Data Science Capstone Project

Predicting Falcon9 first stage landing

PRESENTED BY TANAEEM AHMAD

DATED 02/01/2023

Outline

KEY TOPICS DISCUSSED IN THIS PRESENTATION

- EXECUTIVE SUMMARY
- INTRODUCTION
- METHODOLOGY
- RESULTS
  - VISUALIZATION
  - DASHBOARD
- CONCLUSION
- APPENDIX

Executive Summary

- Predict success of Falcon 9 first stage landing
- Collect data on past launches through API and web scraping
- Build dashboard with Plotly Dash and interactive map with Folium to analyze data
- Split data into training and test sets to determine most effective machine learning method (SVM, Classification Trees, Logistic Regression)
- Use machine learning techniques to determine likelihood of successful landing
- Results can be used by other companies considering bidding against SpaceX for rocket launch

INTRODUCTION

Nature of analysis:

- Predictive analysis

Problem:

- Determine the likelihood of a successful first stage landing for a SpaceX Falcon 9 rocket launch

Questions for analysis:

- What factors contribute to the success of a Falcon 9 first stage landing?
- Can we accurately predict the success of a first stage landing using past data and machine learning techniques?
- Which machine learning method is the most effective in predicting the success of a first stage landing?

METHODOLOGY

Data sources:

1. RESTful API
2. Web scraping

Plans for collected data:

- Convert data into a dataframe
- Perform data wrangling to clean and prepare data for analysis
- Build a dashboard with Plotly Dash to interactively analyze launch records
- Build an interactive map with Folium to analyze launch site proximity
- Split data into training and test sets for machine learning analysis
- Determine most effective machine learning method for predicting success of first stage landing using test data

1.RESTful API

1

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

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-SkillsNetw
```

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

```
response.status_code
```

```
200
```

3

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']
```

Now that we have removed some values we should reset the FlgihtNumber column

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

4

FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs
1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False
2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False
3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False
4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False
5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

```
# Calculate the mean value of PayloadMass column
mean_payload_mass = df['PayloadMass'].mean()

# Replace the np.nan values with its mean value
df['PayloadMass'] = df['PayloadMass'].replace(np.nan, mean_payload_mass)
df['PayloadMass'].isnull().sum()
```

```
0
```

You should see the number of missing values of the `PayLoadMass` change to zero.

click on icon to view notebook

2.Webscraping

1

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
response = requests.get(static_url)
html_content = response.text
status_code = response.status_code
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_content, "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>
```

2

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

```
# 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")
```

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

3

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

[6]:
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_name
headers = first_launch_table.find_all("th")
for header in headers:
    name = extract_column_from_header(header)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

Check the extracted column names

[7]:
```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

4

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

Code in link .

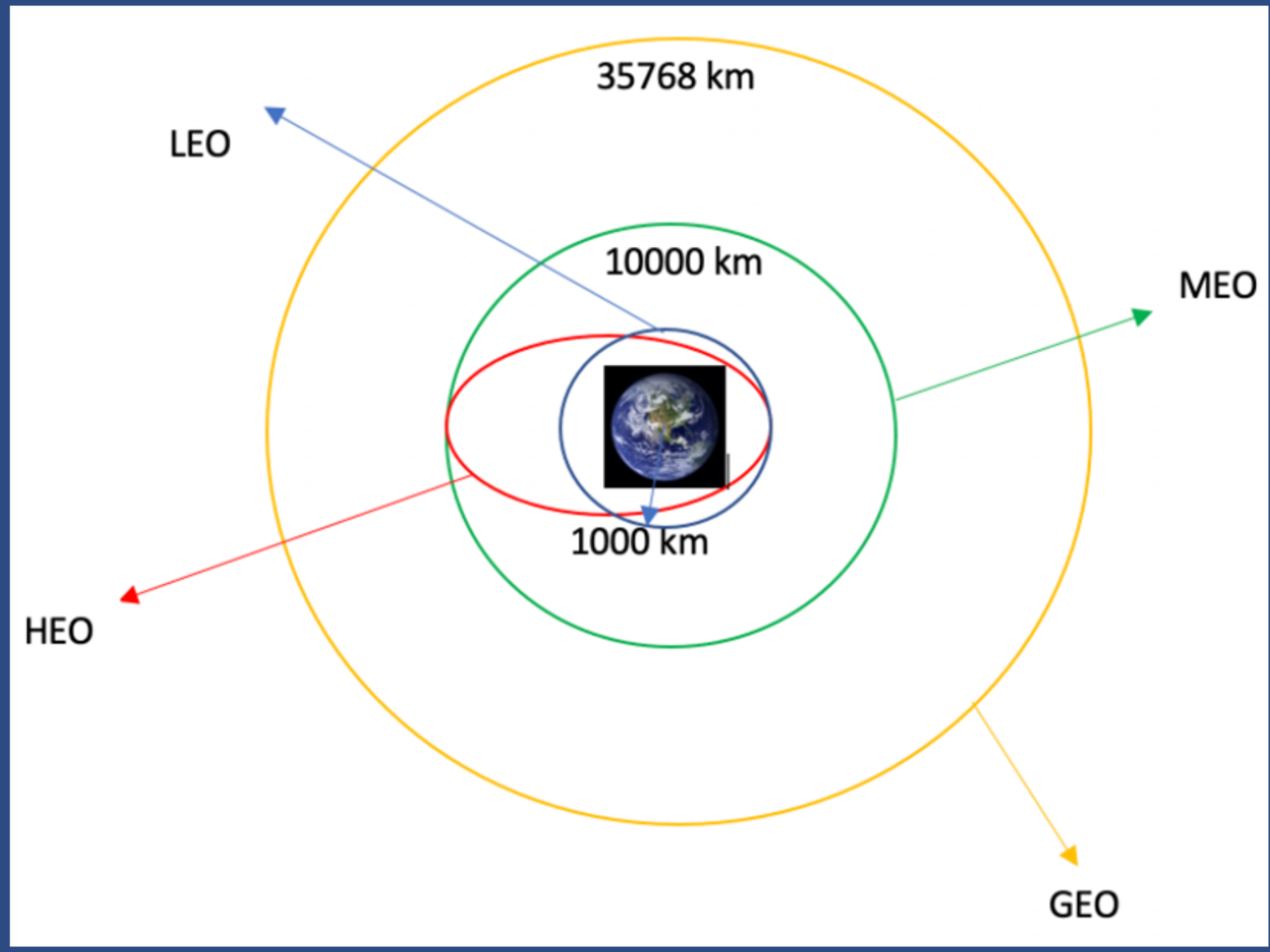
We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt

click on icon to view notebook

Data wrangling

We calculated the number of launches at each site, and the number and occurrence of each orbits



TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

]:
```
# Apply value_counts() on column LaunchSite
launch_site_counts = df["LaunchSite"].value_counts()
print(launch_site_counts)
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

```
7]: # Apply value_counts on Orbit column
orbit_counts = df["Orbit"].value_counts()
print(orbit_counts)
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: i
```

TASK 3: Calculate the number and occurence of mission outcome per orbit type

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

```
# landing_outcomes = values on Outcome column
landing_outcomes = df["Outcome"].value_counts()
print(landing_outcomes)
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [0 if row in bad_outcomes else 1 for row in df['Outcome']]
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

```
df['Class']=landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0

We can use the following line of code to determine the success rate:

```
df["Class"].mean()
```

```
0.6666666666666666
```

click on icon to view notebook

Results

A. Data Analysis

B. Charts & Dashboard

C. Best ML Model

A. Data Analysis

1

Using SQL

2

Using PANDAS & MATPLOTLIB

1

SQL

Task 1

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

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEX;
```

```
 * ibm_db_sa://kcv92123:***@21fecfd8-47b7-4937-840d-d791d0218660
d:31864/bludb
Done.
```

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

Task 2

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

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

```
 * ibm_db_sa://kcv92123:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od
d:31864/bludb
Done.
```

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

Task 3

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

```
]:
%%sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_Mass
FROM SPACEX
WHERE customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://k
d:31864/bludb
Done.
```

]: **total_mass**

total_mass
45596

Task 4

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

```
%%sql SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Mass
FROM SPACEX
WHERE BOOSTER_VERSION = 'F9 v1.1'
```

```
 * ibm_db_sa://kcv92123:***@21fecfd8-47b7-4937-840d-d791d0218660.
d:31864/bludb
Done.
```

: **average_mass**

average_mass
2928

Task 5

List the date when the first successful landing outcome in ground pad was acheived.

