



IBM Developer  
SKILLS NETWORK

# Winning Space Race with Data Science

Volha Beliai  
August 13<sup>th</sup>, 2022



# Outline

---

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion



# Executive Summary

---

- **Summary of methodologies:**

- Data collection
- Data wrangling
- EDA with data visualization
- EDA with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly Dash - Predictive analysis (Classification)

- **Summary of all results:**

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

# Introduction

---

- **Project background and context**

***SpaceX** advertises **Falcon 9 rocket launches** on its website with a cost of 62 million dollars; other providers cost upwards 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 compete with SpaceX for a rocket launch.*

- **Problems we need to find answers for**

- What influences if the rocket will land successfully?
- How different relationships of certain rocket variables will impact the landing success?
- What conditions does SpaceX have to achieve to get the best rocket launch results?
- Will SpaceX reuse the first stage?



Section 1

# Methodology

# Methodology

---

## Executive Summary

- Data collection methodology:
  - *SpaceX REST API using the endpoint “api.spacexdata.com/v4/launches/past”*
  - *Web scraping related Wiki pages*
- Perform data wrangling
  - *One Hot Encoding data fields for Machine Learning and dropping irrelevant columns*
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Plotting : Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data.*
- 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 has been collected using below means:**

- ❖ **SpaceX Rest API** – Open Source REST API for launch, rocket, core, capsule, starlink, launchpad, and landing pad data.
- ❖ **Web Scrapping from Wikipedia:** using the Python BeautifulSoup package to web scrape some HTML tables that contain valuable Falcon 9 launch records.



# Data Collection – SpaceX API

- Request rocket launch data from SpaceX API:



- Request and parse the SpaceX launch data using the GET request:

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

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

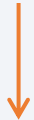
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```



- Decode the response content as a turn it into a Pandas dataframe:

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

- Extract necessary data from columns "rocket", "payloads", "launchpad" and "cores" and construct a new dataset:



```
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}
```



# Data Collection – SpaceX API

- Create a Pandas data frame from the dictionary launch\_dict: `df = pd.DataFrame.from_dict(launch_dict)`

- Filter the dataframe to only include Falcon 9 launches: `data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']`

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

**New DataFrame will look like below**

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-80.577366	28.561857
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-80.577366	28.561857
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003	-120.610829	34.632093
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-80.577366	28.561857

[My GitHub Data Collection link](#)

# Data Collection - Scraping

## Objectives

[Web scrap Falcon 9 launch records with BeautifulSoup:](#)

- ✓ Extract a Falcon 9 launch records HTML table from [Wikipedia](#)
- ✓ Parse the table and convert it into a Pandas data frame

## 1. Get a response from HTML

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

## 2. Create a BeautifulSoup object

```
soup = BeautifulSoup(data, 'html.parser')
```

## 3. Find all tables

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

## 4. Get column names

```
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)
```

## 5. Create a dictionary

```
launch_dict = dict.fromkeys(column_names)
```

```
del launch_dict['Date and time ( )']
```

```
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'] = []
launch_dict['Version Booster'] = []
launch_dict['Booster landing'] = []
launch_dict['Date'] = []
launch_dict['Time'] = []
```

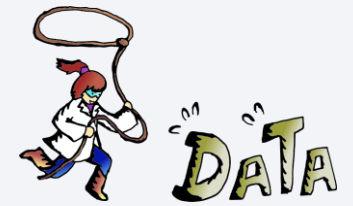
## 6. Append data keys

```
extracted_row = 0
for table_number, table in enumerate(soup.find_all('table', "wikitable plainrowheaders collapsible")):
    for rows in table.find_all("tr"):
        if rows.th:
            if rows.th.string:
                flight_number = rows.th.string.strip()
                flag = flight_number.isdigit()
            else:
                flag = False
            row = rows.find_all('td')
            if flag:
                extracted_row += 1
                date = datatimelist[0].strip(',')
                launch_dict[flight_number] = {}
                launch_dict[flight_number]['Date'] = date
```

## 7. Convert dictionary to dataframe and export df to .CSV

```
df = pd.DataFrame(launch_dict)
df.head()
df.to_csv('spacex_web_scraped.csv', index=False)
```

# Data Wrangling



## 1. Calculate the number of launches on each site

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

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

## 2. Calculate the number and occurrence of each orbit

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

```
GTO      0.300000
ISS      0.233333
VLEO     0.155556
PO       0.100000
LEO      0.077778
SSO      0.055556
MEO      0.033333
SO       0.011111
GEO      0.011111
HEO      0.011111
ES-L1    0.011111
Name: Orbit, dtype: float64
```

## 3. Calculate the number and occurrence of mission outcome per orbit type

```
landing_outcomes = df["Outcome"].value_counts()
bad_outcomes = set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

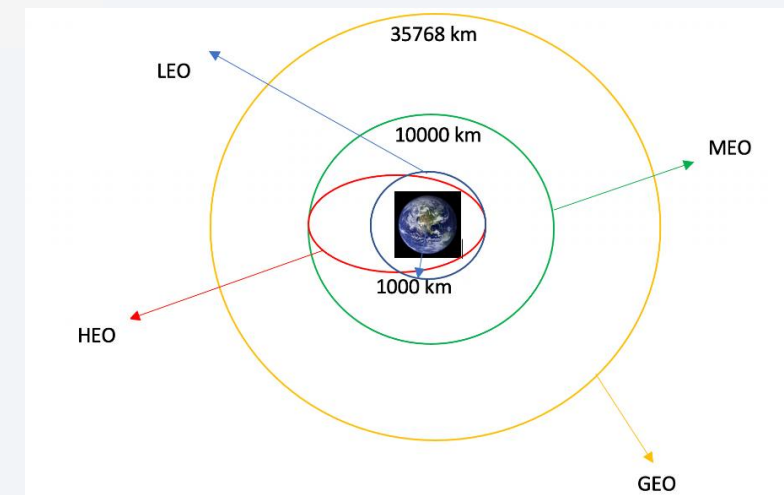
## 4. Create a landing outcome label from Outcome column

