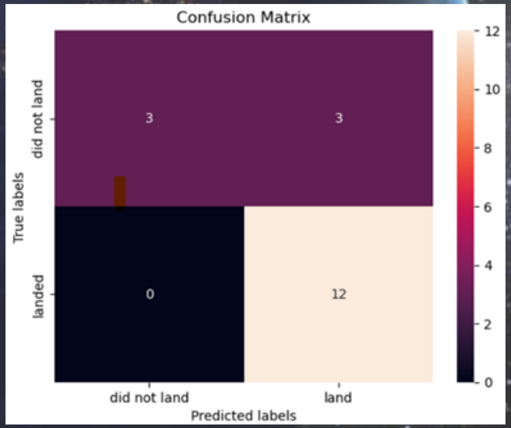
KNN

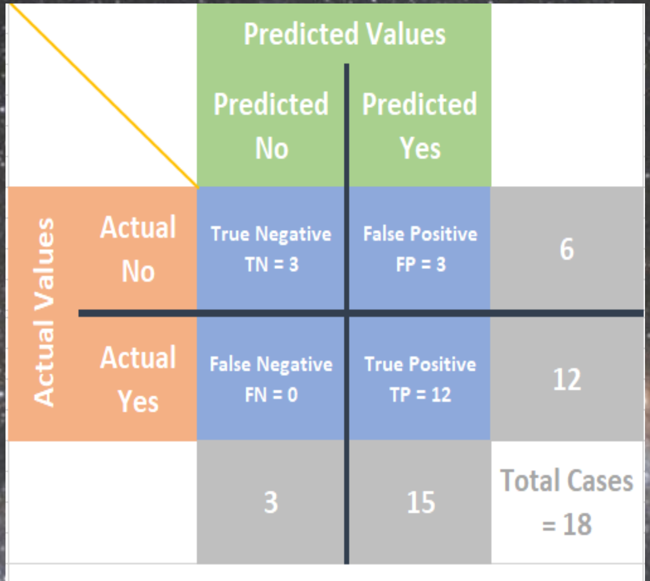


SVM



Decision Tree





Accuracy: (TP+TN)/Total = (12+3)/18 = 0.8333

Misclassification Rate: (FP+FN)/Total = (3+0)/18 = 0.1667

True Positive Rate: TP/Actual Yes = 12/12 =1

False Positive Rate: FP/Actual No = 3/6 = 0.5

True Negative Rate: TN/Actual No = 3/6 = 0.5

Precision: TP/Predicted Yes = 12/15 = 0.8

Prevalence: Actual Yes/Total = 12/18 = 0.6667

Test accuracy are all equal, so that the confusion matrices are also identical.
The main problem of these models are false positives.

Conclusions

- The success of a mission can be explained by several factors such as the launch site, the orbit and especially the number of previous launches. Indeed, we can assume that there has been a gain in knowledge between launches that allowed to go from a launch failure to a success.
- The Orbit Types with highest Success Rate are: ES-L1, GEO, HEO and SSO followed by VLEO.
- Depending on the orbits, the payload mass can be a criterion to take into account for the success of a mission. Generally low weighted payloads perform better than the heavy weighted payloads.
- KSC LC-39A is the best launch site. With the current data, we cannot explain why some launch sites are better than others. To get an answer to this problem, we could obtain atmospheric or other relevant data.
- For this dataset, we choose the Decision Tree Algorithm as the best model even if the test accuracy between all the models used is equal. We choose Decision Tree Algorithm because it has a better train accuracy.

Thank you!