IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Andrea Sorbello
15 Jan 2023

# Outline

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

# Executive Summary

- Summary of methodologies
  1. Data Collection through API and Web Scraping
  2. Data Wrangling
  3. Exploratory Data Analysis with SQL and Data Visualization
  4. Interactive Visual Analytics with Folium
  5. Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

# Introduction

## Project background and context

We will predict if the Falcon 9 first stage will land successfully. SpaceX 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 SpaceX 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 SpaceX for a rocket launch.

## Problems you want to find answers

- What factors determine if the rocket will land successfully?
- The relationship between the various features that determine the success rate of a successful landing.
- What conditions need to be implemented to ensure the best result
- how to determine the price of each launch?

Section 1

# Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - API from SpaceX website
  - WebScraping from 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

Data collection is **the process of collecting, measuring and analyzing different types of information using a set of standard validated techniques**. The main objective of data collection is to gather information-rich and reliable data, and analyze them to make critical business decisions.

The data was collected using various methods:

### API

1. Getting data from API create a dataframe from this data
2. Then cleaned the data, checked for missing value and fill missing values where necessary
3. Filter the dataframe to only  include Falcon 9 launches

### WEBSCRAPING

1. We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
2. We parsed the table and converted it into a pandas dataframe

# Data Collection – SpaceX API

Getting Response from API

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

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

Converting response to a .json file

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

```
static_json_url='https://cf-courses-data.s3.us.cloud-
```

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

Apply custom functios to clean data

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

```
data_falcon9.isnull().sum()
```

Assign list to dictionary for dataframe

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
```

Filter only falcon 9 and export file (.csv)

```
data_falcon9= data[data.BoosterVersion =='Falcon 9']
data_falcon9
```

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

| FlightNumber | Date | BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome | Flights | GridFins | Reused | Legs | LandingPad | Block | ReusedCount | Serial |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2010-06-04 | Falcon 9 | NaN | LEO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | 0 | B0003 |
| 2 | 2012-05-22 | Falcon 9 | 525.0 | LEO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | 0 | B0005 |
| 3 | 2013-03-01 | Falcon 9 | 677.0 | ISS | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | 0 | B0007 |
| 4 | 2013-09-29 | Falcon 9 | 500.0 | PO | VAFB SLC 4E | False Ocean | 1 | False | False | False | None | 1.0 | 0 | B1003 |
| 5 | 2013-12-03 | Falcon 9 | 3170.0 | GTO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | 0 | B1004 |

Notebook:

[Data Collection with API]

# Data Collection – Scraping

**Getting Response from HTML**

```
static_url = "https://en.wikipedia.org
# assign the response to a object

response= requests.get(static_url)
```

**Creating BeautifulSoup**

```
soup = BeautifulSoup(response.text, 'html.parser')
soup
```

**Finding Tables and extract columns names**

```
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')

column_names = []

labels = first_launch_table.find_all('th')
for label in labels:
    name = extract_column_from_header(label)
    # header = str(label.text).strip()
     #header = str(header).split("($)Footnote", 1)[0]
     if name != None:
         if len(name) > 0:
             column_names.append(name)
```

**Create an empty 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'] = []
```

**Converting dictionary to daframe**

```
headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if  ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

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

**Export dataframe (.csv)**

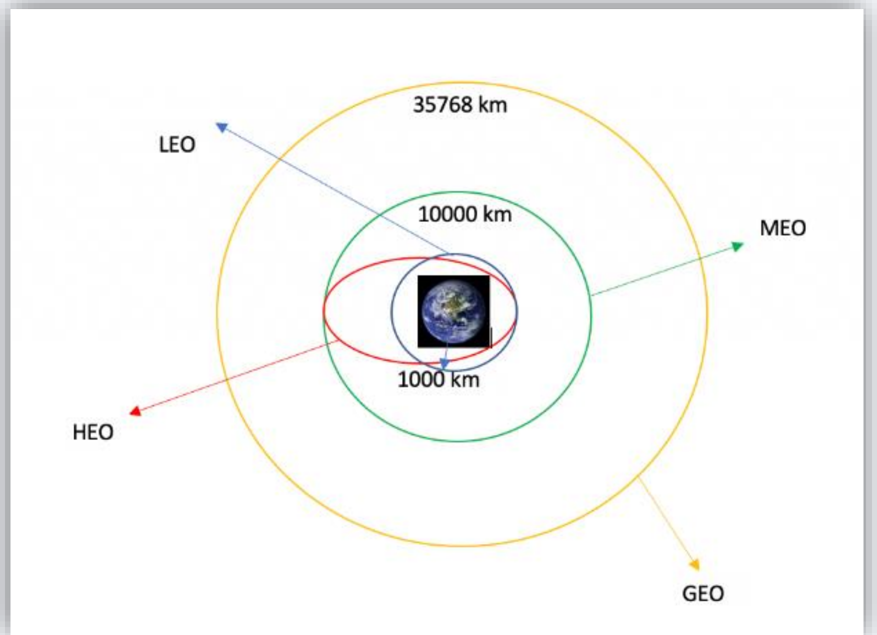
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Notebook:
[Data collection WebScraping]

# Data Wrangling

Data wrangling is **the process of removing errors and combining complex data sets to make them more accessible and easier to analyze**. You need to present your data wrangling process using key phrases and flowcharts

1. I did some exploratory data analysis (EDA) to find some patterns in the data and determine what the label would be for training supervised models.
2. Then I convert these results into training labels with **1** meaning the booster arrived successfully **0** meaning it was unsuccessful
3. I create a landing outcome label from **Outcome** column and
4. exported the results to **csv.**




Notebook:
[Data Wrangling]()

# Data Wrangling

Calculate the number and occurrence of each orbit

Calculate the number and occurence of mission outcome per orbit type

Create a landing outcome label from Outcome column