Hint:Use min function

```
%%sql SELECT MIN(DATE) AS First_Successful_Landing
FROM SPACEX1
WHERE LANDING__OUTCOME='Success';
```

```
 * ibm_db_sa://kcv92123:***@21fec
Done.
```

first_successful_landing
02-03-2019

Task 6

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

```
%%sql SELECT PAYLOAD AS Names_of_boosters
FROM SPACEX
WHERE LANDING__OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

```
 * ibm_db_sa://kcv92123:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.
d:31864/bludb
Done.
```

names_of_boosters
JCSAT-14
JCSAT-16
SES-10
SES-11 / EchoStar 105

Task 7

List the total number of successful and failure mission outcomes

```
%%sql SELECT MISSION_OUTCOME, COUNT(*) AS Total
FROM SPACEX
GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://kcv92123:***@21fecfd8-47b7-493
d:31864/bludb
Done.
```

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

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%%sql SELECT BOOSTER_VERSION
FROM SPACEX
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX);
```

```
 * ibm_db_sa://kcv92123:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.database
d:31864/bludb
Done.
```

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

Task 9

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

```
%%sql SELECT DATE,LANDING__OUTCOME,BOOSTER_VERSION,LAUNCH_SITE
FROM SPACEX1
WHERE  DATE LIKE '%2015' AND LANDING__OUTCOME='Failure (drone ship)';
```

```
 * ibm_db_sa://kcv92123:***@21fecfd8-47b7-4937-840d-d7
d:31864/bludb
Done.
```

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

Task 10

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

```
%%sql SELECT LANDING__OUTCOME, COUNT(*) as count
FROM SPACEX1
WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017'
GROUP BY LANDING__OUTCOME
ORDER BY count DESC
```

```
 * ibm_db_sa://kcv92123:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.clo
d:31864/bludb
Done.
```

landing__outcome	COUNT
Success	20
No attempt	11
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Controlled (ocean)	3
Failure	3
Failure (parachute)	2

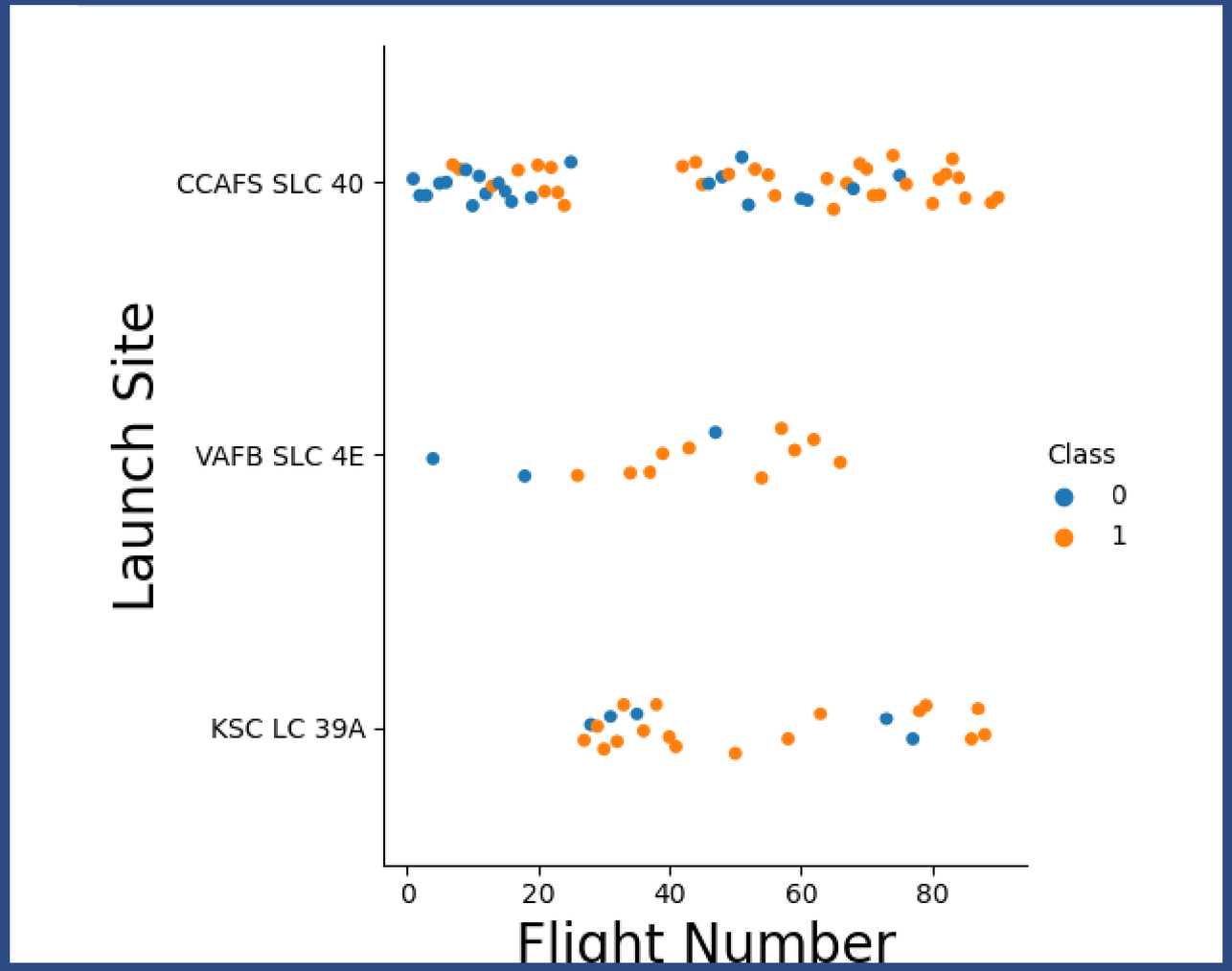
click on icon to view notebook

2

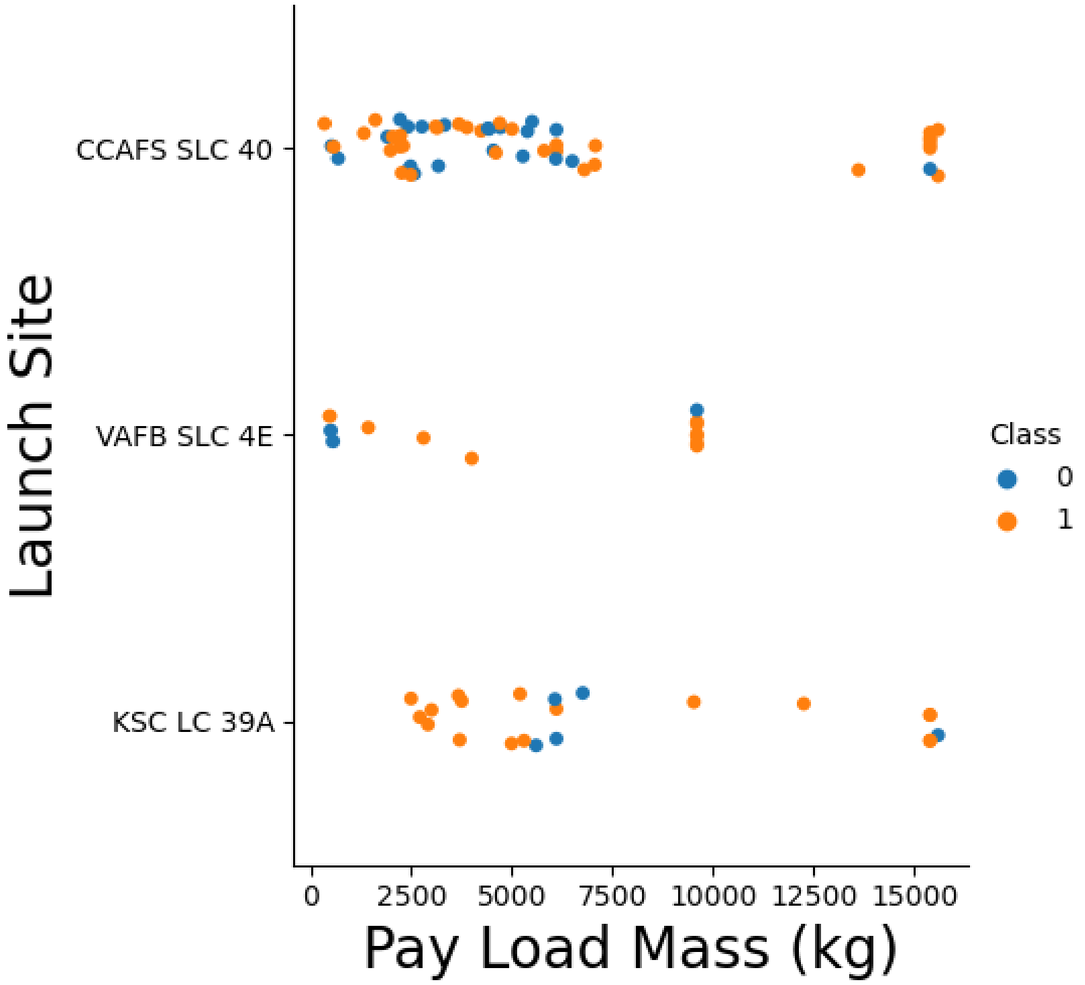
Pandas & Matplotlib

We view the relationships between different aspects of data set.

