
# Winning Space Race with Data Science

By - Naman Vats

11th September 2022

# Outline

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

# Executive Summary

- **Summary of methodologies**
  - Data collection
  - 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**
  - The era of commercial space has arrived, and there are several companies that are making space travel affordable for everyone. Perhaps the most successful of them is *SpaceX*, and one of the reasons is that their rocket launch is relatively inexpensive.
  - *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, we will predict if the *Falcon 9* first stage will land successfully. 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**
  - Correlations between each rocket variables and successful landing rate
  - Conditions to get the best results and ensure the best successful landing rate

Section 1

# Methodology

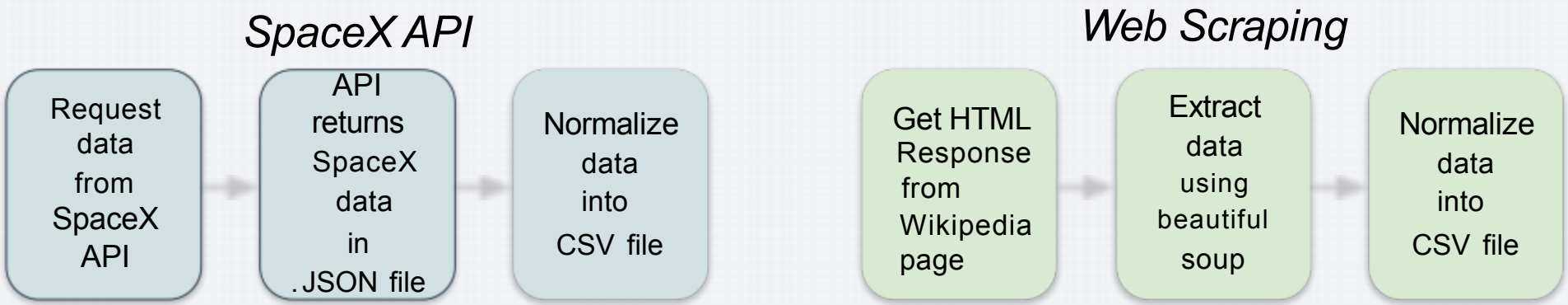
# Methodology

## Executive Summary

- Data collection methodology:
  - SpaceX API
  - WebScrapping from a Wikipedia page
- Perform data wrangling
  - Convert outcomes into Training Labels with the booster successfully/unsuccessful landed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Find best Hyperparameter for SVM, Classification Trees and Logistic Regression

# Data Collection

- The data collection process includes a combination of API requests from the SpaceX API and web scraping data from a table in the Wikipedia page of SpaceX, *Falcon 9 and Falcon Heavy Launches Records*.
  - SpaceX API Data Columns: **FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude, Latitude**
  - Wikipedia Web Scrape Data Columns: **Flight No., Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Version Booster, Booster landing, Date, Time**



# Data Collection – SpaceX API

1.Requesting rocket launch data from SpaceX API

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

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

2.Converting Response to a JSON file

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

3.Using custom functions to clean data

```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getCoreData
getCoreData(data)
```

Github Url

4.Combining the columns into a dictionary to create data frame

```
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}
```

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

5.Filtering dataframe and exporting to a CSV

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_df[launch_df['BoosterVersion'] == 'Falcon 9']
```

```
data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

# Data Collection – Scraping

1.Getting response from HTML

keys

```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url).text
```

2.Creating a BeautifulSoup object

```
# Use BeautifulSoup() to create a BeautifulSoup object
soup = BeautifulSoup(html_data, 'html5lib')
```

3.Finding all tables and assigning the result to a list

```
# Use the find_all function in the BeautifulSoup obje
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

4.Extracting column name one by one

```
column_names = []

# Apply find_all() function with `th` element on fir
# Iterate each th element and apply the provided ext
# Append the Non-empty column name (`if name is not
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.Creating an empty dictionary with

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each valu
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. Filling up the `launch_dict` with launch records

*(Too long to put in here, so please refer to the notebook)*

7.Creating a Dataframe and exporting it to a CSV

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

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

Github Url

# Data Wrangling

- There are several cases in which the booster failed to successfully land on the dataset, and sometimes it attempted to land but failed because of accident.
  - `True Ocean`: the mission result has successfully landed in a specific area of the ocean
  - `False Ocean`: the mission result has not successfully landed in a specific area of the ocean
  - `True RTLS`: the mission result successfully landed on the ground pad
  - `False RTLS`: the mission result has not successfully landed on the ground pad
  - `True ASDS`: the mission result has successfully landed on the drone ship
  - `False ASDS`: the mission result has not landed on the drone ship
- Converting these results into training labels:
  - `1` = successful / `0` = failure

Github Url

# Data Wrangling

1.Calculating the number of launches at each site

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

2.Calculating the number and occurrence of each orbit

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

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

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

4.Creating a landing outcome label from Outcome column

```
landing_class = []
for outcome in df.Outcome:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

```
df['Class']=landing_class
```

5.Calculating the success rate for every landing in dataset

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