```
# Apply value_counts on Orbit column
df.value_counts("Orbit")
```

```
Orbit
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
GEO       1
HEO       1
SO        1
dtype: int64
```

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

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

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

| | Class |
|---|---|
| **0** | 0 |
| **1** | 0 |
| **2** | 0 |
| **3** | 0 |
| **4** | 0 |
| **5** | 0 |
| **6** | 1 |
| **7** | 1 |

Notebook:

[Data Wrangling]

Section 2

# Insights drawn from EDA

# EDA with Data Visualization

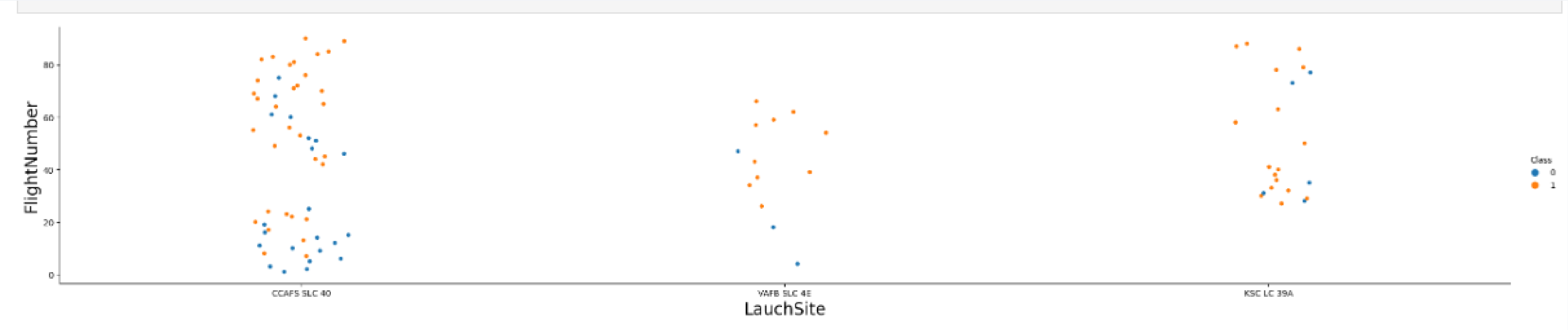
Perform exploratory Data Analysis and Feature Engineering using Pandas and Matplotlib

Next, let's drill down to each site visualize its detailed launch records.

Notebook:
EDA with data visualization

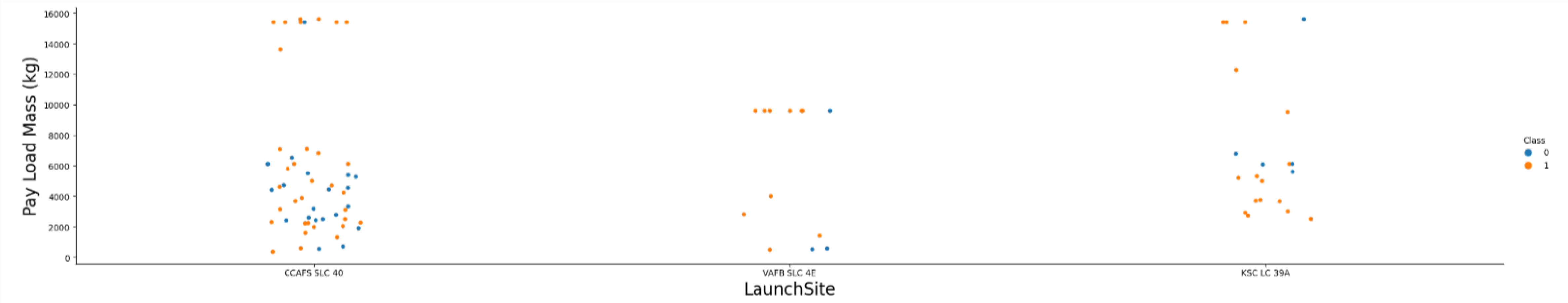
# Flight Number vs Lauch Site

- **CCAFS LC-40** has a success rate of 60 %,
- **KSC LC-39A** has a success rate of 77%,
- **VAFB SLC 4E** has a success rate of 77%.



# Payload and Launch Site

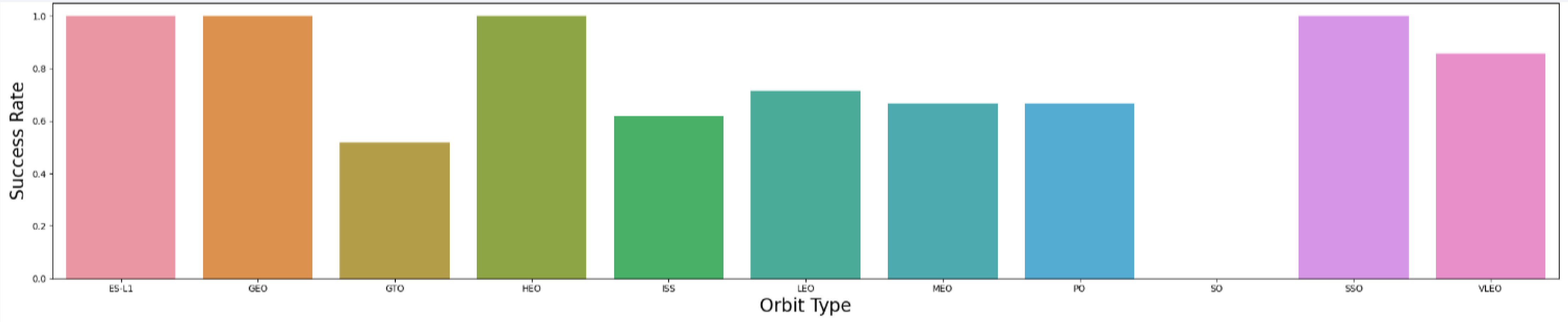
Now if you observe Payload Vs. Launch Site scatter point chart you will find for the **VAFB-SLC** launchsite there are no rockets launched for heavypayload mass(greater than 10000