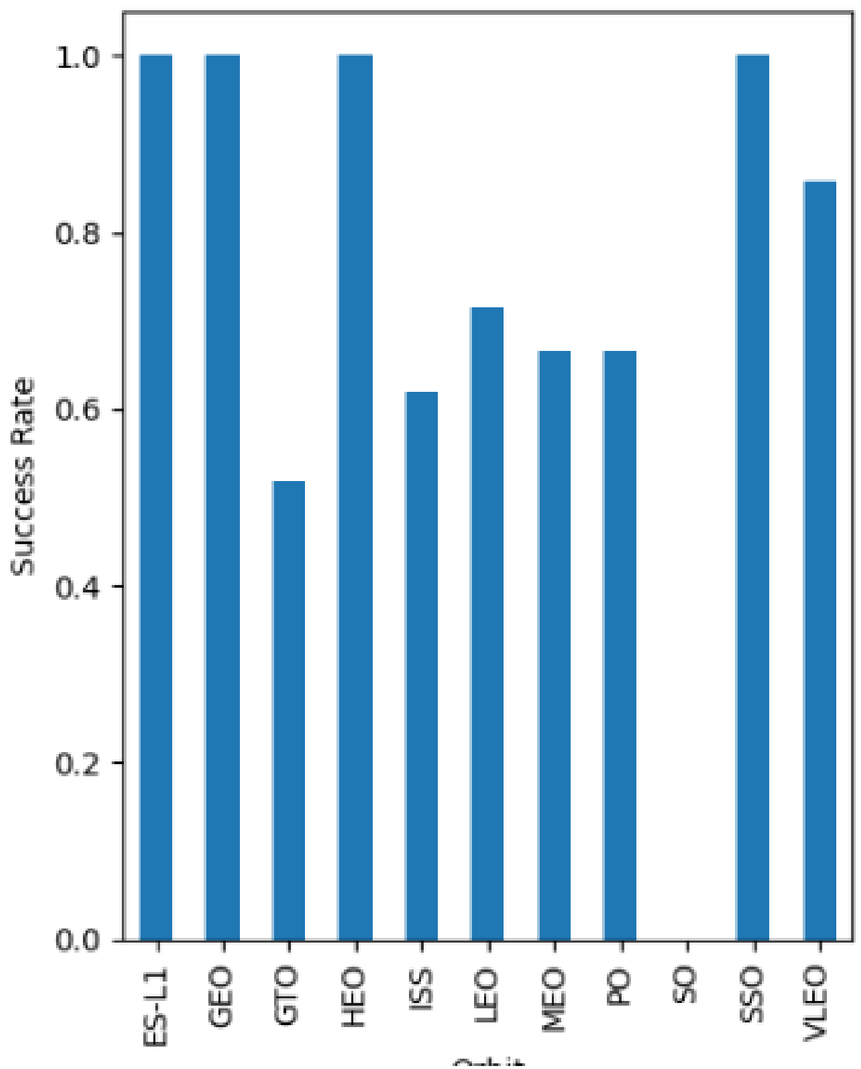
Class 0 Success
Class1 Failure



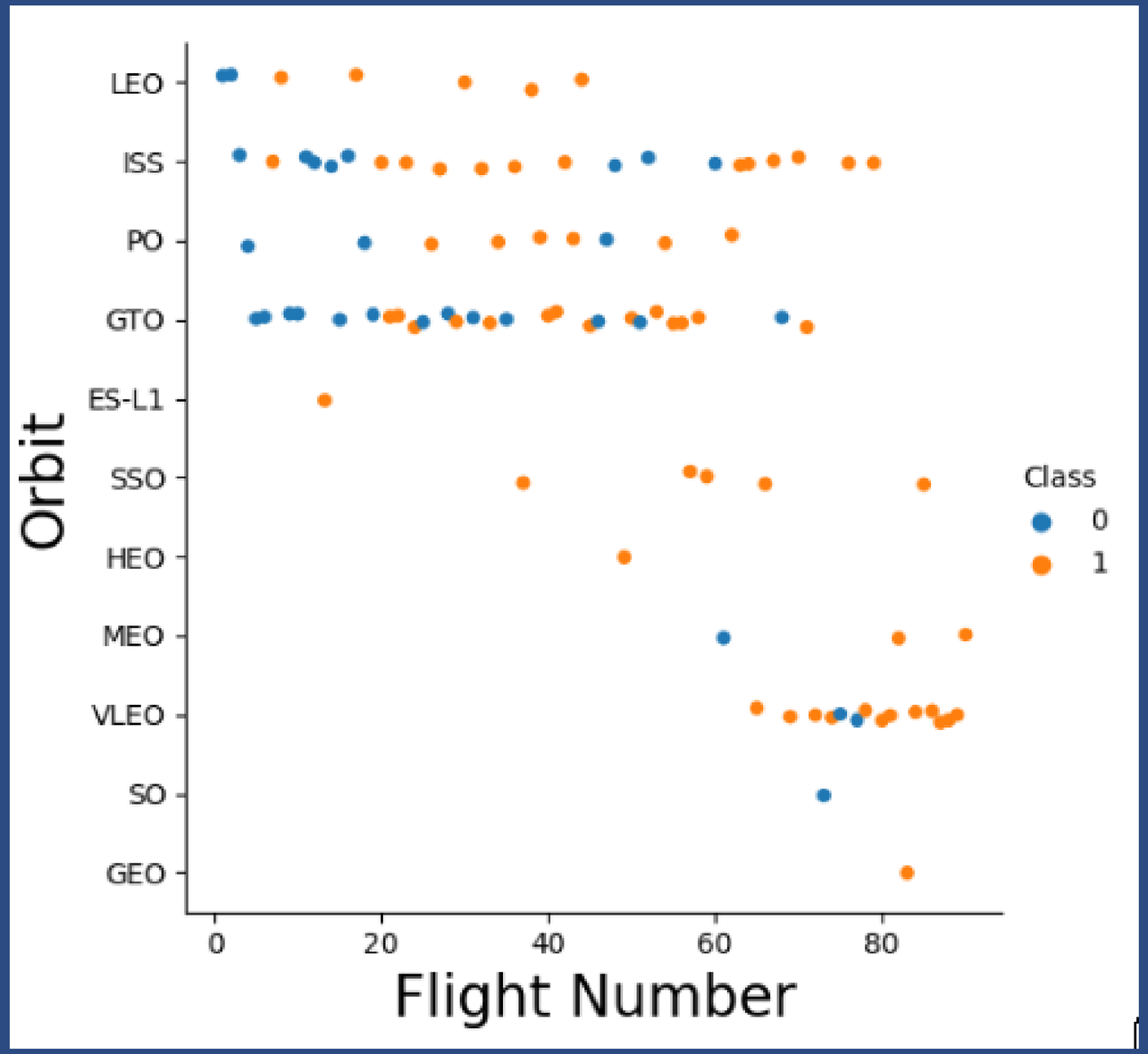
Larger the flight amount at a launch site, the greater the success rate