```
# 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)
    else:
        landing_class.append(1)
```

```
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

## ORBITS

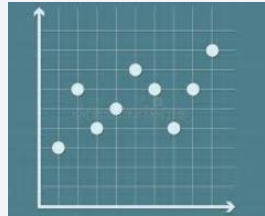


[My GutHub Data Wrangling link](#)

# EDA with Data Visualization

## Scatter Graphs being plotted:

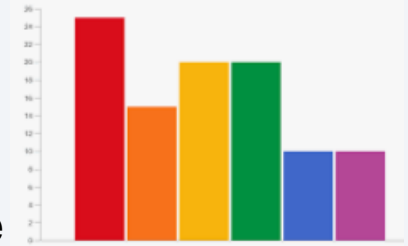
- Flight Number VS. Payload Mass
- Flight Number VS. Launch Site
- Payload VS. Launch Site
- Flight Number VS. Orbit
- Payload Mass VS. Orbit



Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation. Scatter plots usually consist of a large body of data.

## Bar Graph being plotted:

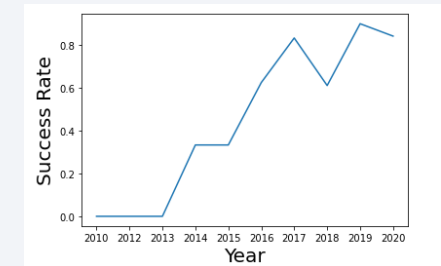
- Orbit Type VS. Success Rate



A bar diagram makes it easy to compare sets of data between different groups at a glance. The graph represents categories on one axis and a discrete value in the other. The goal is to show the relationship between the two axes. Bar charts can also show big changes in data over time.

## Line Graph being plotted:

- Success Rate VS. Year



Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded



# EDA with SQL

---

## SQL queries performed to gather information about the dataset:

- Displaying the names of the unique launch sites in the space mission .
- Displaying 5 records where launch sites begin with the string 'CCA'.
- Displaying the total payload mass carried by boosters launched by NASA (CRS).
- Displaying average payload mass carried by booster version F9 v1.1 .
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000.
- Listing the total number of successful and failure mission outcomes .
- Listing the names of the booster\_versions which have carried the maximum payload mass.
- Listing the records which will display the month names, failure landing\_outcomes in drone ship ,booster versions, launch\_site for the months in year 2015.
  - Ranking the count of successful landing\_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.



# Build an Interactive Map with Folium

---

To visualize the Launch Data into an interactive map we

- ❑ Took the Latitude and Longitude Coordinates at each launch site and added a **Circle Marker** around each launch site with a label of the name of the launch site.
- ❑ Created a **folium Map object**, with an initial center location to be NASA Johnson Space Center at Houston, Texas.
- ❑ Assigned the dataframe launch\_outcome to **classes 0 (if a launch failed) and 1 (for a successful launch)** with **Green** and **Red** markers on the map in a **MarkerCluster()**.
- ❑ Calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. Lines are drawn on the map to measure distance to landmarks.



# Predictive Analysis (Classification)

---

## **BUILDING A MODEL**

- Load our dataset into NumPy and Pandas
- Create a NumPy array from the column "Class" in data, by applying the method to\_numpy() then assign it to the variable Y
- Standardize the data in X then reassign it to the variable X
- Split our data into training and test data sets
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train our dataset.

## **EVALUATING MODEL**

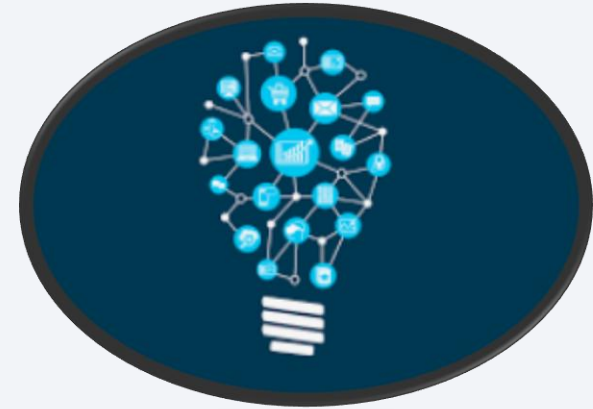
- Check the accuracy for each model
- Get tuned hyperparameters for each type of algorithms
- Plot Confusion Matrix

## **IMPROVING MODEL**

- Feature Engineering
- Algorithm Tuning FINDING THE BEST PERFORMING CLASSIFICATION

## **BEST MODEL**

- The model with the best accuracy score wins the best performing model



[My Prediction Analysis notebook url](#)

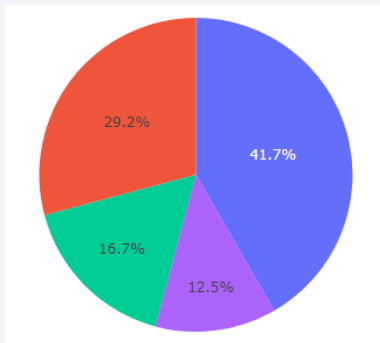
# Build a Dashboard with Plotly Dash

**Interactive visual analytics** enables users to explore and manipulate data in an interactive and real-time way. With interactive visual analytics, **users could find visual patterns faster and more effectively.**

Instead of presenting your findings in static graphs, interactive data visualization, or dashboarding, **can always tell a more appealing story.**

**Pie Chart** is showing the total count of success launches by a certain site/all sites.

It displays proportions of multiple classes of data depending on chosen parameters.