# Success rate of each orbit type

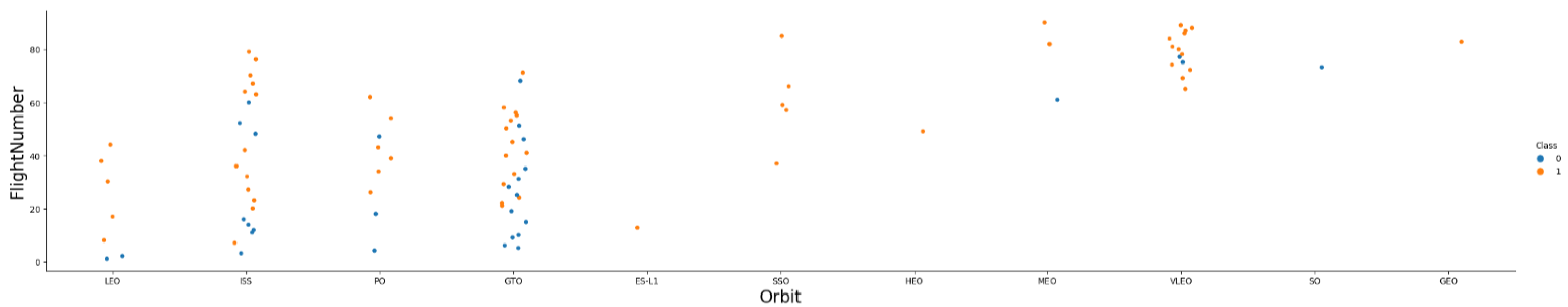
Analyze the ploted bar chart try to find which orbits have high sucess rate.

- GEO
- HEO
- SSO
- ES-L1



# FlightNumber and Orbit type

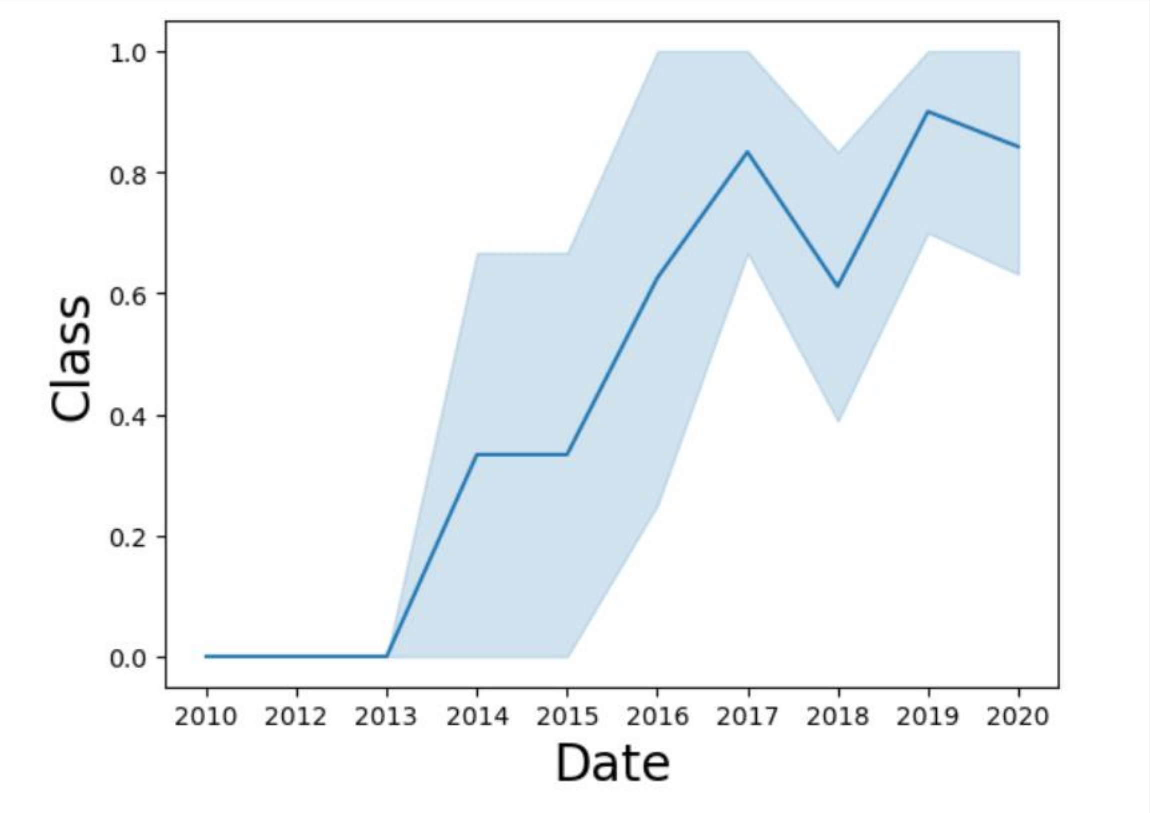
You should see that in the **LEO** orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in **GTO** orbit..



# Visualize the launch success yearly trend

I draw a line graph with the x-axis as the Year and the y-axis as the average success rate, to get the average launch success trend.

We can observe that the sucess rate since 2013 kept increasing till 2020



# EDA with SQL

**SQL (Structured Query Language)** is a standard language used for managing relational databases.

With SQL, you **can create, modify, and query** tables within a database, as well as manage users and permissions.

It also allows you to **select, insert, update and delete** data in a database.

SQL is a declarative language, which means that the user specifies what needs to be done, but not how it needs to be done.

Notebook:
[EDA with SQL]

# 1. Display the names of the unique launch sites in the space mission

RENAME TABLE  from **SPACE_X** to **x**