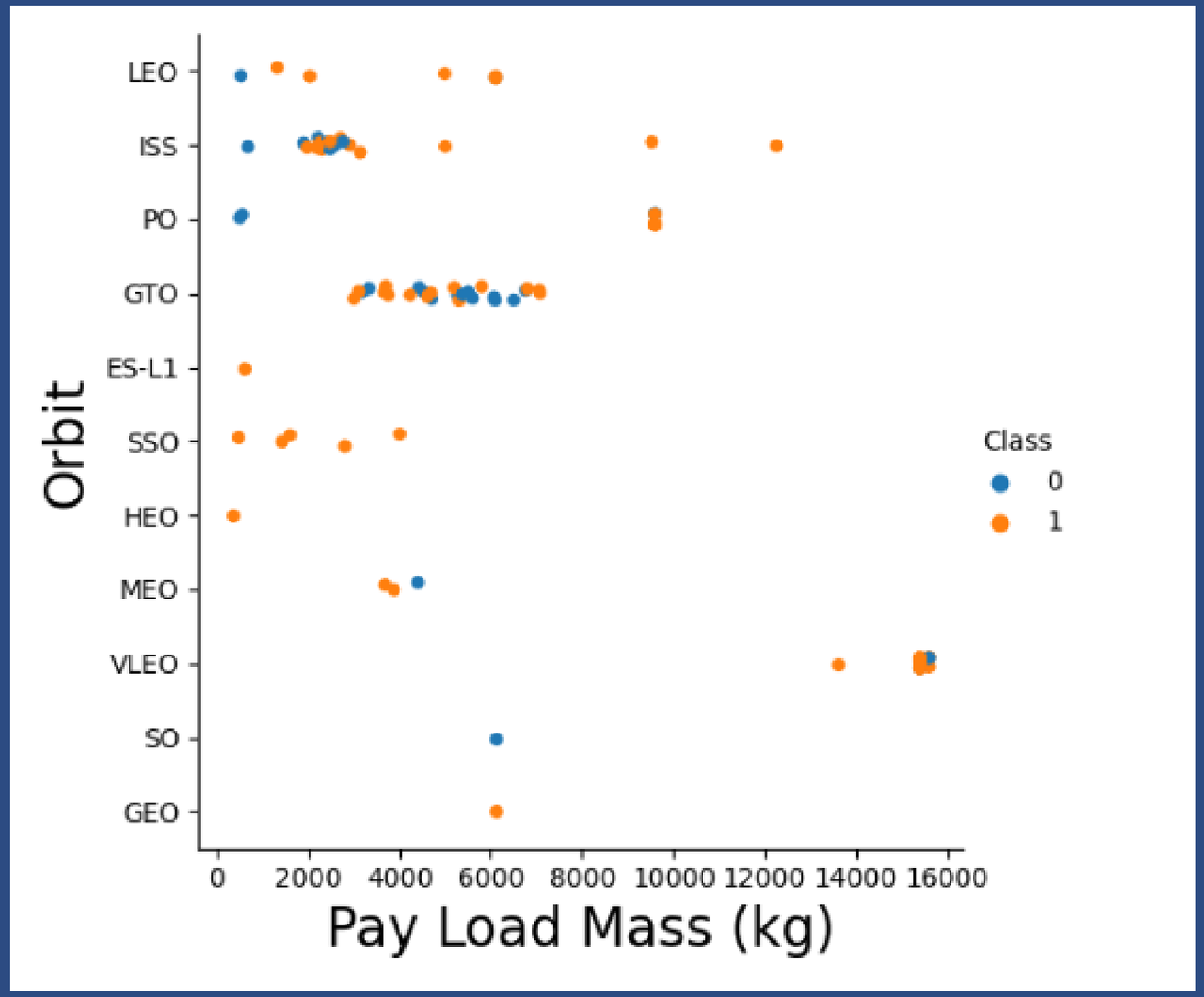
Greater the Payload Mass of CCAFS SLC-40 & KSC LC 39A greater the success rate



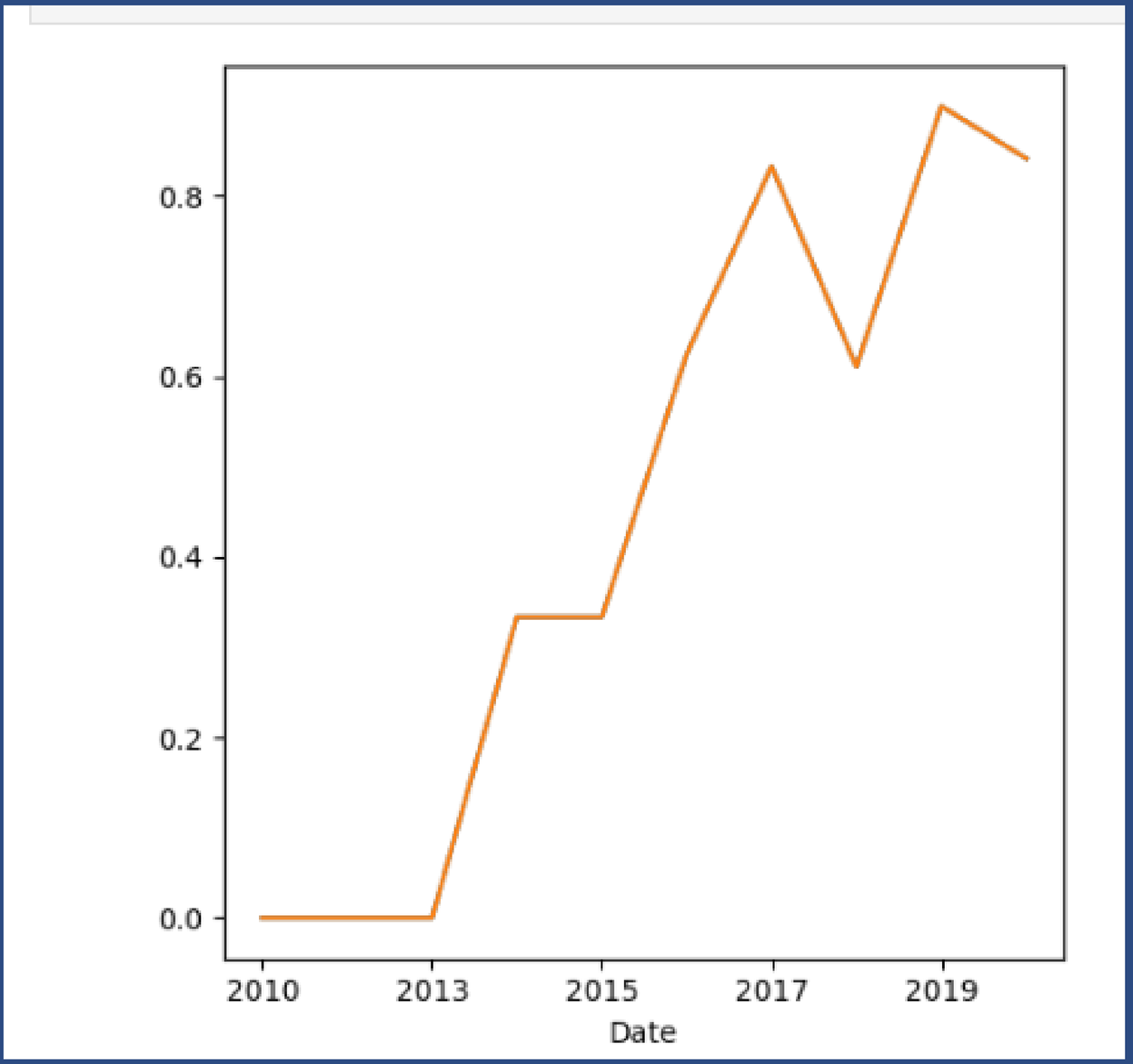
ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



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.



Heavy the payloads successful the landing are and more accurate for PO, LEO and ISS orbits.