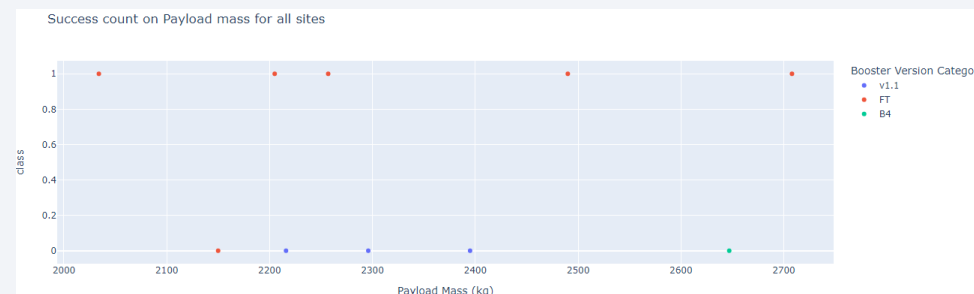


[My SpaceX Dash notebook url](#)

**Scatter Graph** built with Plotly Dash is showing relationship between Payload Mass (Kg) and Launch Outcome for different Booster Versions.

We can set the Payload Mass (Kg) parameter from 0 to 10k, choose all launch sites or one particular launch site and clearly observe the dependence.

It is the best method to show a non-linear pattern.





# Results

---

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





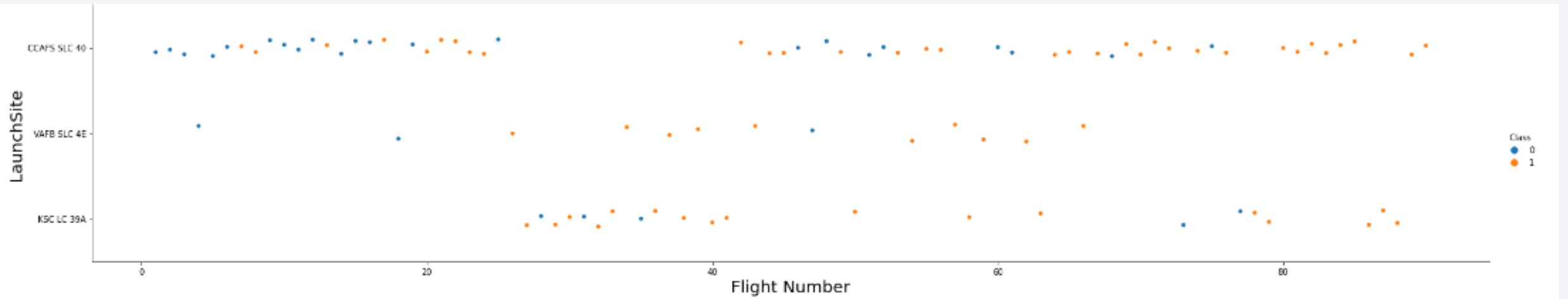
The background of the slide is an abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks in shades of red and cyan. A faint, light blue grid pattern is also visible, particularly in the upper right quadrant. The overall effect is dynamic and technological.

Section 2

# Insights drawn from EDA

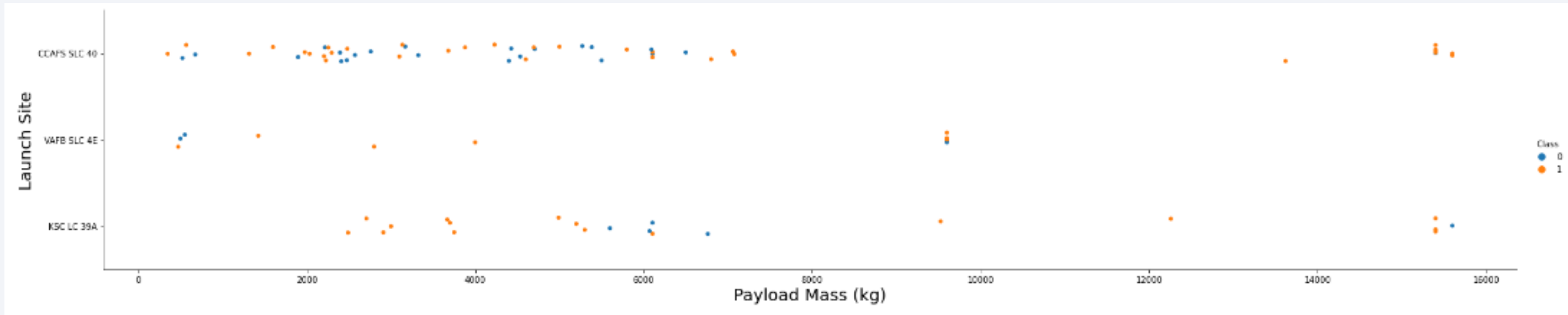


# Flight Number vs. Launch Site



With the increasing number of flights at a launch site the success rate is getting higher