```
0.6666666666666666
```

6.Exporting dataset to a CSV

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

# EDA with Data Visualization

- Scatter chart:
  - *Flight Number vs. Launch Site*
  - *Payload vs. Launch Site*
  - *Flight Number vs. Orbit Type*
  - *Payload vs. Orbit Type*
  - A scatter plot shows how much one variable is affected by another. The relationship between two variables is called a correlation. This plot is generally composed of large data bodies.
- Bar chart:
  - *Orbit Type vs. Success Rate*
  - A Bar chart makes it easy to compare datasets between multiple groups at a glance. One axis represents a category and the other axis represents a discrete value. The purpose of this chart is to indicate the relationship between the two axes.
- Line chart:
  - *Year vs. Success Rate*
  - A Line chart shows data variables and trends very clearly and helps predict the results of data that has not yet been recorded.

Github Url

# EDA with SQL

- Loading the dataset into the corresponding table in a Db2 database, and executing SQL queries to answer following questions:
  - 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 when the first successful landing outcome in ground pad was achieved
  - Listing the names of the boosters which have success in drone ship 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 failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  - 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

Github Url

# Build an Interactive Map with Folium

- Objects created and added to a folium map:
  - Markers that show all launch sites on a map
  - Markers that show the success/failed launches for each site on the map
  - Lines that show the distances between a launch site to its proximities
- By adding these objects, following geographical patterns about launch sites are found:
  - Are launch sites in close proximity to railways? *Yes*
  - Are launch sites in close proximity to highways? *Yes*
  - Are launch sites in close proximity to coastline? *Yes*