```
%sql select distinct launch_site from x
```

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

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

```
%sql SELECT * FROM x WHERE launch_site LIKE 'CCA%' LIMIT 5;
```

| 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 |

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

```
%sql SELECT customer,sum(payload_mass__kg_) as total_payload FROM x WHERE CUSTOMER = 'NASA (CRS)' group by CUSTOMER;
```

| customer | total_payload |
|---|---|
| NASA (CRS) | 45596 |

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

```
%sql select booster_version,avg(payload_mass__kg_) as average_payload_mass from x where booster_version='F9 v1.1' group by booster_version;
```

| booster_version | average_payload_mass |
|---|---|
| F9 v1.1 | 2928 |

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

```
%sql select DATE,landing__outcome from x where landing__outcome like 'Success (ground pad)' limit 1
```

| DATE | landing__outcome |
|---|---|
| 22-12-2015 | Success (ground pad) |

# 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 booster_version from x where landing__outcome
like 'Success (drone ship)' and payload_mass__kg_>4000 and payload_mass__kg_<6000
```

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

# 7. List the total number of successful and failure mission outcomes

```
%sql SELECT mission_outcome, COUNT(*) as total FROM x GROUP BY mission_outcome;
```

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

# 8. List the names of the booster_versions which have carried the maximum payload mass

```
%sql select distinct(booster_version),payload_mass__kg_ from x where payload_mass__kg_ in (select max(payload_mass__kg_) from x)
```

| booster_version | payload_mass__kg_ |
|---|---|
| F9 B5 B1048.4 | 15600 |
| F9 B5 B1048.5 | 15600 |
| F9 B5 B1049.4 | 15600 |
| F9 B5 B1049.5 | 15600 |
| F9 B5 B1049.7 | 15600 |
| F9 B5 B1051.3 | 15600 |
| F9 B5 B1051.4 | 15600 |
| F9 B5 B1051.6 | 15600 |
| F9 B5 B1056.4 | 15600 |
| F9 B5 B1058.3 | 15600 |
| F9 B5 B1060.2 | 15600 |
| F9 B5 B1060.3 | 15600 |

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

```
%sql select landing__outcome,booster_version,launch_site from x where landing__outcome= 'Failure (drone ship)' and date like '%2015'
```

| landing__outcome | booster_version | launch_site |
|---|---|---|
| Failure (drone ship) | F9 v1.1 B1012 | CCAFS LC-40 |
| Failure (drone ship) | F9 v1.1 B1015 | CCAFS LC-40 |

# 10. Rank the count of landing outcomes (Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20

```
%sql select landing__outcome,count(*),date
from x
where landing__outcome='Failure (drone ship)' or landing__outcome='Success (ground pad)'
and date>= '04-06-2010' and date<='20-03-2017'
group by landing__outcome,date
```

| landing__outcome | 2 | DATE |
|---|---|---|
| Failure (drone ship) | 1 | 04-03-2016 |
| Success (ground pad) | 1 | 07-09-2017 |
| Success (ground pad) | 1 | 08-01-2018 |
| Failure (drone ship) | 1 | 10-01-2015 |
| Failure (drone ship) | 1 | 14-04-2015 |
| Success (ground pad) | 1 | 14-08-2017 |
| Failure (drone ship) | 1 | 15-06-2016 |
| Success (ground pad) | 1 | 15-12-2017 |
| Failure (drone ship) | 1 | 17-01-2016 |
| Success (ground pad) | 1 | 18-07-2016 |
| Success (ground pad) | 1 | 19-02-2017 |

Section 3

# Launch Sites Proximities Analysis

# Build an Interactive Map with Folium

The launch success rate may depend on many factors such as payload mass, orbit type, and so on.

It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories.

Finding an optimal location for building a launch site certainly involves many factors and hopefully we could discover some of the factors by analyzing the existing launch site locations.

Notebook:
[Visual analitic with Folium]