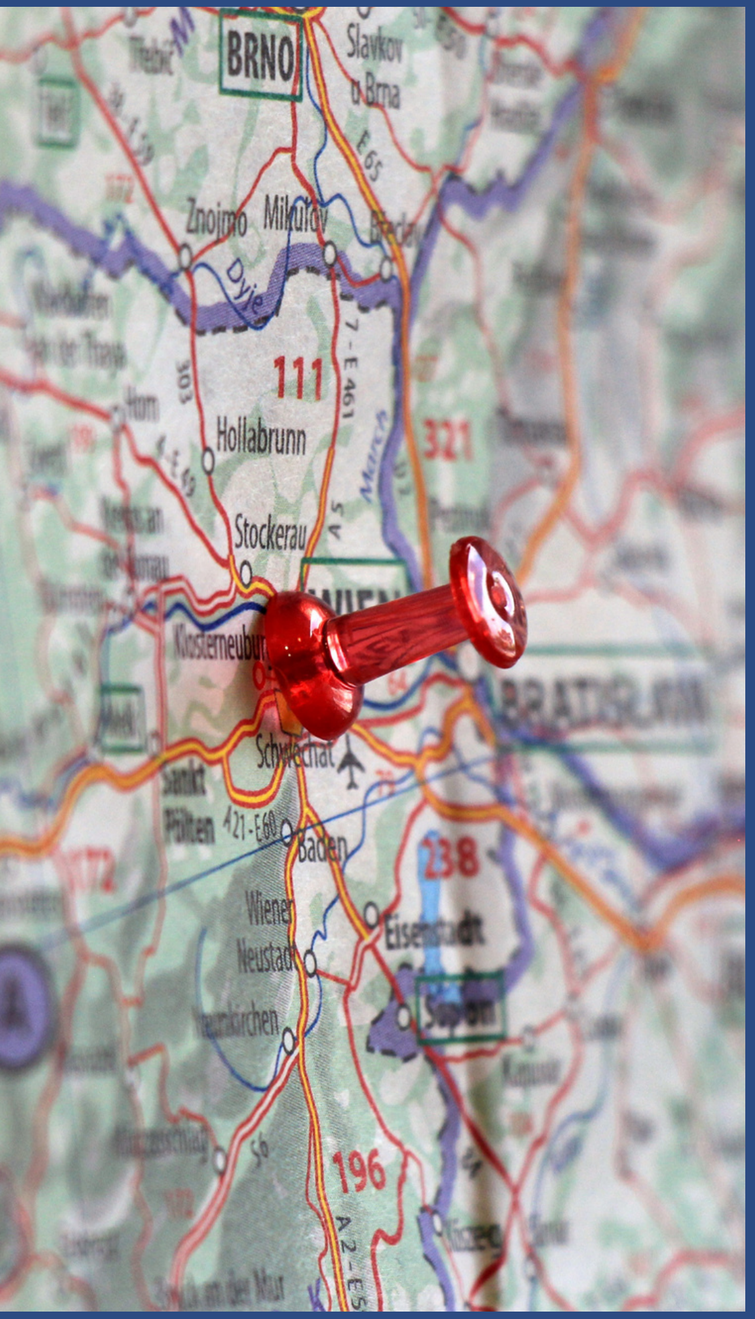
Yearly launch success

click on icon to view notebook

B. Charts & Dashboard

1. Insights From Interative Visuals(Folium)

2. Insights From Interative Visuals(Plotly Dash)



1. Insights From Interative Visuals

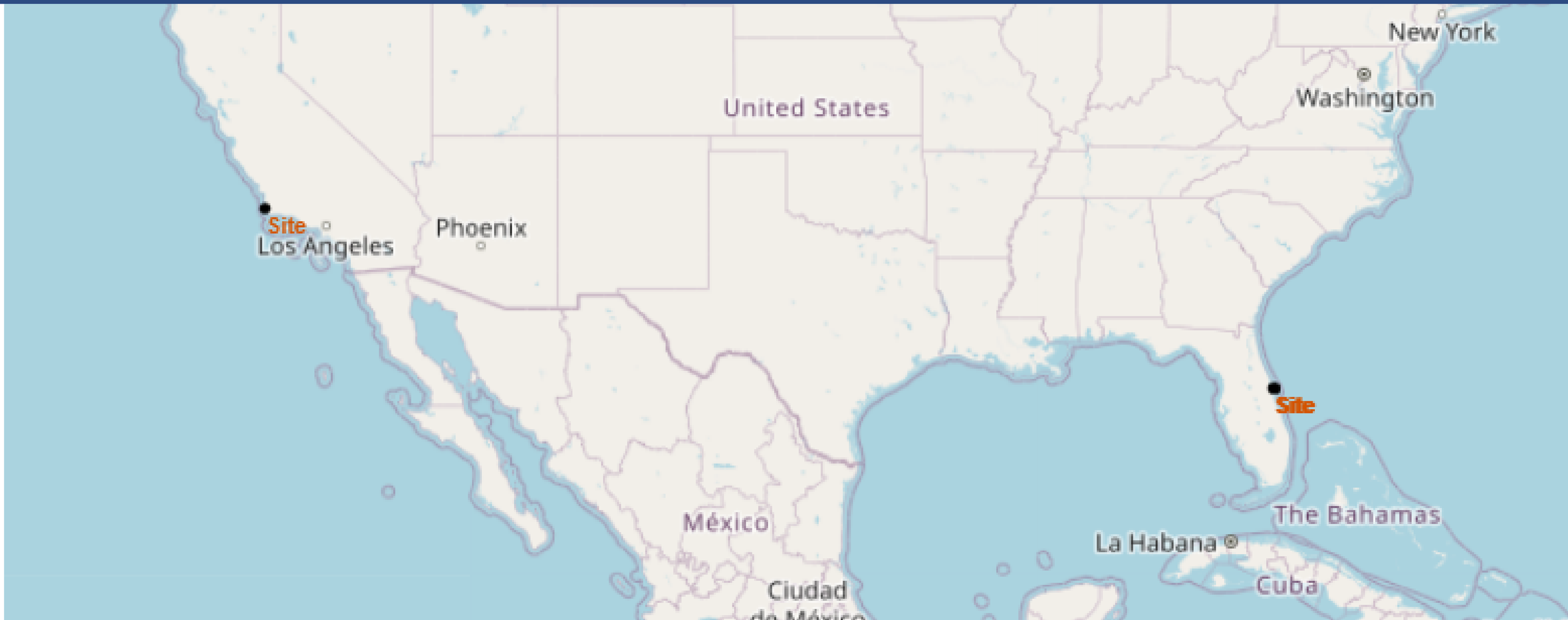
DONE BY USING FOLIUM

What factors contribute to the success of a Falcon 9 first stage landing?

Are all launch sites in very close proximity to the coast?

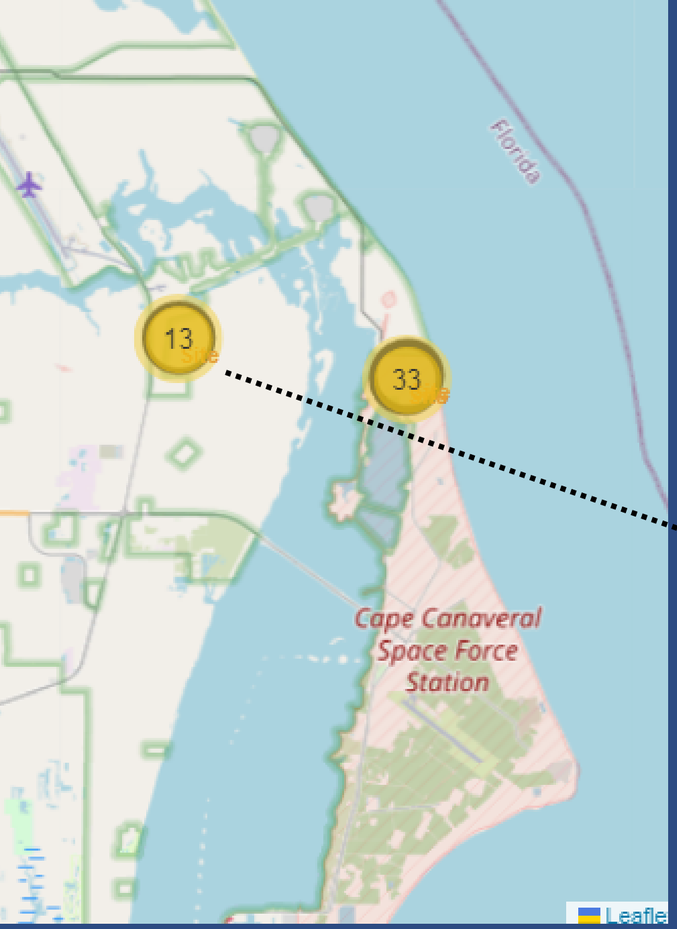
Which Site has more success rate?

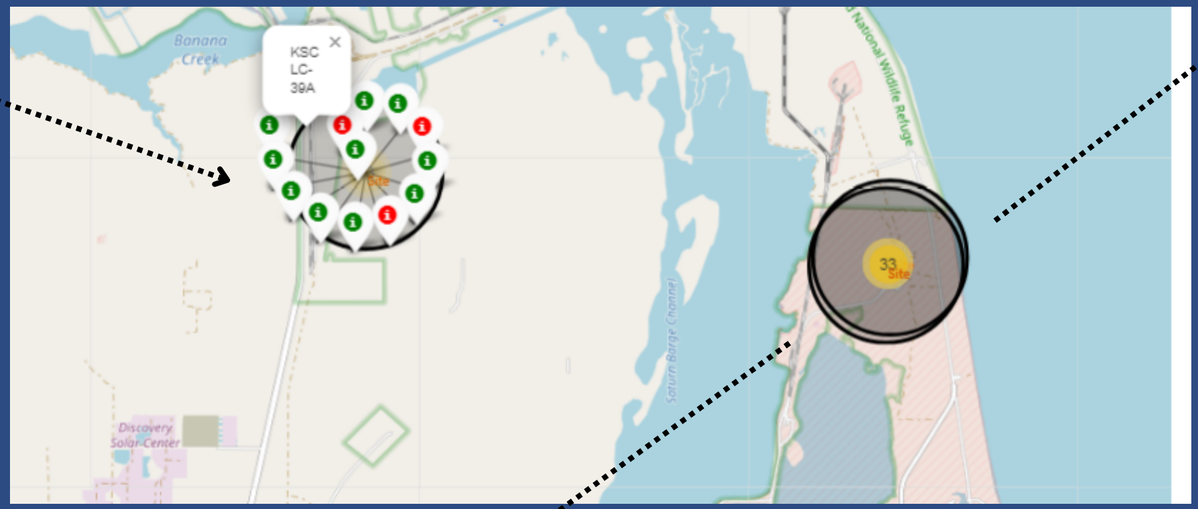
Are all launch sites in very close proximity to the coast? YES