  - Do launch sites keep certain distance away from cities? *Yes*

Github Url

# Build a Dashboard with Plotly Dash

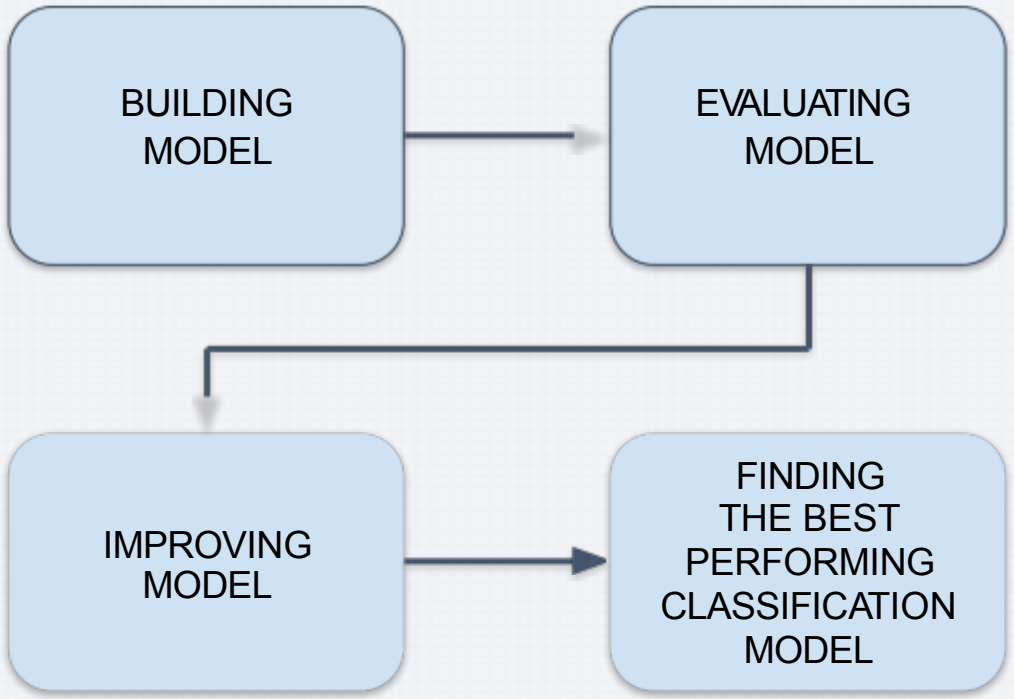
- The dashboard application contains a pie chart and a scatter point chart.
  - *Pie chart*
    - For showing total success launches by sites
    - This chart can be selected to indicate a successful landing distribution across all launch sites or to indicate the success rate of individual launch sites.
  - *Scatter chart*
    - For showing the relationship between `Outcomes` and `Payload mass (Kg)` by different boosters
    - Has 2 inputs: All sites/individual site & Payload mass on a slider between 0 and 10000 kg
    - This chart helps determine how success depends on the launch point, payload mass, and booster version categories.



Github Url

# Predictive Analysis (Classification)

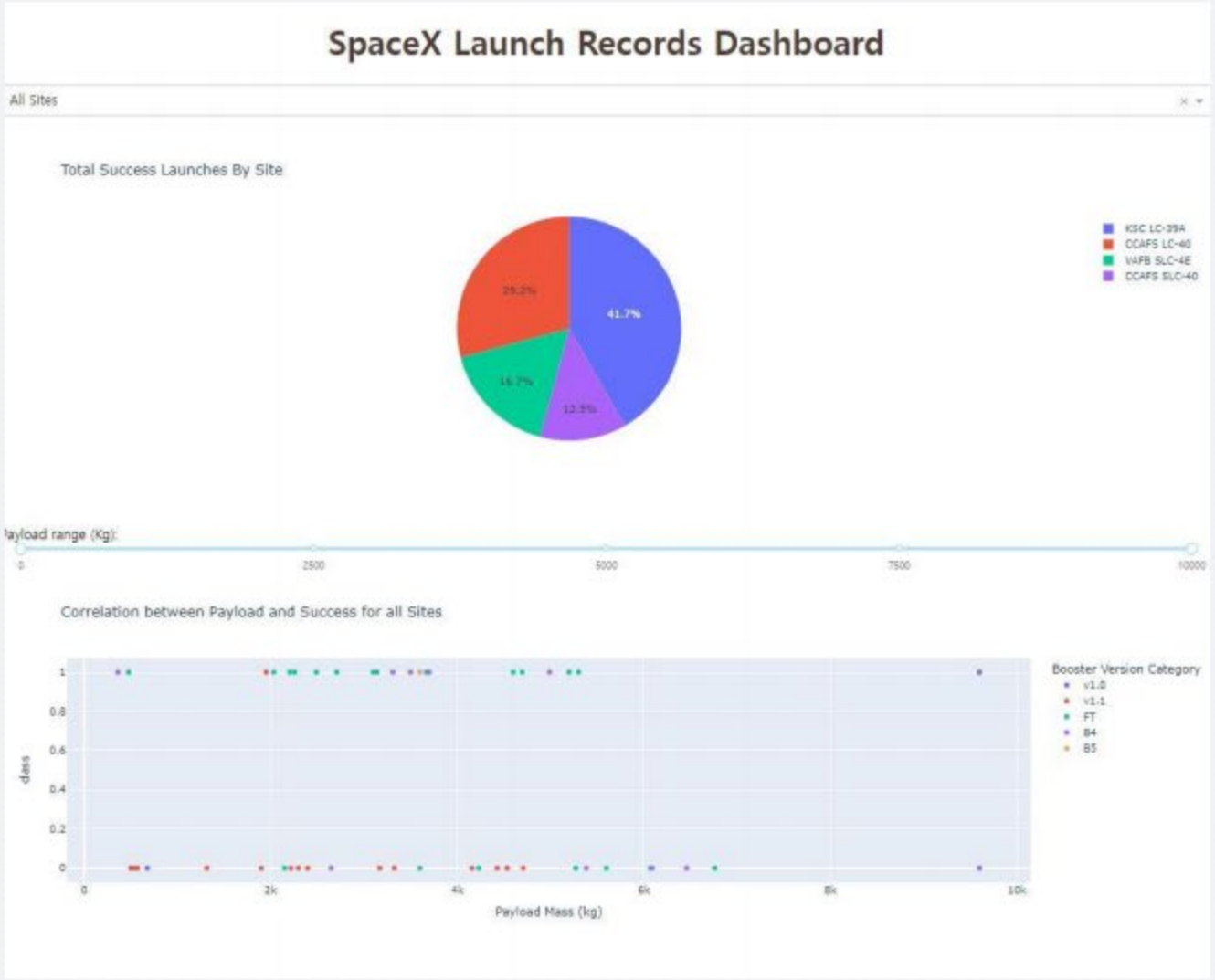
- Perform exploratory Data Analysis and determine Training Labels
  - Create a column for the class
  - Standardize the data
  - Split into training data and test data
- Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
- Find the method performs best using test data






Github Url

# Results




- The results of EDA with visualization, EDA with SQL, Interactive Map with Folium, and Interactive Dashboard will be shown in the next slides.
- The left screenshot is a preview of the Dashboard with Plotly Dash.
- Comparing the accuracy of the four methods, all return the same accuracy of about *83%* for test data.

Section 2

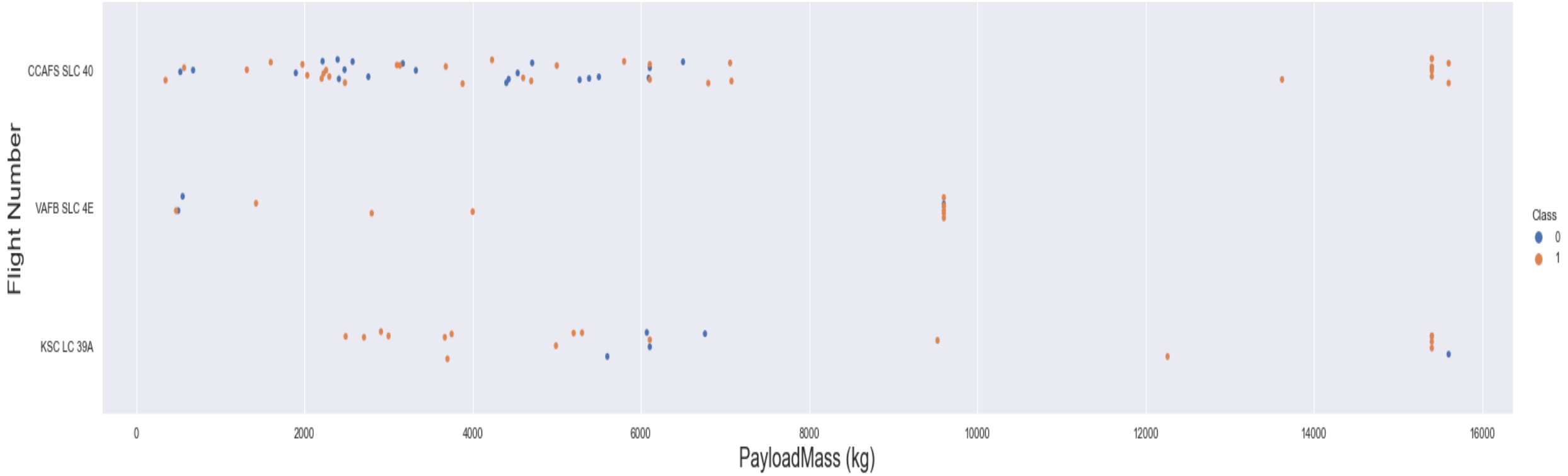
# Insights drawn from EDA

# Flight Number vs. Launch Site

- Class 0 (blue) represents  unsuccessful launch, and Class 1 (orange) represents successful launch.
- This figure shows that **the success rate increased as the number of  flights increased**.
- As the success rate has  increased considerably since the *20th* flight, this point seems to be a big breakthrough.

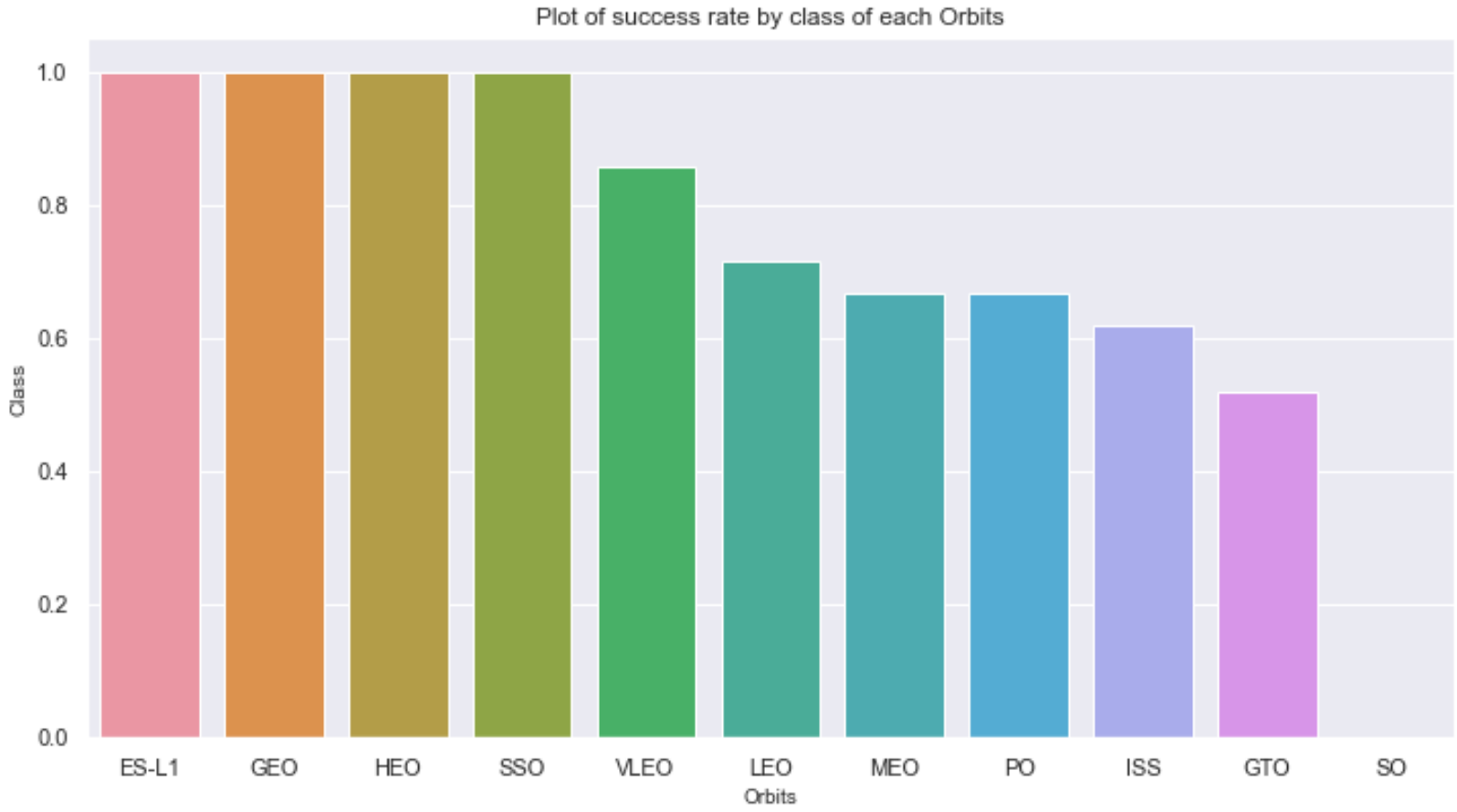