# Payload vs. Launch Site

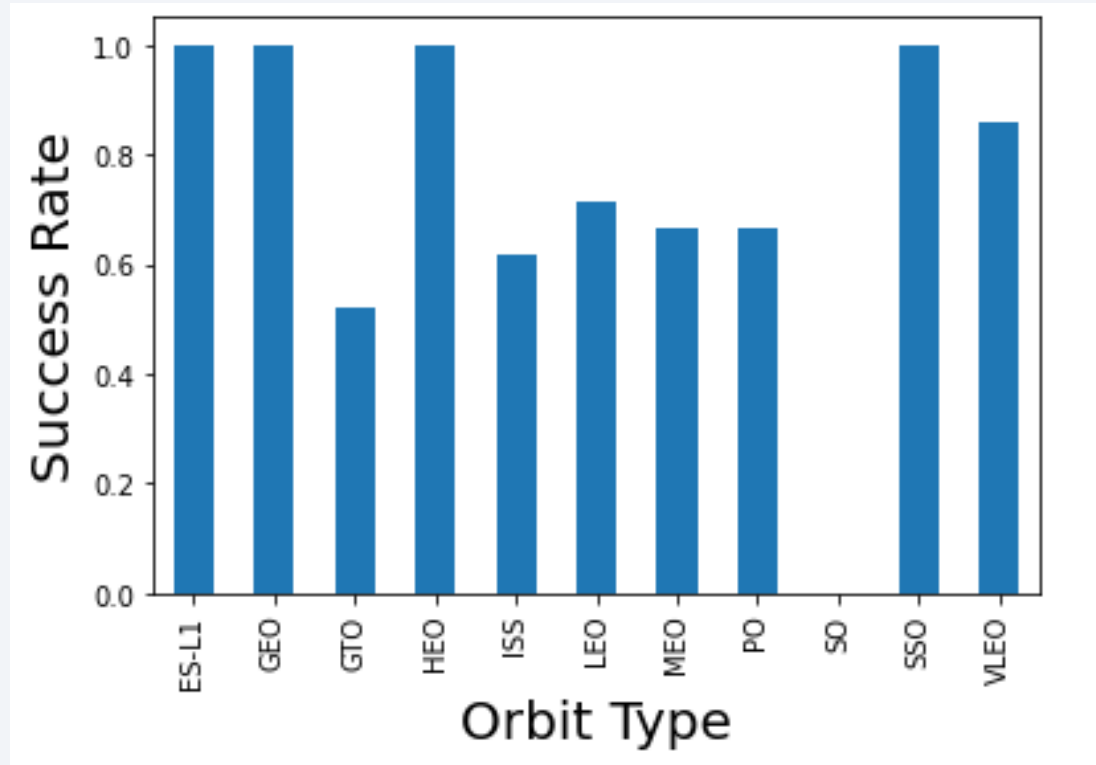


The bigger is the payload mass of a Rocket, the higher the launch success rate at a Launch Site. We can also observe that at VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).



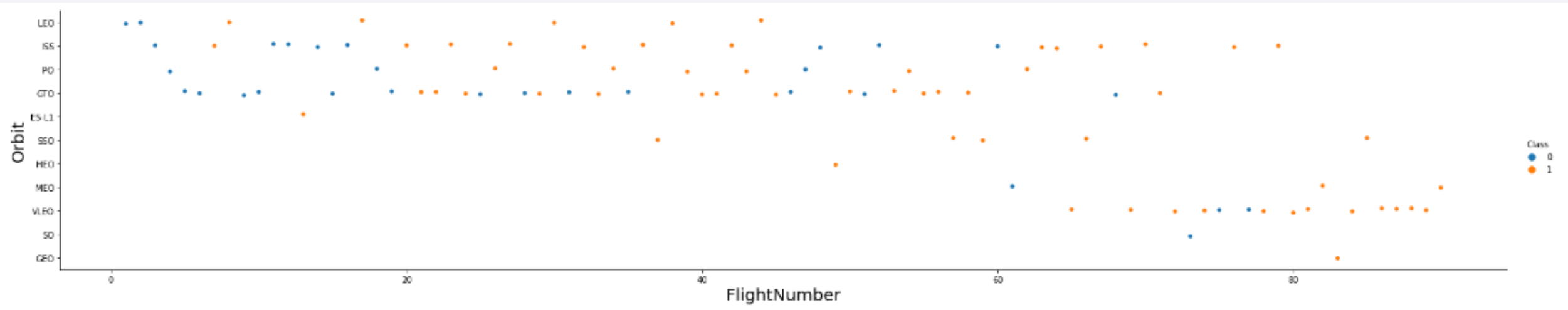
# Success Rate vs. Orbit Type

---



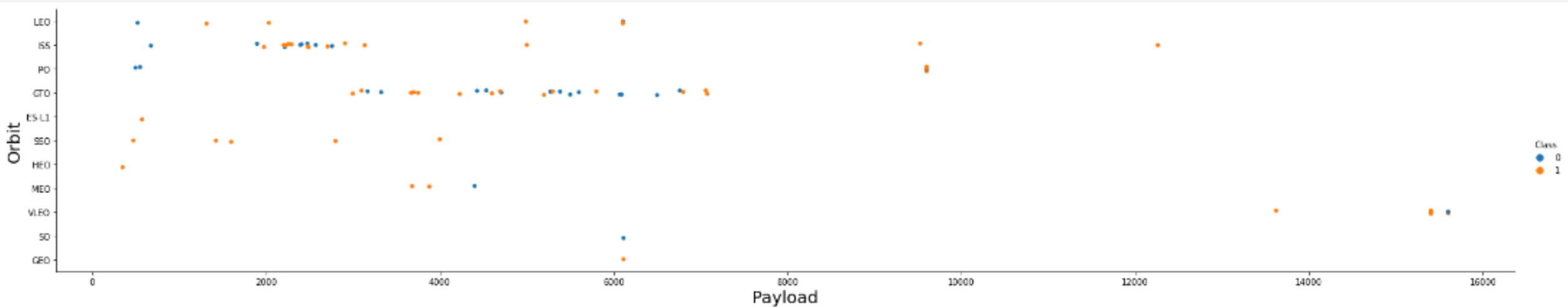
The highest success rate is at ES-L1, GEO, HEO and SSO Orbits.