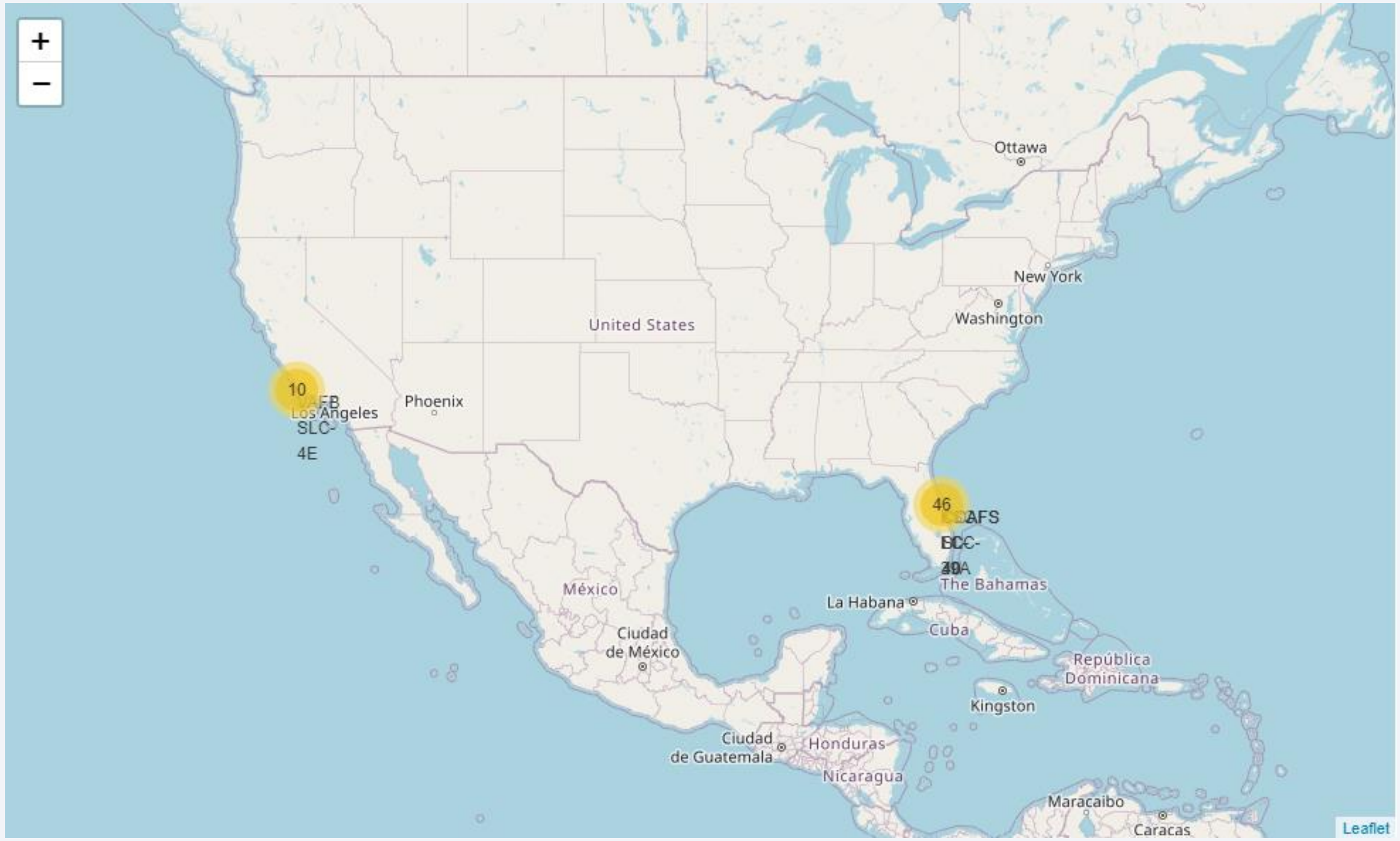
# Launch sites global map

We can see that all launch sites are located in the united states, close to the coast and approximately close to the equator.

- In partilucar are located in Floria and California

# Launch sites global map markers

We can see there are 10 lauch sites in California and 46 in Florida.

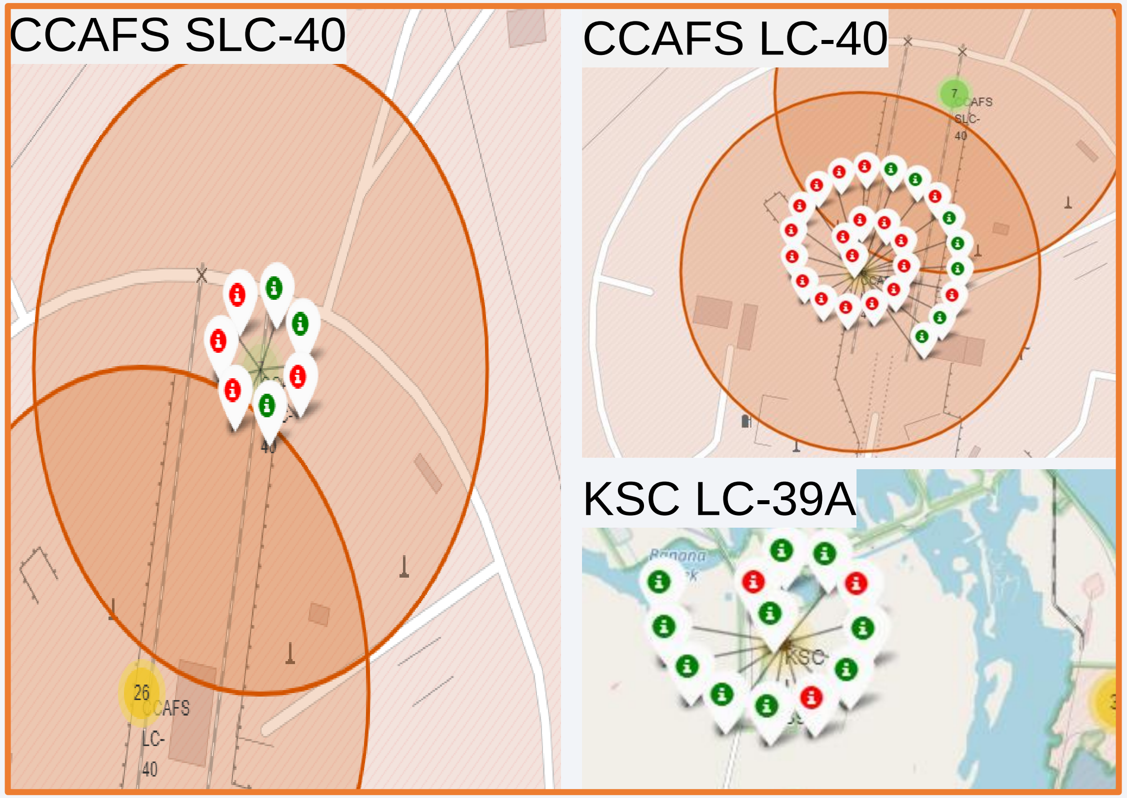


# Mark the success/failed launches for each site on the map

Green marker show successful launches and Red marker shows Failures

- **CCAFS LC-40**: 7 success launches of 26
- **CCAFS SLC-40**: 3 success launches of 7
- **VAFB SLC-4E**: 4 success launches of 10
- **KSC LC-39A**: 10 success launches of 13