# Payload vs. Launch Site

- Class 0 (blue) represents unsuccessful launch, and Class 1 (orange) represents successful launch.
- At first glance, the larger pay load mass, the higher the rocket's success rate, but it seems difficult to make decisions based on this figure because **no clear pattern can be found between successful launch and Pay Load Mass**.



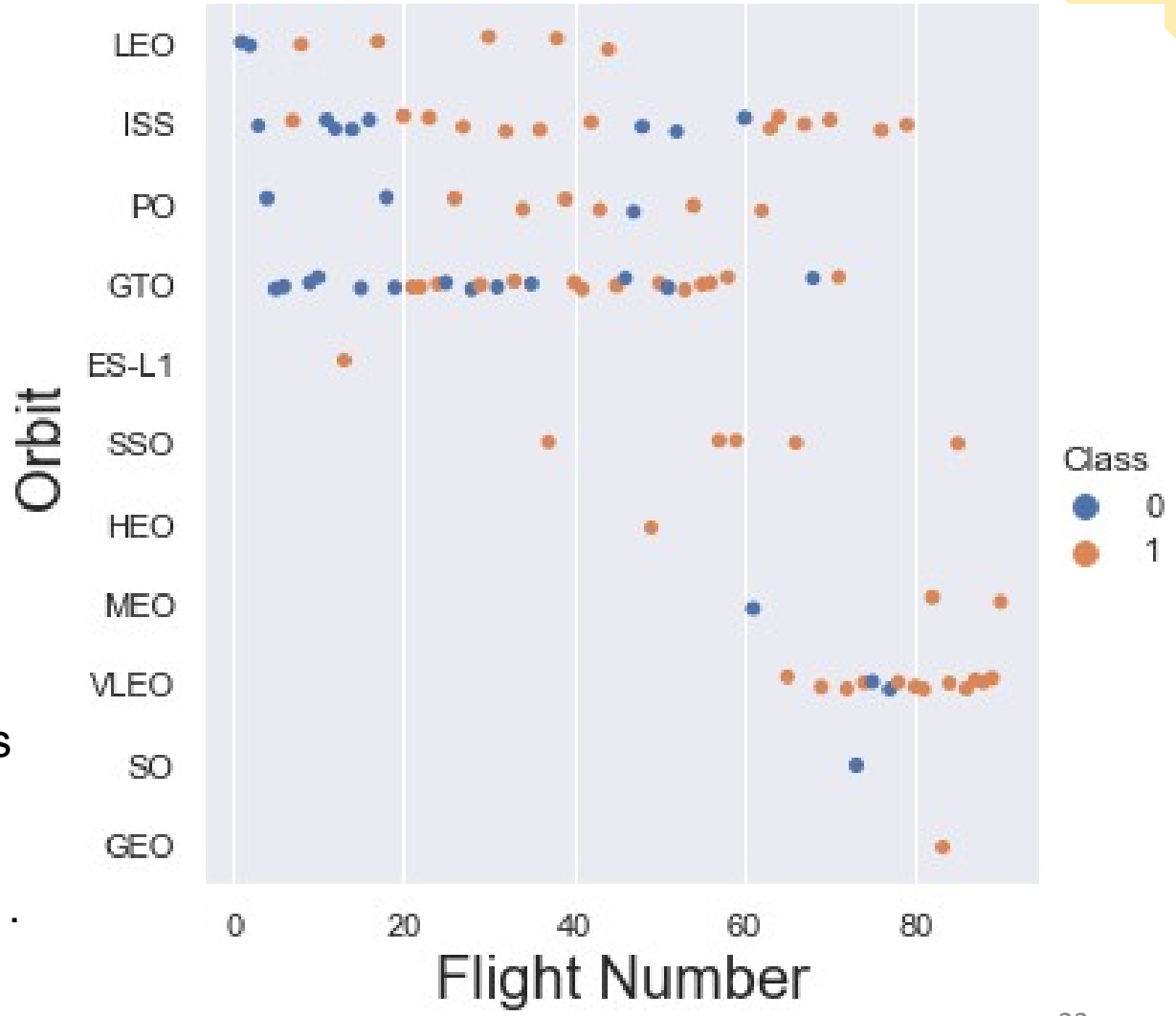
# Success Rate vs. Orbit Type

- Orbit types **SSO**, **HEO**, **GEO**, and **ES-L1** have **the highest success rates** (*100%*).
- On the other hand, the success rate of orbit type **GTO** is only *50%*, and it is the **lowest** except for type SO, which recorded failure in a single attempt.



# Flight Number vs. Orbit Type

- Class 0 (blue) represents unsuccessful launch, and Class 1(orange) represents successful launch.
- In most cases, the launch outcome seems to be correlated with the flight number.
- On the other hand, in **GTO** orbit, there seems to be **no** relationship between flight numbers and success rate.
- SpaceX starts with LEO with a moderate success rate, and it seems that VLEO, which has a high success rate, is used the most in recent launches .

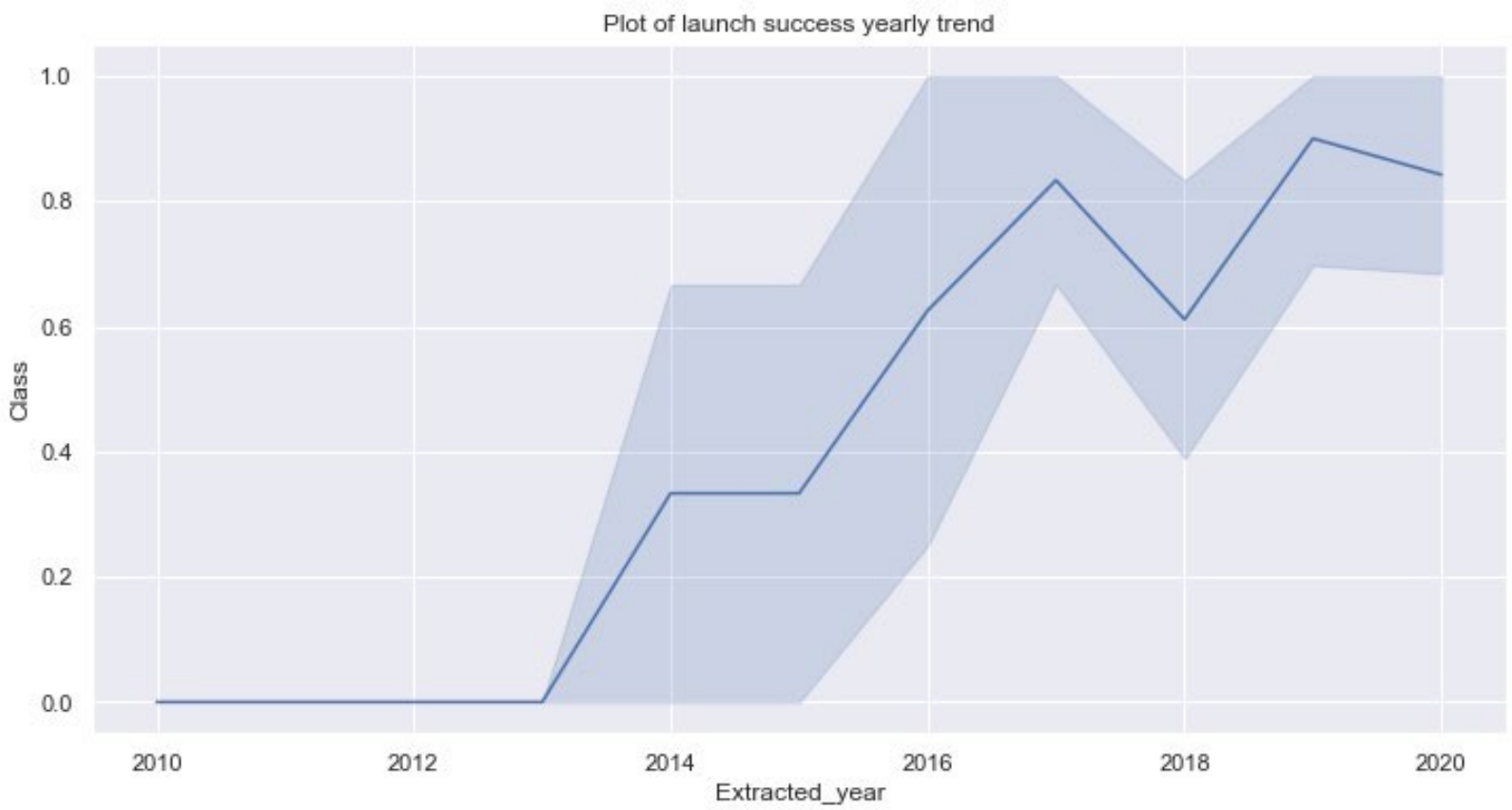


# Payload vs. Orbit Type

- Class 0 (blue) represents unsuccessful launch, and Class 1(orange) represents successful launch.
- With heavy payloads the successful landing or positive landing rate are more for LEO and ISS.
- However, in the case of GTO, it is hard to distinguish between the positive landing rate and the negative landing because they are all gathered

# Launch Success Yearly Trend

- Since 2013, the success rate has continued to **increase** until 2017.
- The rate decreased slightly in 2018.
- Recently, it has shown a success rate of about *80%*.



# Unique launch sites in the space mission

- We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

### Task 1

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

[3]:
```
%%sql
select distinct Launch_Site from SpaceX;
```