# Flight Number vs. Orbit Type



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

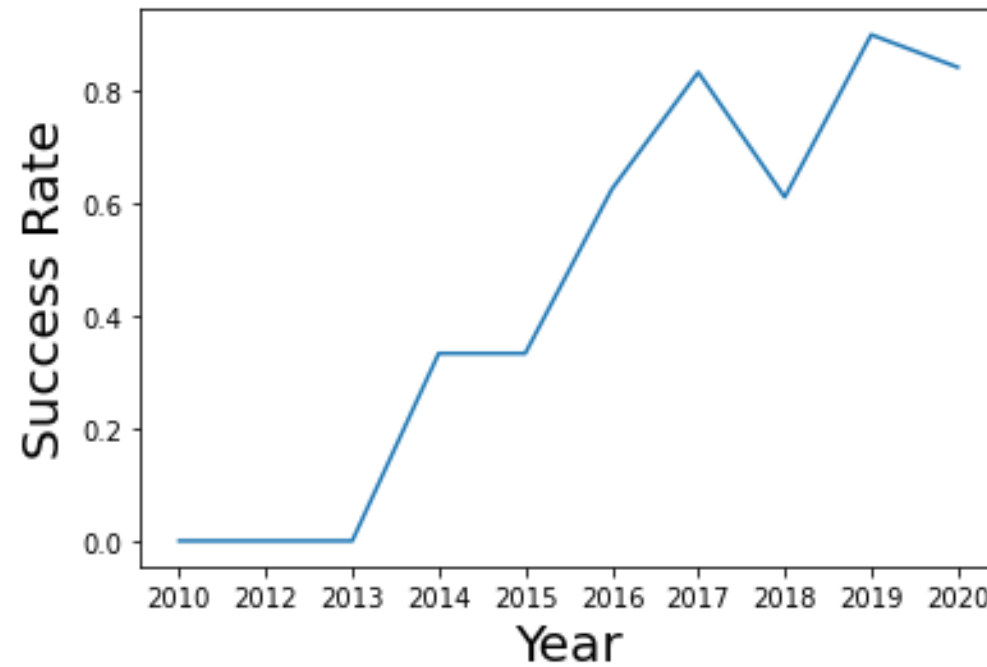
# Payload vs. Orbit Type



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 rate and negative landing (unsuccessful mission) are both here.

# Launch Success Yearly Trend

---



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



# All Launch Site Names

---

```
%sql select DISTINCT(LAUNCH_SITE) from SPACEXTBL;
```

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

**SQL “DISTINCT” query will show unique values in the column “Launch Site” from SPACEXTBL table:**



# Launch Site Names Begin with 'CCA'

---

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

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Using the word **LIMIT 5** in the query means that it will only show 5 records from SpaceX table.

**LIKE** keyword has a wild card with the words 'CCA%'. The percentage in the end suggests that the Launch\_Site name must start with CCA.

# Total Payload Mass

Calculate the total payload carried by boosters from NASA

```
%sql select sum(PAYLOAD_MASS_KG_) as payloadmass from SPACEXTBL where CUSTOMER ="NASA (CRS)";
```

payloadmass
45596

Function **SUM** summates the total in the column  
PAYLOAD\_MASS\_KG\_

The **WHERE** clause filters the dataset to only perform  
calculations on Customer **NASA (CRS)**



# Average Payload Mass by F9 v1.1

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



```
%sql select avg(PAYLOAD_MASS_KG_) from SPACEXTBL where BOOSTER_VERSION like "F9 v1.1%";
```

avg(PAYLOAD_MASS_KG_)
-----------------------

2534.6666666666665
--------------------

Function **AVG** calculates the average in the column **PAYLOAD\_MASS\_KG\_**

The **WHERE** clause filters the dataset to only perform calculations on **Booster\_version F9 v1.1**

# First Successful Ground Landing Date

---

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

```
%%sql
SELECT min(substr(Date,7,4) || substr(Date,4,2) || substr(Date,1,2)) from SPACEXTBL where "Landing _Outcome"="Success (ground pad)"
```

```
min(substr(Date,7,4) || substr(Date,4,2) || substr(Date,1,2))
```

```
20151222
```

Function **MIN** gives the minimum date in the column Date .

The **WHERE** clause filters the dataset to only perform calculations on Landing\_Outcome Success (drone ship).



# Successful Drone Ship Landing with Payload between 4000 and 6000

---

The names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
%%sql
select DISTINCT BOOSTER_VERSION from SPACEXTBL where "Landing_Outcome" ='Success (drone ship)' and PAYLOAD_MASS_KG_ BETWEEN 4000 and 6000;
```

## Booster\_Version

F9 FT B1022

F9 FT B1026

F9 FT B1021.2

F9 FT B1031.2

## Selecting only **Booster\_Version**

The **WHERE** clause filters the dataset to Landing\_Outcome = Success (drone ship)

The **AND** clause specifies additional filter conditions  
Payload\_MASS\_KG\_ > 4000 AND Payload\_MASS\_KG\_ < 6000

# Total Number of Successful and Failure Mission Outcomes

---

The total number of successful and failure mission outcomes

```
%sql SELECT MISSION_OUTCOME, COUNT(*) FROM SPACEXTBL GROUP BY MISSION_OUTCOME
```

Mission_Outcome	COUNT(*)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

**COUNT (\*)** function was used to get the number of successful and failure mission outcomes.

Then **GROUP BY** function was used to group the result by the mission outcome.

# Boosters Carried Maximum Payload

---

Names of the booster which have carried the maximum payload mass

```
%sql SELECT BOOSTER_VERSION,PAYLOAD_MASS_KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS_KG_ = (SELECT MAX(PAYLOAD_MASS_KG_) FROM SPACEXTBL);
```

Booster_Version	PAYLOAD_MASS_KG_
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

**Booster\_Version** and **Payload\_Mass\_Kg\_** columns have been selected from the SpaceX table.


The **WHERE** clause was used to set the **MAX** parameter for the **Payload\_Mass\_Kg\_**.

# 2015 Launch Records

---

List of failed landing\_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
%sql SELECT BOOSTER_VERSION, substr(Date, 4, 2), LAUNCH_SITE FROM SPACEXTBL WHERE  
"LANDING_OUTCOME"='Failure (drone ship)' AND substr(Date,7,4) LIKE '2015';
```



Booster_Version	substr(Date, 4, 2)	Launch_Site
F9 v1.1 B1012	01	CCAFS LC-40
F9 v1.1 B1015	04	CCAFS LC-40

In this SQL query we need to select columns Booster\_Version, month as (Date, 4, 2) and Launch\_Site from SpaceX table.