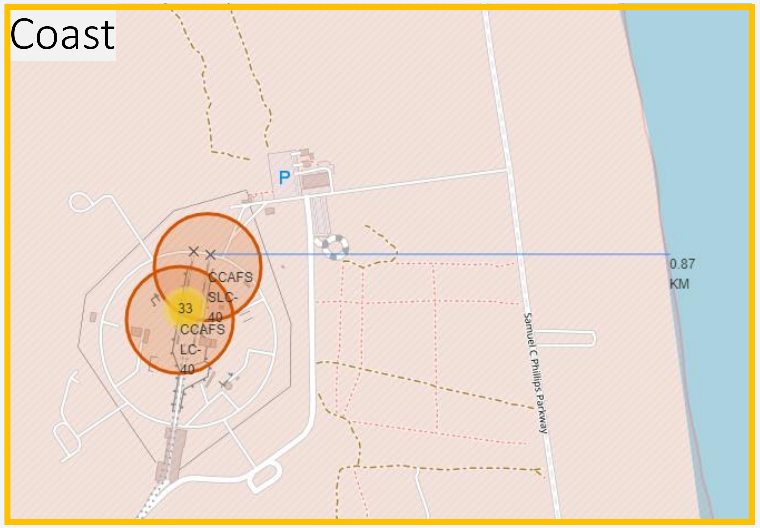

CALIFORNIA

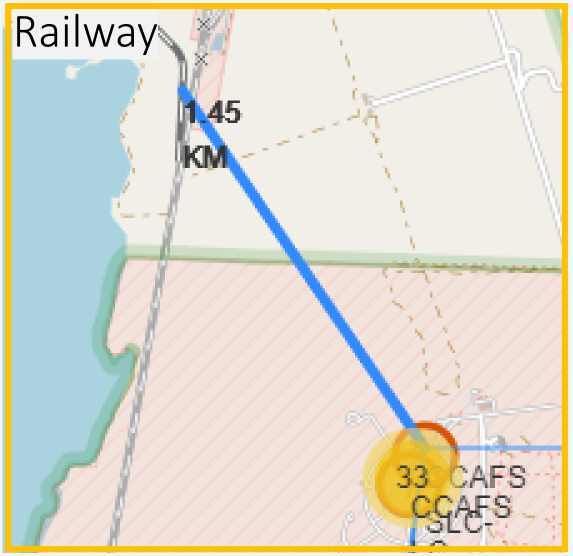



FLORIDA

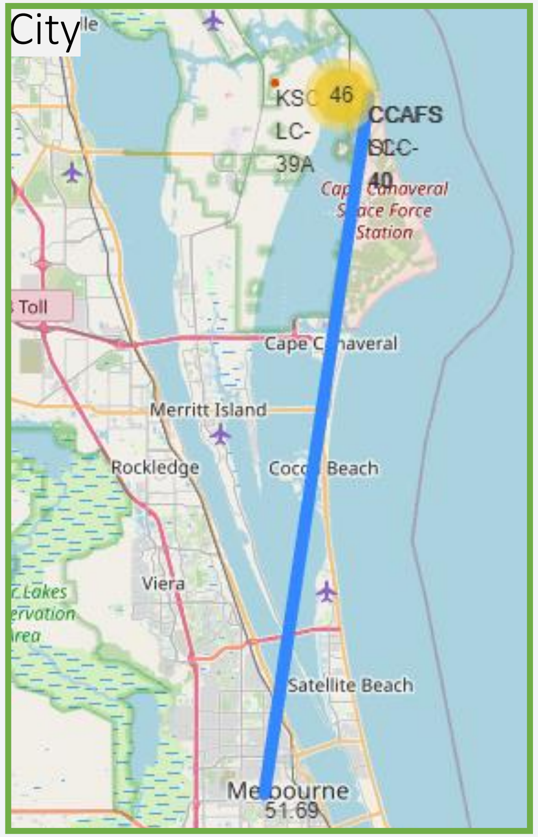
# Distances between a launch site to its proximities

- **Coast**: distance 0.87km
- **Railway**: distance 1.45 KM
- **Highway**: distance 11.16 KM
- **City**: distance 51.69 KM







Section 4

# Build a Dashboard with Plotly Dash

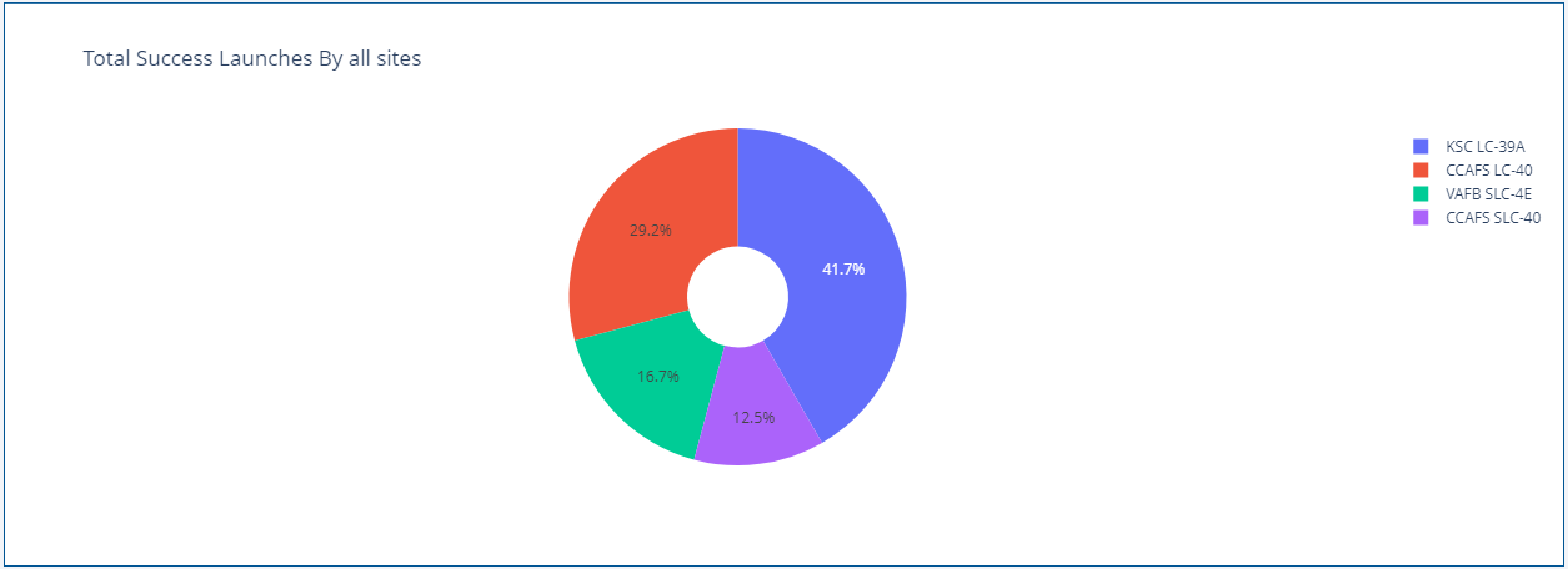
# Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