```
 * ibm_db_sa://qzz40999:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB;security=SSL
C:\Users\Naman\anaconda3\lib\site-packages\sql\run.py:340: SAWarning: Dialect ibm_db_sa:ibm_db_sa will not make use of SQL compilation caching as it d
oes not set the 'supports_statement_cache' attribute to ``True``.  This can have significant performance implications including some performance degra
dations in comparison to prior SQLAlchemy versions.  Dialect maintainers should seek to set this attribute to True after appropriate development and t
esting for SQLAlchemy 1.4 caching support.   Alternatively, this attribute may be set to False which will disable this warning. (Background on this er
ror at: https://sqlalche.me/e/14/cprf)
  result = conn.session.execute(txt, user_namespace)
Done.
```

t[3]:

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

# Launch Site Names Begining with 'CCA'

We used the query below to display 5 records where launch sites begin with `CCA`

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

In [6]:
```
%%sql
Select * from SpaceX where Launch_Site like 'CCA%'
Limit 5 ;
```

```
 * ibm_db_sa://qzz40999:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB;security=SSL
Done.
```

Out[6]:

| DATE | time_utc_ | booster_version | launch_site | payload | payload_mass_kg_ | orbit | customer | mission_outcome | landing_outcome |
|---|---|---|---|---|---|---|---|---|---|
| 2010-06-04 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 2010-12-08 | 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) |
| 2012-05-22 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

# Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 45596 using the query below

## Task 3

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

In [17]:
```
%%sql
Select Sum(PAYLOAD_MASS__KG_) As Total_Payload from SpaceX where Customer like 'NASA (CRS)';
```