The **WHERE** clause is used to limit the result to the Landing\_Outcome *'Failure(drope ship)'*

**LIKE** word is used to set the year to **2015**.

**Note: SQLite does not support month names. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.**

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

---

Ranking 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 qty FROM SPACEXTBL WHERE substr(Date,7,4) BETWEEN '2010-06-04' AND '2017-03-20' and "LANDING _OUTCOME"
```

Landing _Outcome	qty
Success (drone ship)	12
Success (ground pad)	8

**COUNT (\*)** counts records in a column

**WHERE** filters data

**BETWEEN & AND** setting the range of dates



A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The image is a composite of a dark blue sky with stars and a view of the Earth's surface from space. The Earth's surface is mostly dark blue, with a thin layer of white clouds. A bright, glowing arc of city lights is visible along the horizon, indicating a coastal or urban area. The text "Section 3" is overlaid on the left side of the image.

Section 3

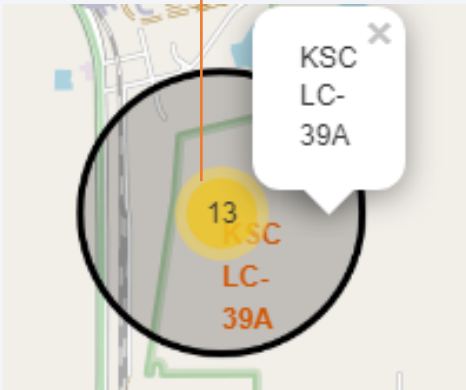
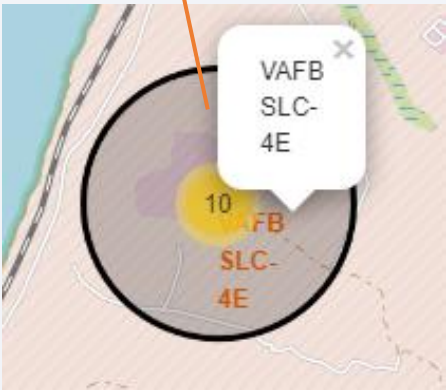
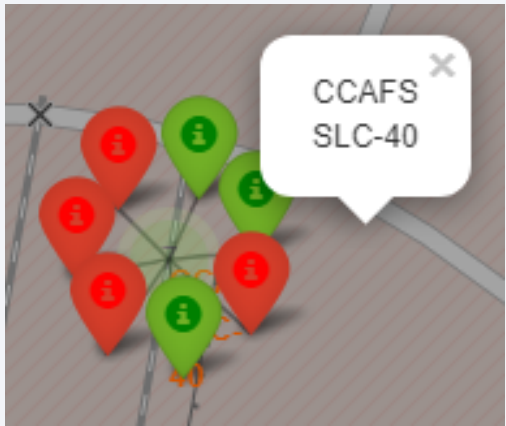
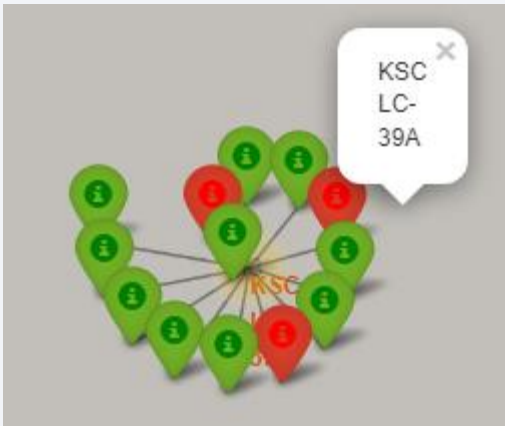
# Launch Sites Proximities Analysis

# All Launch Sites' Location Markers on a Global Map



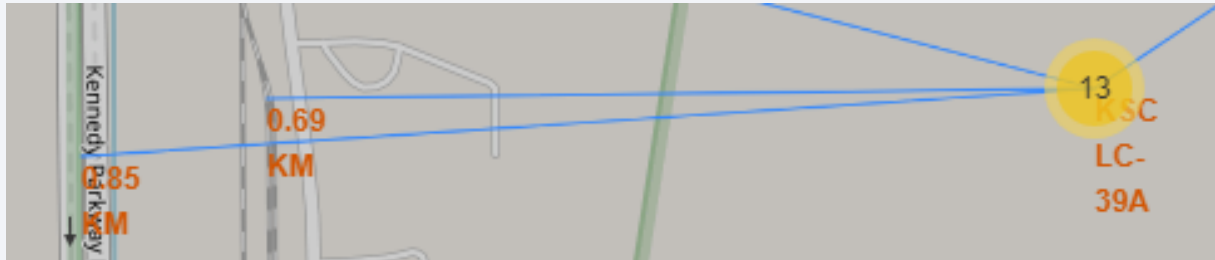