Notebook:
[Dash]

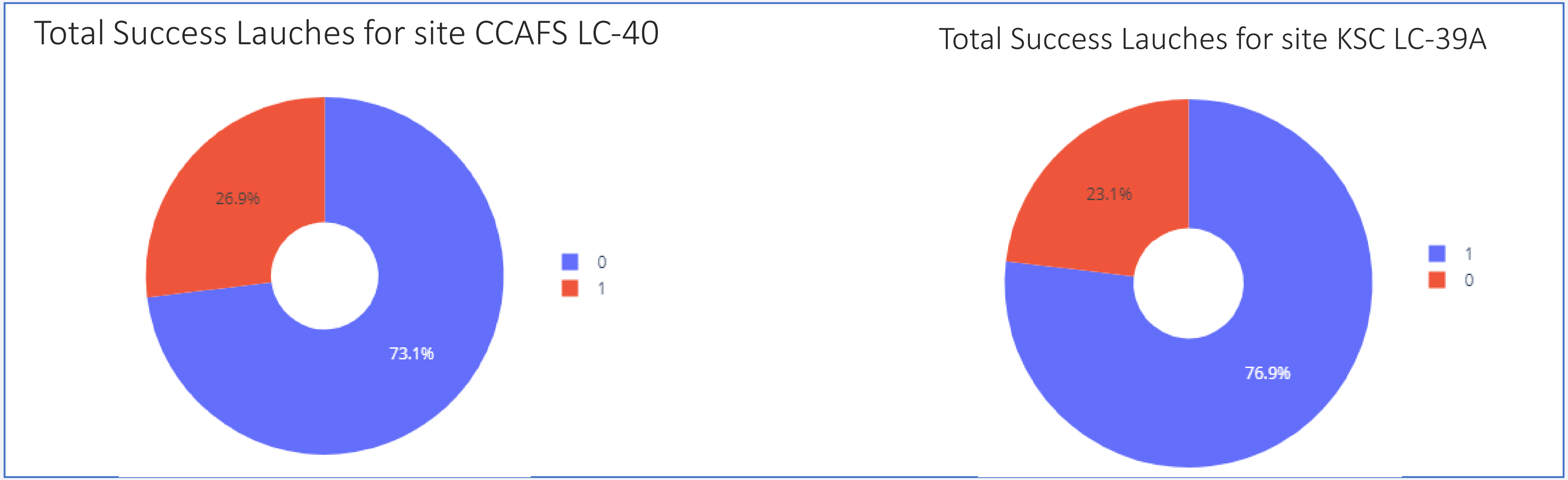
# Success rate by each launch site (Pie Chart)

KSC LC-39A is more successful than other launched locations



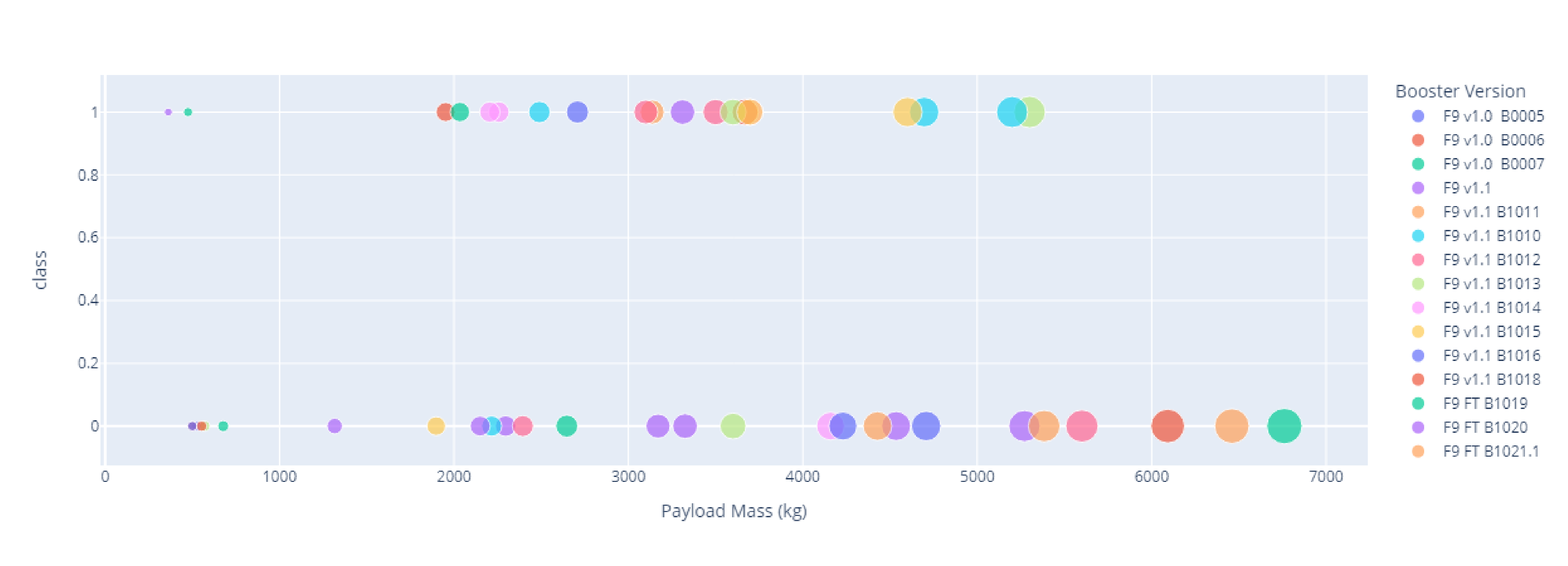
# Success or failure percentage (Pie chart)