```
 * ibm_db_sa://qzz40999:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB;security=SSL
Done.
```

Out[17]:

| total_payload |
|---|
| 45596 |

# Average Payload Mass Carried By Boosters

We calculated the average payload mass carried by boosters version F9v1.1 as 2928 using the query below

## Task 4

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

In [23]:
```
%%sql
Select Avg(PAYLOAD_MASS__KG_) As Avg_Payload from SpaceX where Booster_Version ='F9 v1.1';
```

```
 * ibm_db_sa://qzz40999:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB;security=SSL
Done.
```

Out[23]:

| avg_payload |
|---|
| 2928 |

# First Successful Ground Landing Date

- We observed that the date of the first successful landing outcome on ground pad was 22nd December 2015

## Task 5

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

[32]:
```
%%sql
Select Min(Date) As firstsuccesful_landing_date from SpaceX where Landing__Outcome like 'Success (ground pad)';
```

```
 * ibm_db_sa://qzz40999:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB;security=SSL
Done.
```

[32]:

| firstsuccesful_landing_date |
|---|
| 2015-12-22 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

## 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

[33]:
```
%%sql
Select Booster_Version from SpaceX where Landing__Outcome like 'Success (drone ship)'
And PAYLOAD_MASS__KG_ between 4000 and 6000;
```

```
 * ibm_db_sa://qzz40999:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB;security=SSL
Done.
```

t[33]:

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

# Total Number of Successful and Failure Mission Outcomes

- We used string pattern ’ like %’ to filter for **WHERE** Mission Outcome was a success or a failure and used the **Count** to get the number.

## Task 7

List the total number of successful and failure mission outcomes

[42]:
```
%%sql
Select Count(Mission_Outcome) as Failed_mission_outcomes from SpaceX where Mission_outcome like 'Failure%'
```

```
 * ibm_db_sa://qzz40999:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB;security=SSL
Done.
```

ut[42]:

| failed_mission_outcomes |
|---|
| 1 |

[44]:
```
%%sql
Select Count(Mission_Outcome) as Successful_mission_outcomes from SpaceX where Mission_outcome like 'Success%';
```

```
 * ibm_db_sa://qzz40999:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB;security=SSL
Done.
```

ut[44]:

| successful_mission_outcomes |
|---|
| 100 |

# Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

### Task 8

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

In [47]:
```
%%sql
SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SpaceX
WHERE PAYLOAD_MASS__KG_ = (Select Max(PAYLOAD_MASS__KG_) from SpaceX )
ORDER BY Booster_Version;
```

```
 * ibm_db_sa://qzz40999:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB;security=SSL
Done.
```

Out[47]:

| 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 |

# 2015 Launch Records

- We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

## Task 9

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

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

49]:
```
%%sql
SELECT Date,Booster_Version, Launch_Site, Landing__Outcome FROM SpaceX
WHERE Landing__Outcome LIKE 'Failure (drone ship)'
AND Date BETWEEN '2015-01-01' AND '2015-12-31'
```

```
 * ibm_db_sa://qzz40999:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB;security=SSL
Done.
```

49]:

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

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

- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

[53]:
```
%%sql
SELECT Landing__Outcome, COUNT(Landing__Outcome) as NumberofTimes FROM SpaceX
WHERE Landing__Outcome like 'Success%' And DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing__Outcome
ORDER BY COUNT(Landing__Outcome) DESC
```