**All SpaceX launch sites are on the California and Florida coasts of the USA**

# Success/failed launches for each site on the Folium Map



**Green** Marker shows successful Launches and **Red** Marker shows Failures

## KSC LC-39A proximity to railway, highway, coastline and city on Folium Map



KSC LC-39A launch site proximity to railway (0.69 km) and highway (0.85 km)



KSC LC-39A launch site proximity to the closest city (16.37 km)



KSC LC-39A launch site proximity to coastline (6.45 km)

*We can see that the launch site is located close to railway and highway, not so far from the coastline and relatively far from the closest city.*



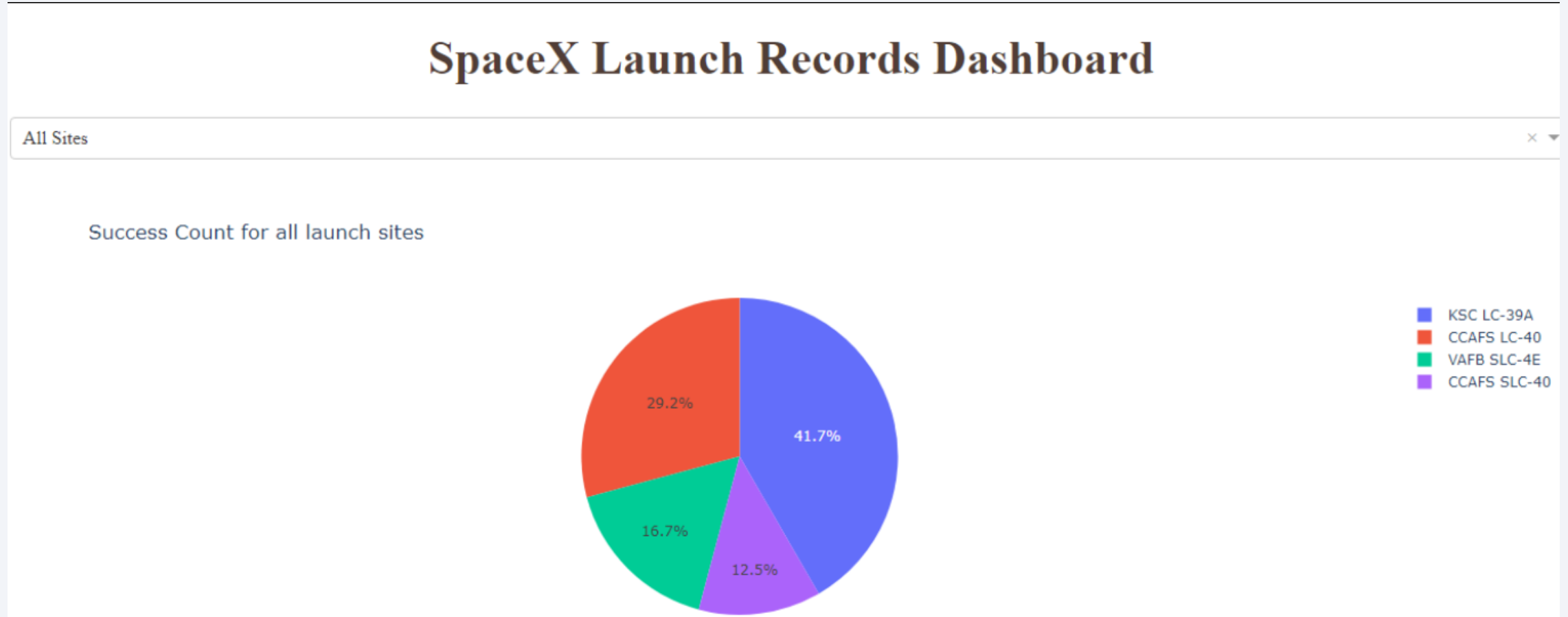


Section 4

# Build a Dashboard with Plotly Dash



# Success Count for All Launch Sites Dashboard



We can see from the dashboard that the highest success launch count is at KSC LC-39A launch site

## DASHBOARD – Pie chart for the launch site with highest launch success ratio

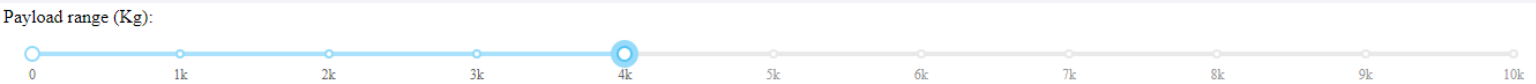
---

Total Success Launches for site KSC LC-39A

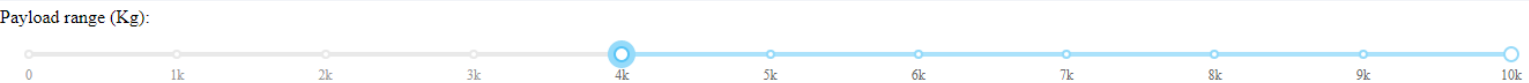
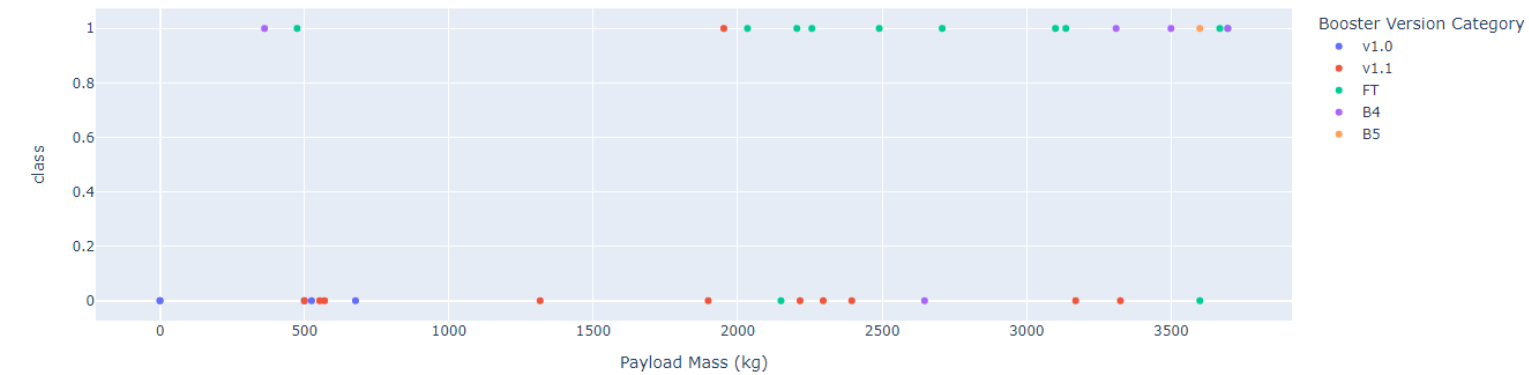


KSC LC-39A achieved a 76.9% success rate and 23.1% failure rate

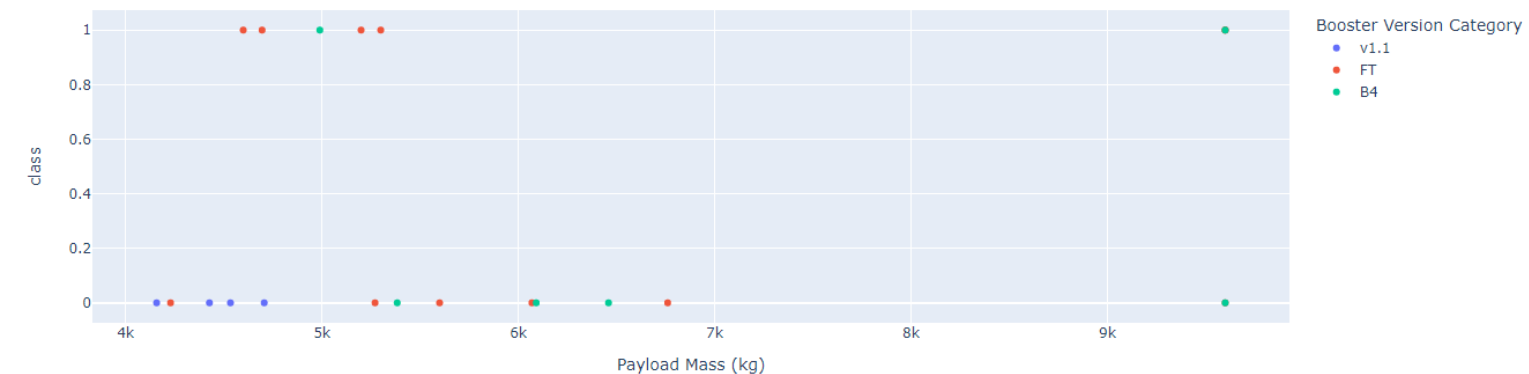
# Dashboard - Payload vs. Launch Outcome scatter plot for all sites



Success count on Payload mass for all sites



Success count on Payload mass for all sites



From these dashboards we can see that the success rate for the low weighted payloads is higher than the heavy weighted payloads



Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

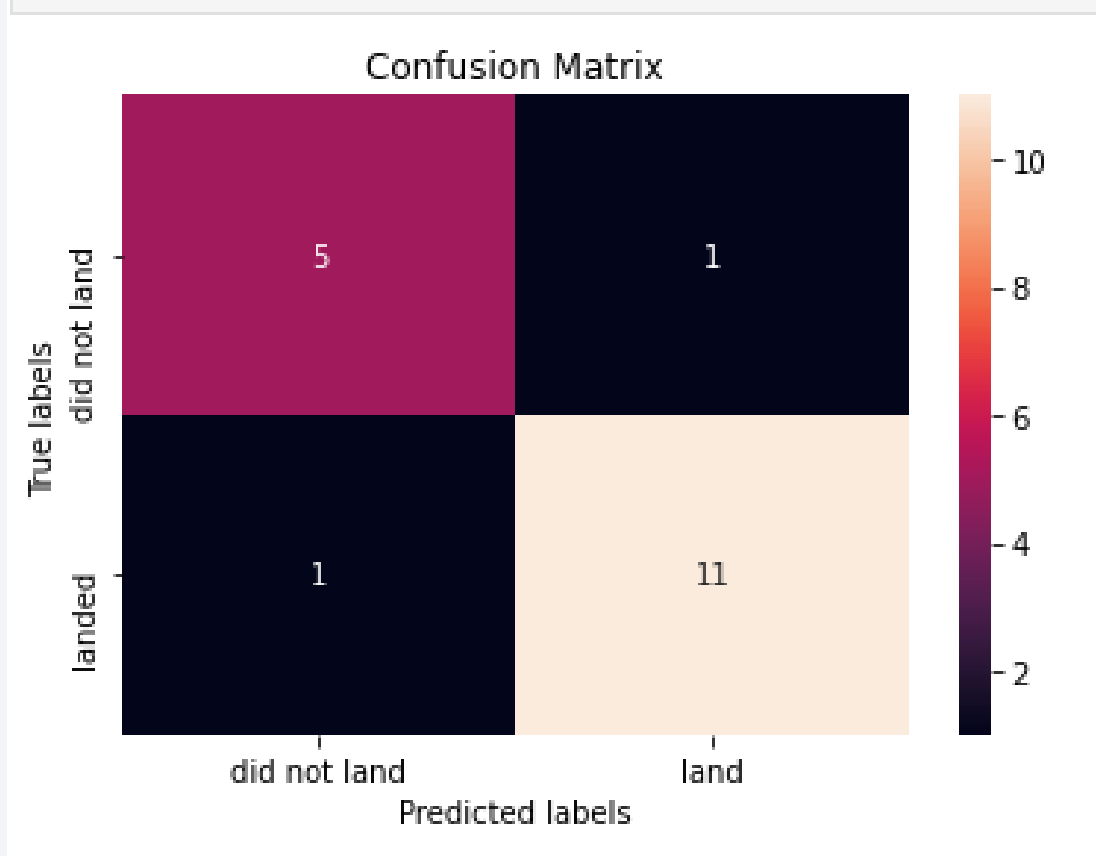
As we can see for below calculations **DECISION TREE** method has the highest classification accuracy

```
print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))  
print('Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))  
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))  
print('Accuracy for K nearest neighbors method:', knn_cv.score(X_test, Y_test))
```

```
Accuracy for Logistics Regression method: 0.8333333333333334  
Accuracy for Support Vector Machine method: 0.8333333333333334  
Accuracy for Decision tree method: 0.8888888888888888 ←  
Accuracy for K nearest neighbors method: 0.8333333333333334
```



# Confusion Matrix



Confusion matrix visualizes and summarizes the performance of a classification algorithm. Decision Tree confusion matrix on the left showed the best accuracy score. The false negative rate is low.



# Conclusions

---



- With the increasing number of flights at a launch site the success rate is getting higher .
- The highest launch success rate is at ES-L1, GEO, HEO and SSO Orbits.
- In the LEO orbit the success appears to be related to the number of flights;  
on the other hand, there seems to be no relationship between flight number when in GTO orbit.
- The highest success launch count is at KSC LC-39A launch site.
- 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 rate and negative landing (unsuccessful mission) are both here.
- The launch site KSC LC-39A is located close to railway and highway, not so far from the coastline and relatively far from the closest city.
- Decision Tree algorithm method showed the highest classification accuracy rate for this dataset.

Thank you!