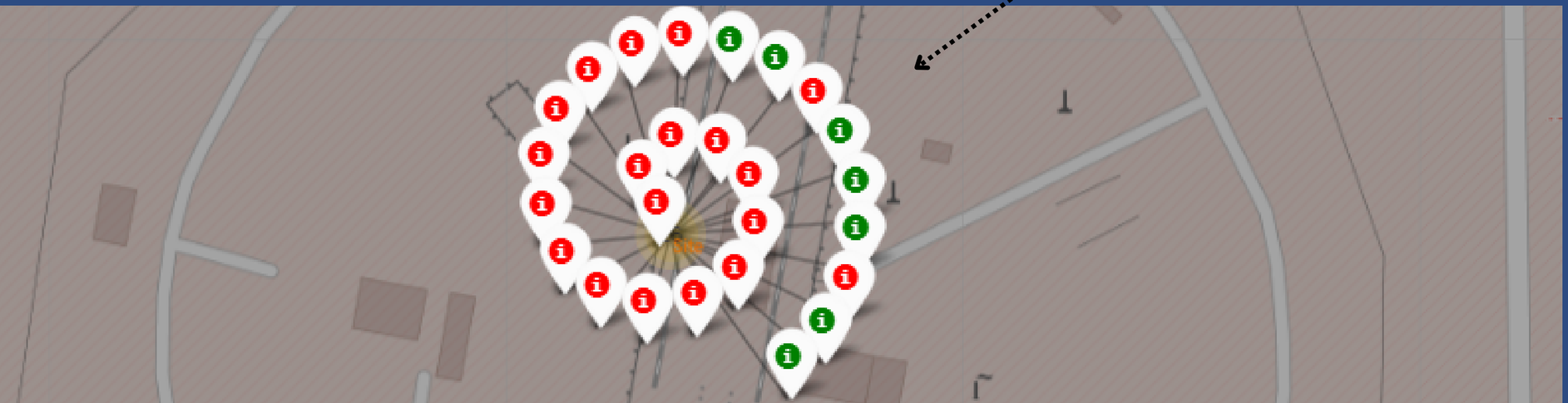


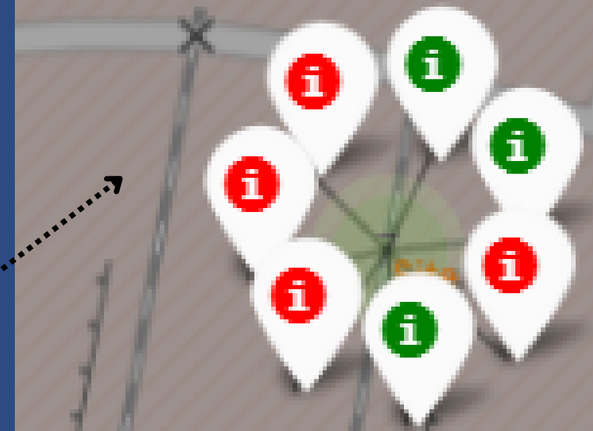
Which Site has more success rate?

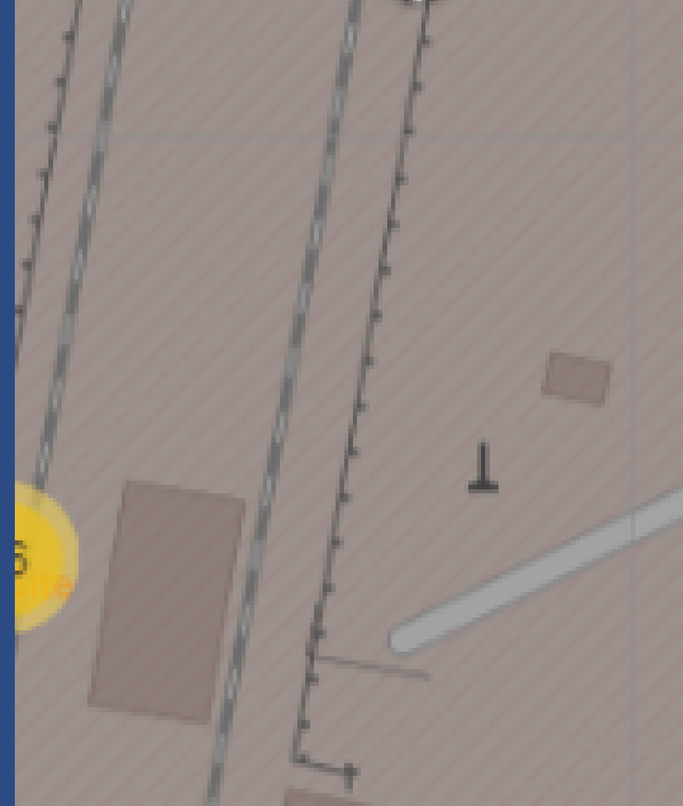
By zooming in the most laucnh site(46) and than comparing each site