```
 * ibm_db_sa://qzz40999:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB;security=SSL
Done.
```

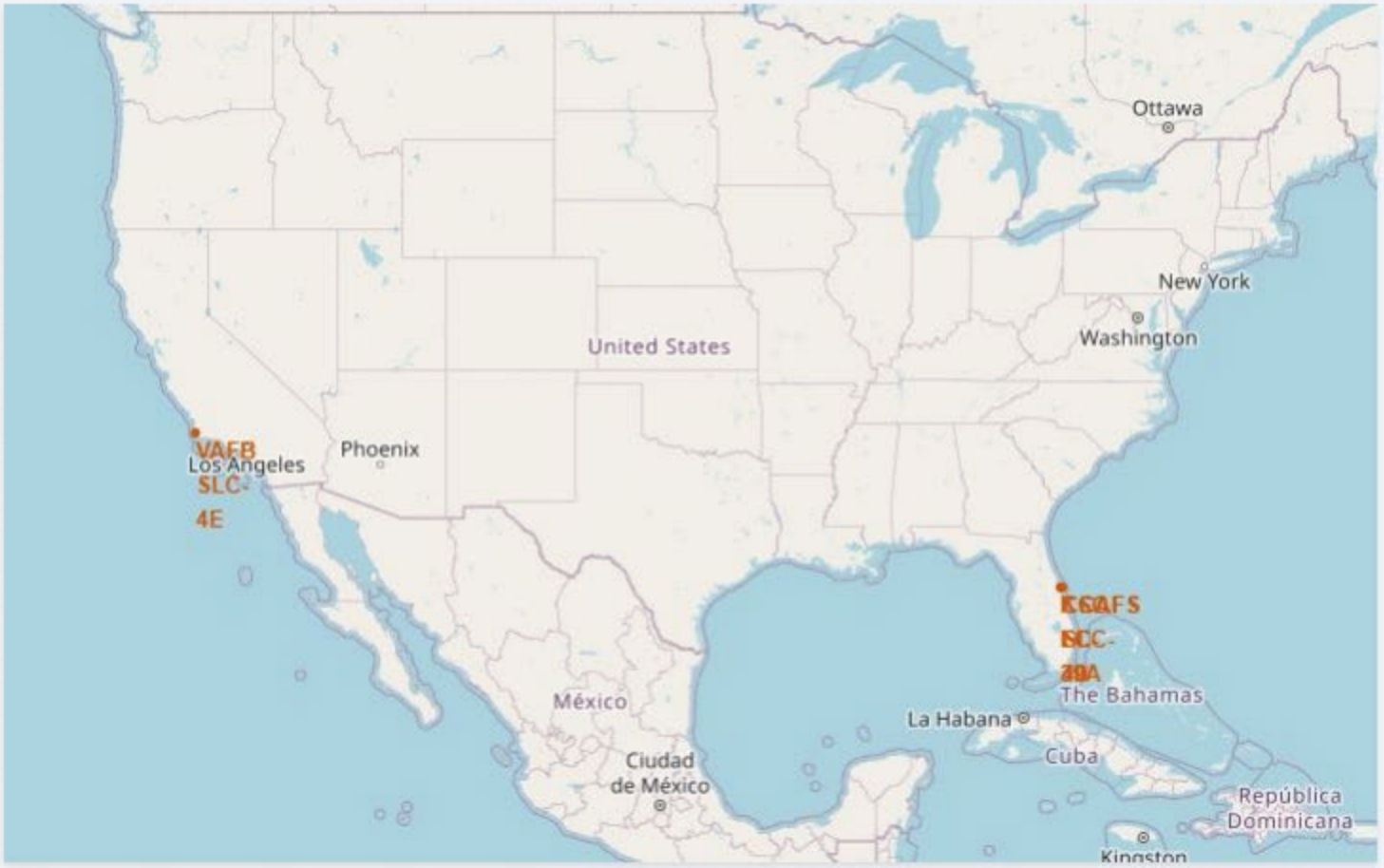
[53]:

| landing__outcome | numberoftimes |
|---|---|
| Success (drone ship) | 5 |
| Success (ground pad) | 3 |

Section 4

# Launch Sites Proximities Analysis

# All Launch Sites' Locations

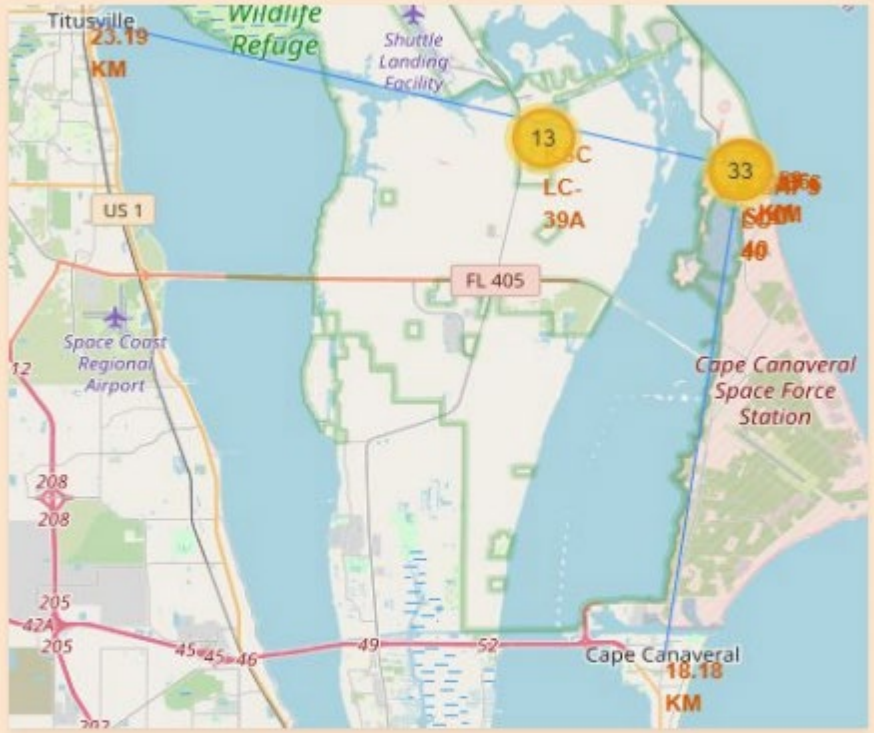



- The left map shows all *SpaceX* launch sites, and the right map also shows that all launch sites are in the United States.
- As can be seen on the map, all launch sites are near the coast.

# Markers showing launch sites with color labels

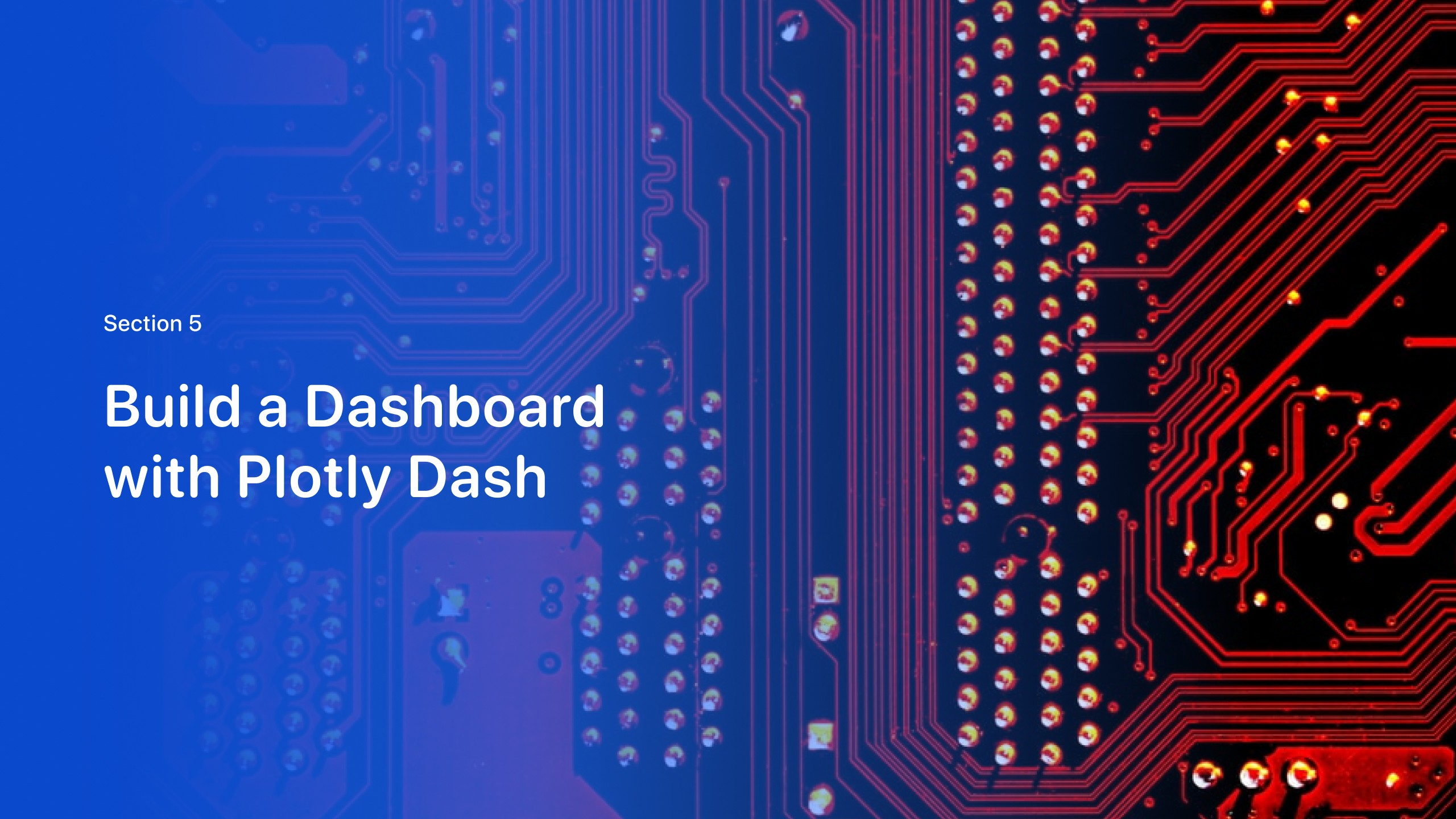
# Proximities of Launch Sites


Are launch sites in close proximity to railways? ***Yes***
Are launch sites in close proximity to highways? ***Yes***
Are launch sites in close proximity to coastline? ***Yes***




Do launch sites keep certain distance away from cities? ***Yes***

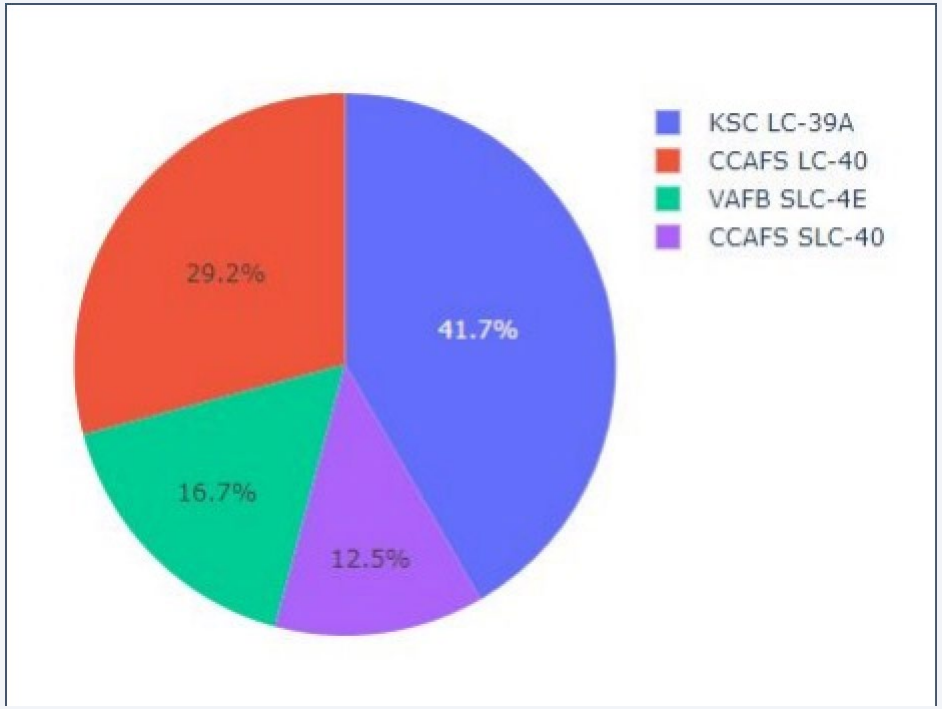
- It can be found that the launch site is **close to railways and highways** for transportation of equipment or personnel, and is also **close to coastline** and relatively **far from the cities** so that launch failure does not pose a threat.



Section 5

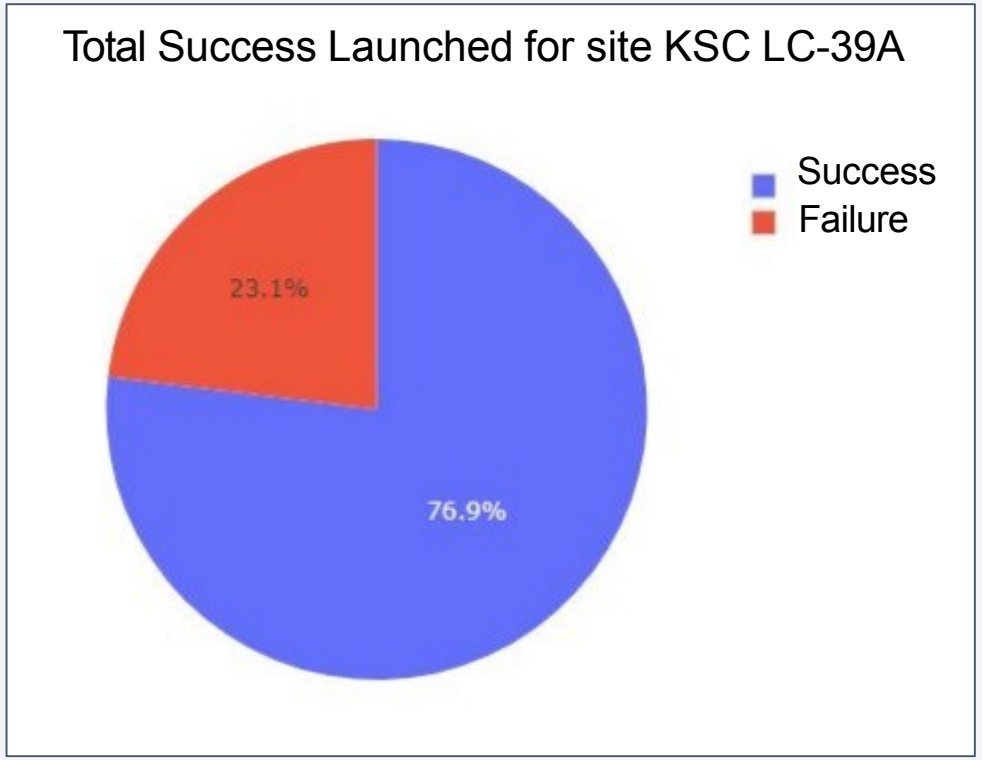
# Build a Dashboard with Plotly Dash

# Total Success Launches By all sites




- KSLC-39A records the most launch success among all sites.
- The VAFB SLC-4E has the fewest launch success, possibly because
  - the data sample is small, or
  - because it is the only site located in California, so the launch difficulty on the west coast may be higher than on the east coast.

# Launch Site with Highest Launch Success Ratio




- KSLC-39A has the highest success rate with 10 landing successes *(76.9%)* and 3 landing failures *(23. 1%)*.

# Payload vs. Launch Outcome Scatter Plot for all sites



- These figures show that **the launch success rate** *(class 1)* **for low weighted payloads*(0-5000 kg)* is higher** than that of heavy weighted payloads*(5000- 10000 kg)* .

Section 6

# Predictive Analysis (Classification)

# Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy

Find the method performs best:

In [30]:
```
algorithms = {'KNN':knn_cv.best_score_,'Decision Tree':tree_cv.best_score_,'Logistic Regression':logreg_cv.best_score_,'SVM':svm_cv.best_score_}
best_algorithm = max(algorithms, key=algorithms.get)