- KSC LC-39A has 76.9% success rate and 23.1% failure rate
- CCAFS LC-40 has 26.9% success rate and 73.1 failure rate



# Payload vs Outcome for all booster version

The most successful launches center on boosters with weights ranging from 18,000 to 38,000



Section 5

# Predictive Analysis (Classification)

# Predictive Analysis (Classification)

We Perform exploratory Data Analysis and determine Training Labels

- create a column for the class and standardize the data
- Split into training data and test data
- Find best Hyperparameter for SVM, Classification Trees and Logistic Regression using GridSearchCV
- Find the method performs best using test data

Notebook:
[Machine Learning Prediction]

# Classification Accuracy

```
method = {'Logistic Regression':logreg_cv.best_score_,'SVM':svm_cv.best_score_,'Decision Tree':tree_cv.best_score_,'KNN':knn_cv.best_score_,}
method_df = pd.DataFrame.from_dict(algorithms, orient='index', columns=['Accuracy'])
method_df
```

| | Accuracy |
|---|---|
| **Logistic Regression** | 0.846429 |
| **SVM** | 0.848214 |
| **Decision Tree** | 0.875000 |
| **KNN** | 0.848214 |

The method performs best is **Decision Tree** with a score: **0.875**

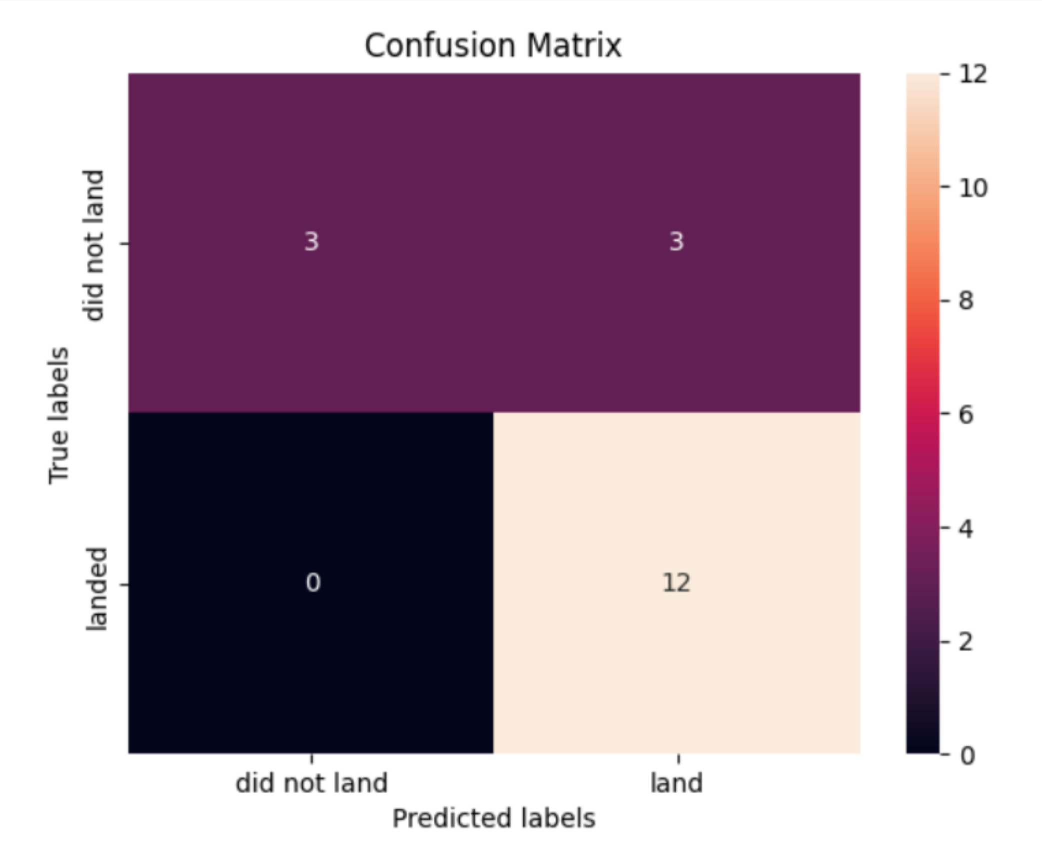
## Decision Tree with the best parameters

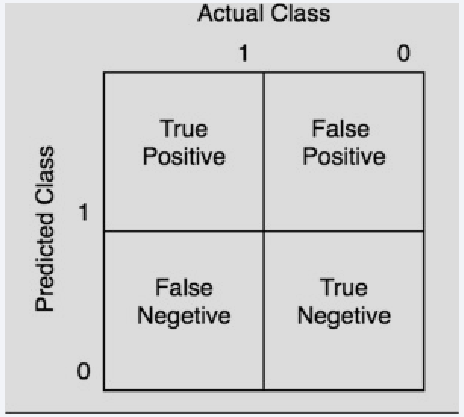
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10,
'splitter': 'random'}
accuracy : 0.875
```

# Confusion Matrix

- By examining the confusion matrix, we can see that the **Decision Tree** knows how to distinguish between the different classes. positive.
- The main problem is **false positives**, i.e. failed landing marked as successful landing by the classifier.



# Conclusion

On the basis of all that has been analysed, we can conclude by saying that:

- The **Decision Tree Classifier** is the best for Machine Learning for this dataset
- Ranging from 18,000 to 38,000 **Weighted payloads** perform better than the other payloads
- The launch success rate started to increase in 2013 until 2020
- We can see that **KSC LC-39A** had the most successful launches from all the sites
- Orbit **GEO,HEO,SSO,ES-L1** had the most Success Rate

# Thank you!