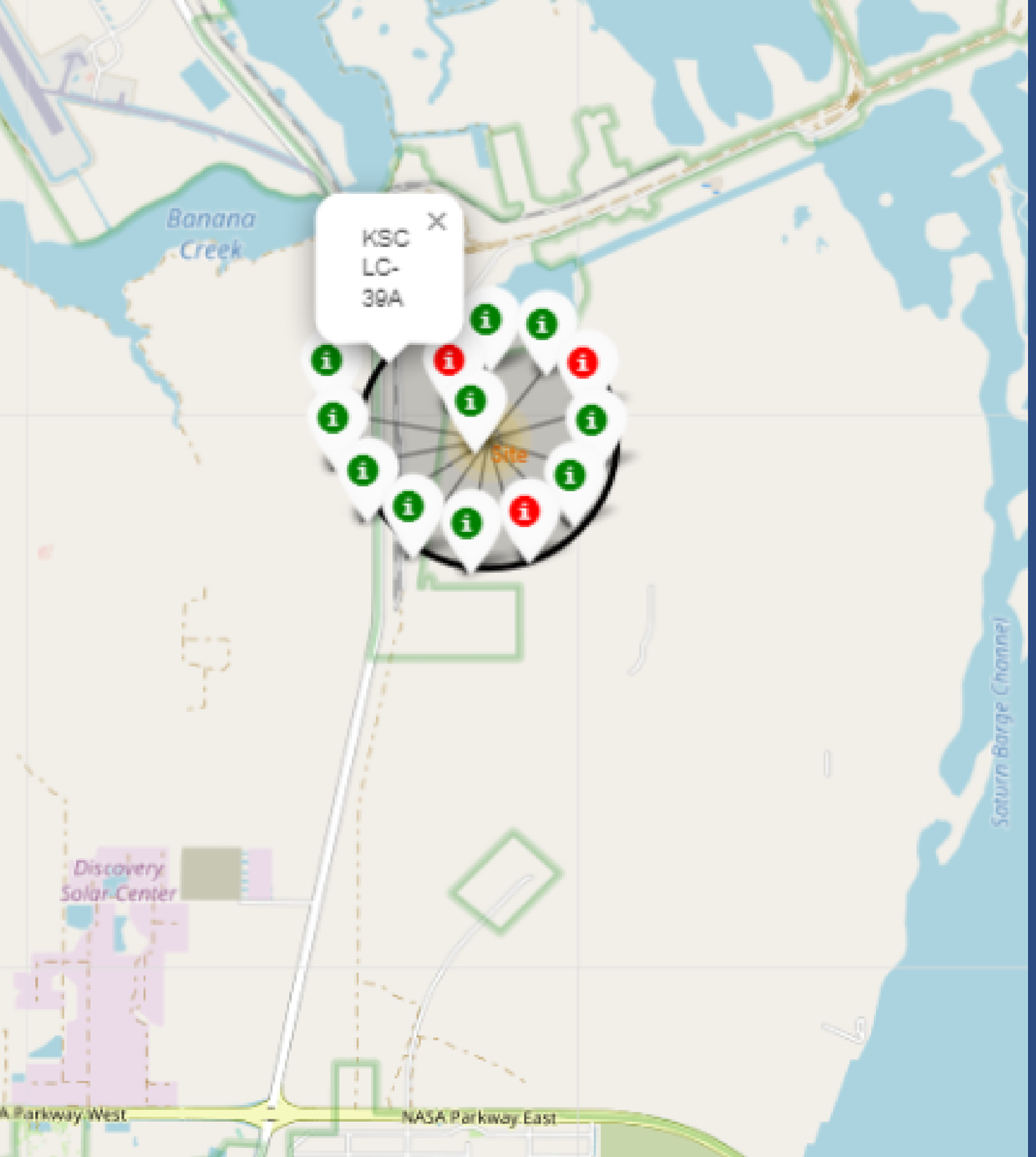












Highest success rate

Highest success rate is of lauch site KSC LC-39A

click on icon to view notebook

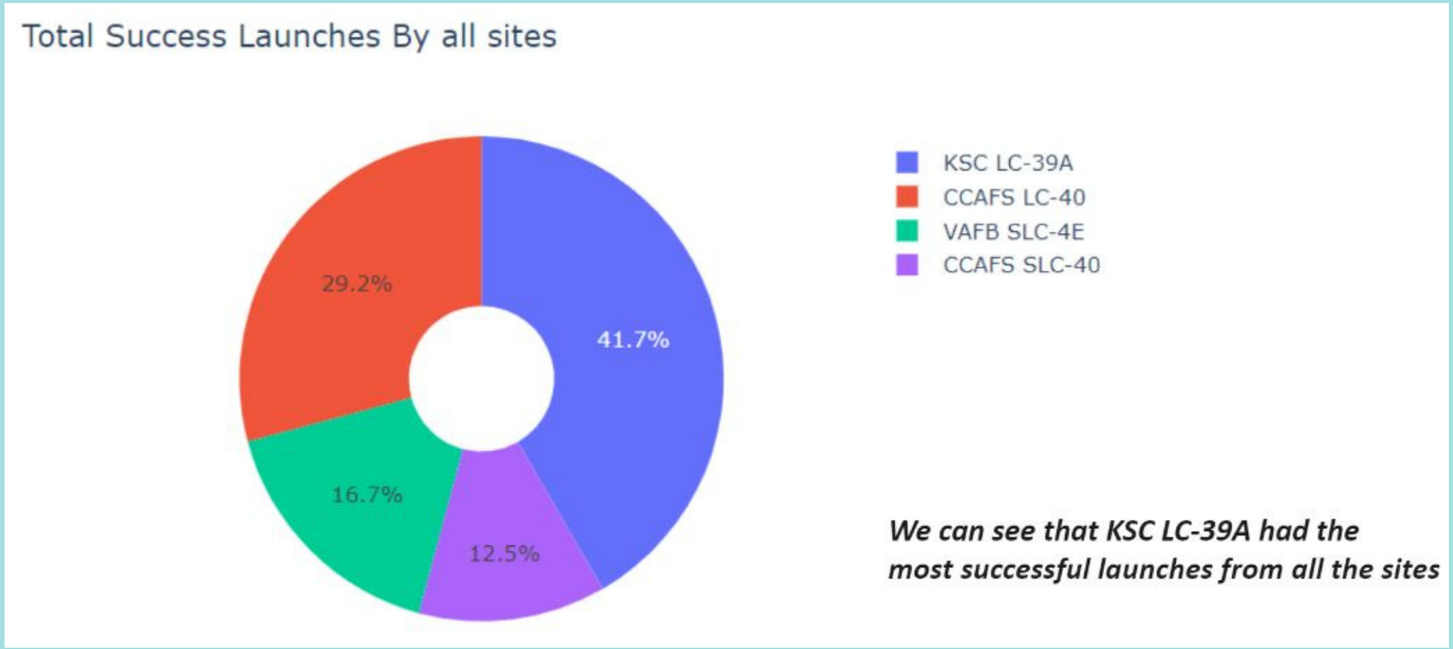
2.Insights From Interative Visuals

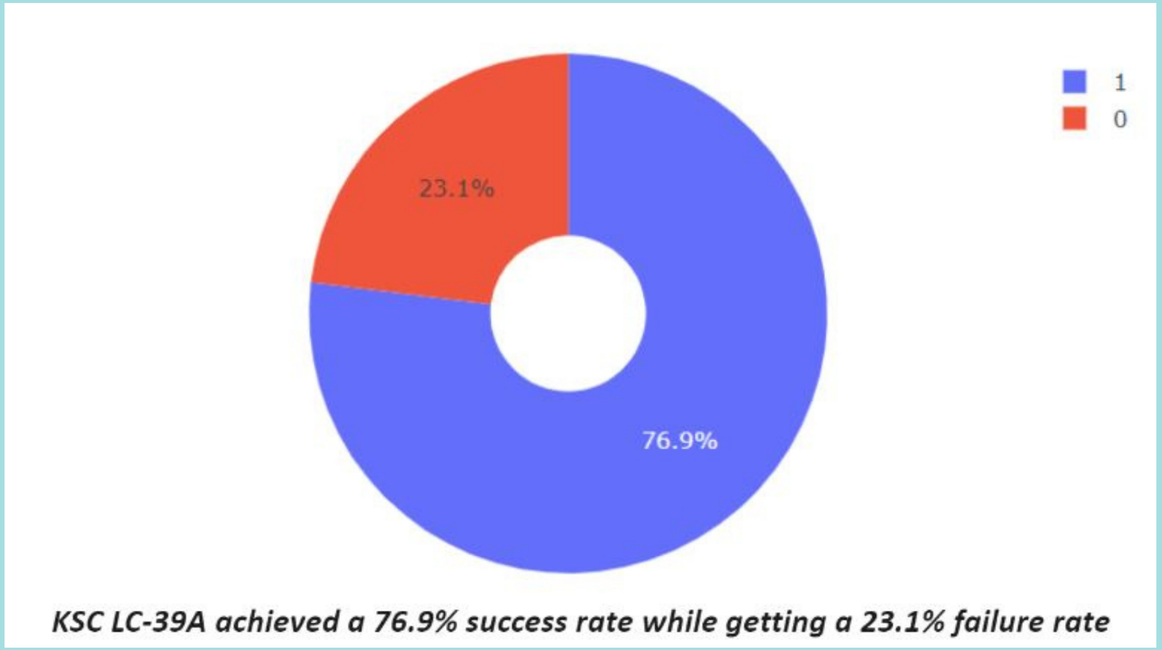
USING PLOTLY DASH APPLICATION

What factors contribute to the success of a Falcon 9 first stage landing?

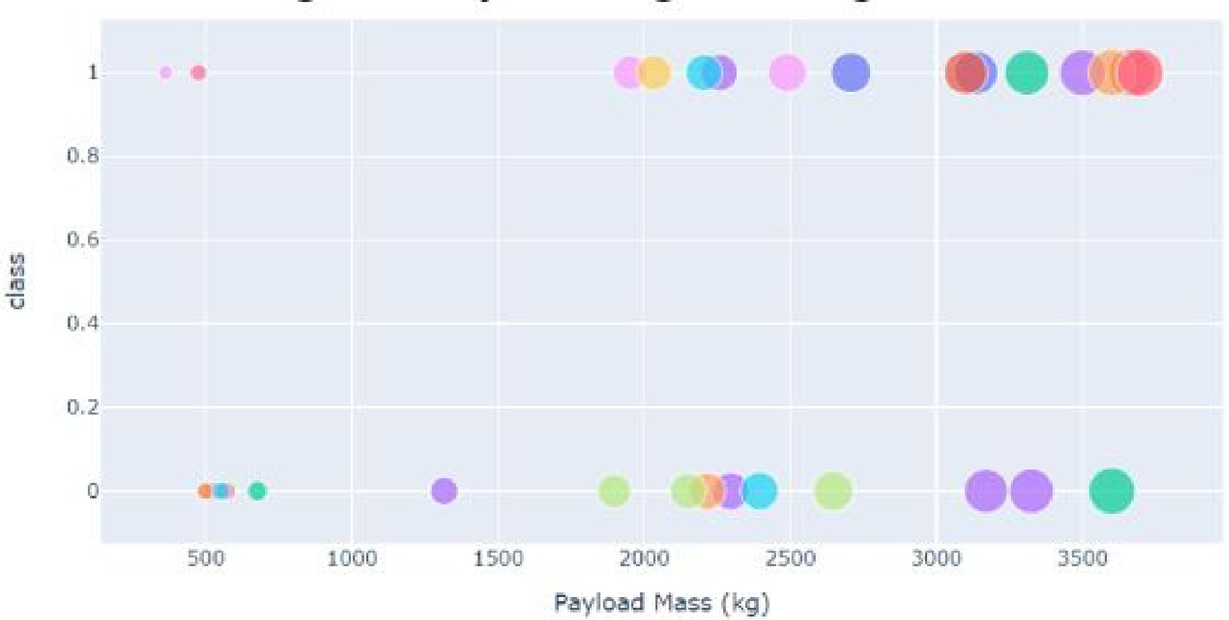
Which Site has more success rate?