print('The method which performs best is ',best_algorithm, 'with a score of',algorithms[best_algorithm])
```

```
The method which performs best is  Decision Tree with a score of 0.8875000000000002
```

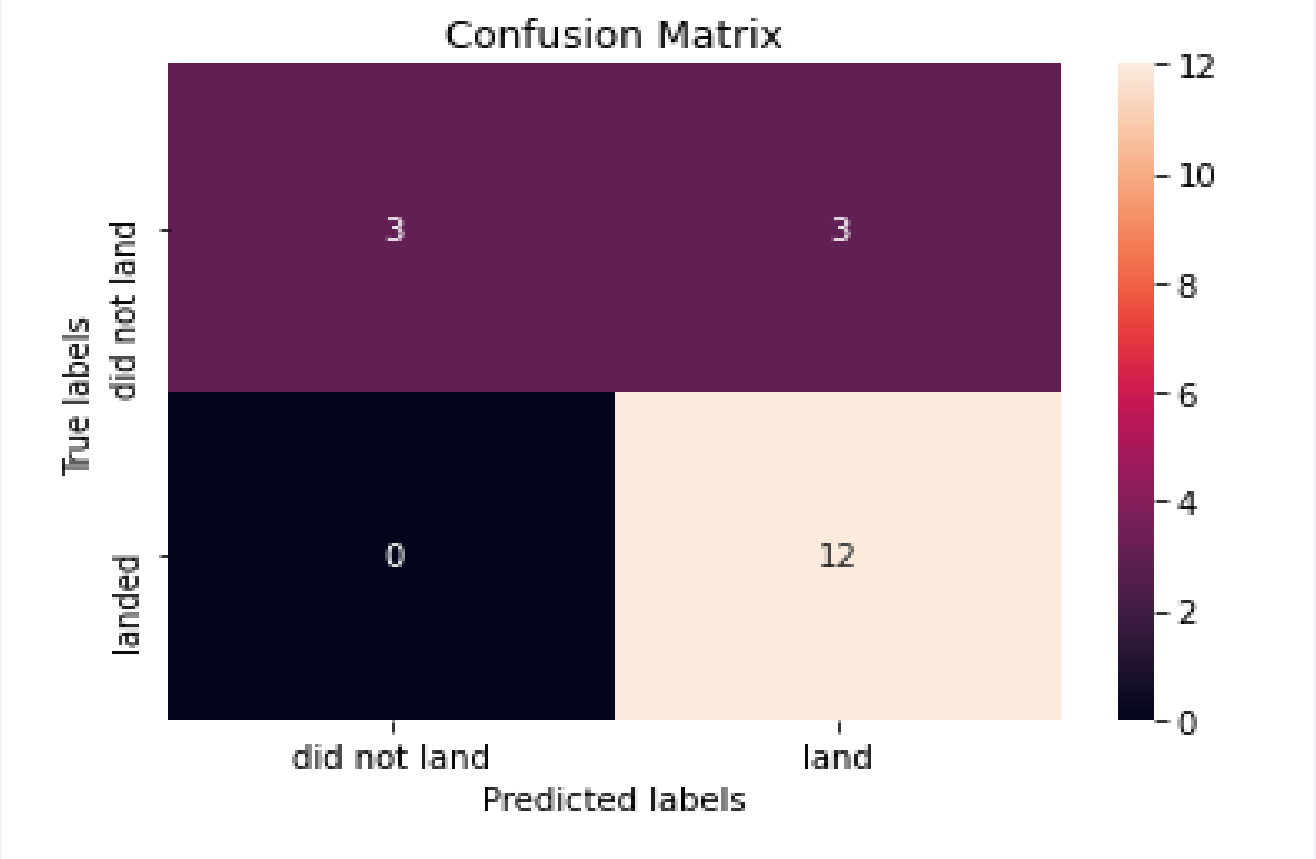
In [31]:
```
algorithms_df = pd.DataFrame.from_dict(algorithms, orient='index', columns=['Accuracy'])
algorithms_df
```

Out[31]:

| | Accuracy |
|---|---|
| **KNN** | 0.848214 |
| **Decision Tree** | 0.887500 |
| **Logistic Regression** | 0.846429 |
| **SVM** | 0.848214 |

# Confusion Matrix

- The confusion matrix is the same for all models because all models performed  the same for the test set.
- The models predicted 12 successful landings when the true label was successful and 3 failed landings when the true label was failure. But there were also 3 predictions that said successful landings when the true label was failure *(false positive)*.
- Overall, **these models predict successful landings**



# Conclusions

- As the number of flights increased, the success rate increased, and recently it has exceeded 80%.
- Orbital types SSO, HEO, GEO, and ES-L1 have the highest success rate (100%).
- The launch site is close to railways, highways, and coastline, but far from cities.
- KSLC-39A has the highest number of launch successes and the highest success rate among all sites.
- The launch success rate of low weighted payloads is higher than that of heavy weighted payloads.
- In this dataset, the accuracy of all models was virtually the same at 83.33%, so it seems that more data is needed to determine the optimal model due to the small data size.

# Thank you!