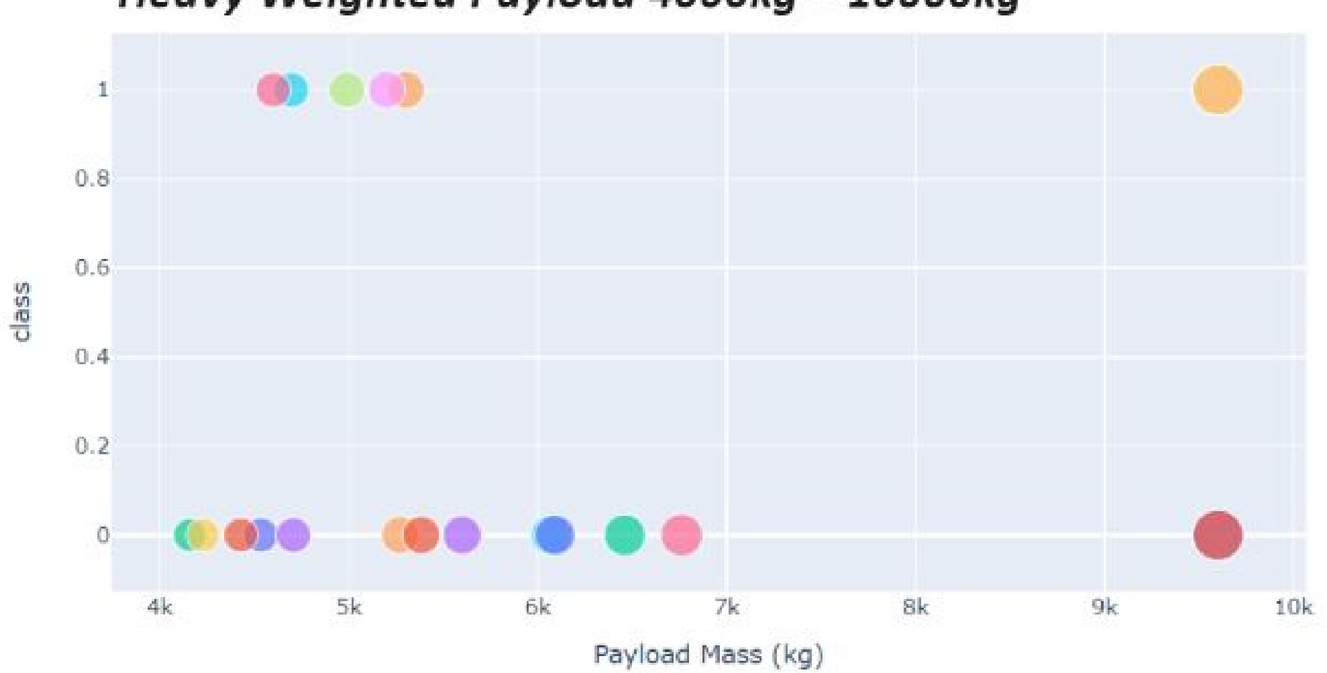
What is the effect of payload mass on success rate?





Highest success rate is of lauch site KSC LC-39A 76.9%





Class 0 Success
Class1 Failure

Success rate is higher for heavy weight payloads

click on icon to view notebook

ML Models Used

1. Logistic Regression
2. Support Vector Machine
3. Decision Tree
4. K-Nearest Neighbour

TASK 1

Create a NumPy array from the column `Class` in `data` , by applying the method `to_numpy()` then assign it to the variable `Y` ,make sure the output is a Pandas series (only one bracket df['name of column']).

```
Y = data['Class']
Y = Y.to_numpy()
```

TASK 2

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

```
# students get this
transform = preprocessing.StandardScaler().fit(X).transform(X)
transform[0:5]
```

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

In [16]:
```
X_train, X_test, Y_train, Y_test = train_test_split( transform, Y, test_size=0.2, random_state=2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

```
Train set: (72, 83) (72,)
Test set: (18, 83) (18,)
```

we can see we only have 18 test samples.

In [17]:
```
Y_test.shape
```

Out[17]:
```
(18,)
```

TASK 12

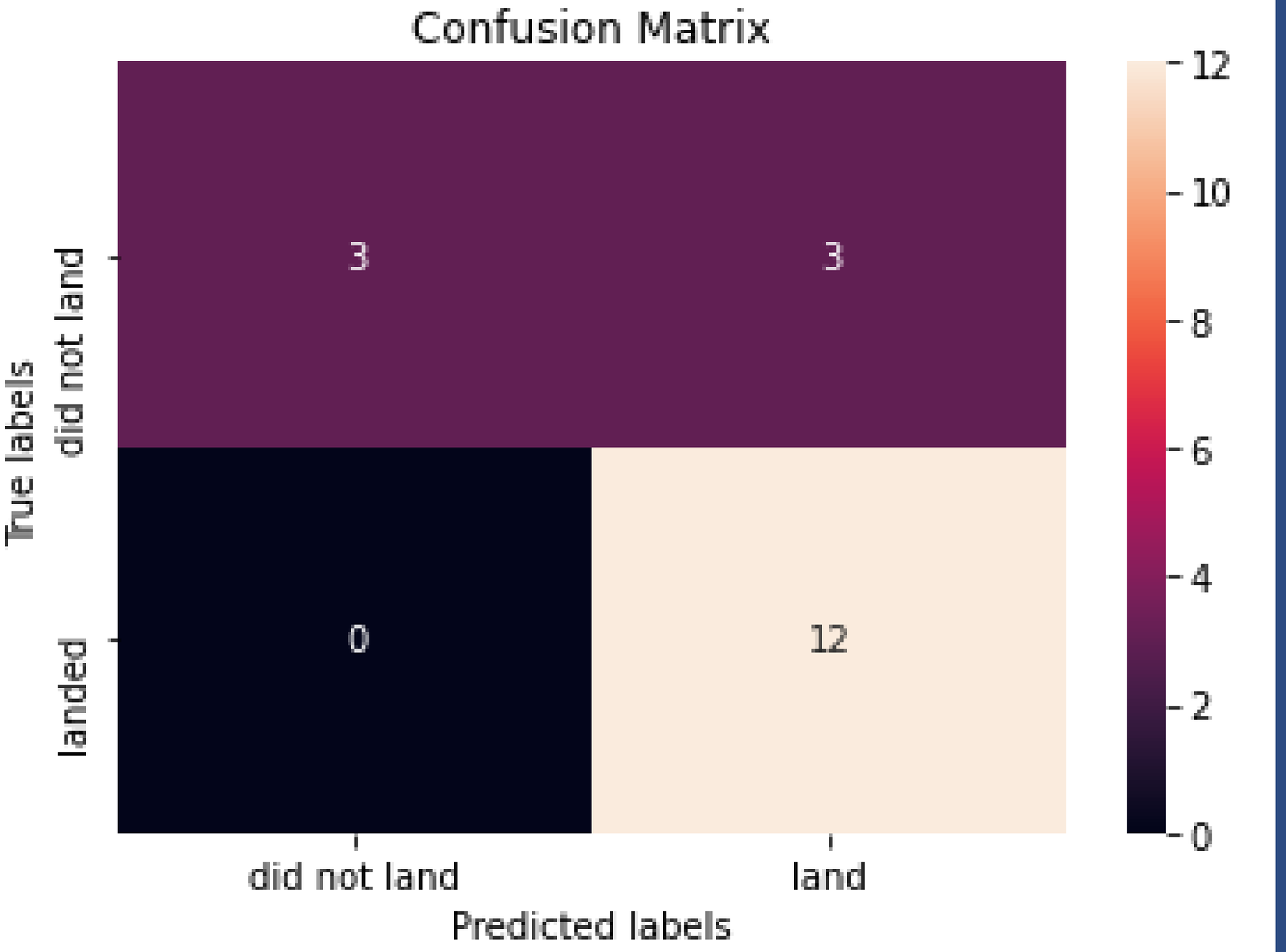
Find the method performs best:

In [69]:
```
val = {'Logistic Regression':logreg_cv.best_score_,'SVM':svm_cv.best_score_,
          'Decision Tree':tree_cv.best_score_,'KNN':knn_cv.best_score_}

accuracy = max(val.values())*100
Algo = max(val, key=val.get)
print("The Best Algorithm is:",Algo,"And the Accuracy is:",accuracy,"%")
```

```
The Best Algorithm is: Decision Tree And the Accuracy is: 87.5 %
```

After ML pipeline and grid search.The Best Performing model is Decision Tree



Confusion Matrix

The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier

click on icon to view notebook

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.

Tanaeem Ahmad

Thank you!

Email:tanaeemdar02@gmail.com