IBM Developer
SKILLS NETWORK

Winning Space Race With Data Science

Kumud Prajapati
25th-March-2024

Outline

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

Executive Summary

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

Introduction

- Project background and context

  Space X 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 Space X 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 space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

  i. What factors determine if the rocket will land successfully?

  ii. The interaction amongst various features that determine the success rate of a successful landing.

  iii. What operating conditions needs to be in place to ensure a successful landing program

Section 1

Methodology

Methodology

Executive Summary

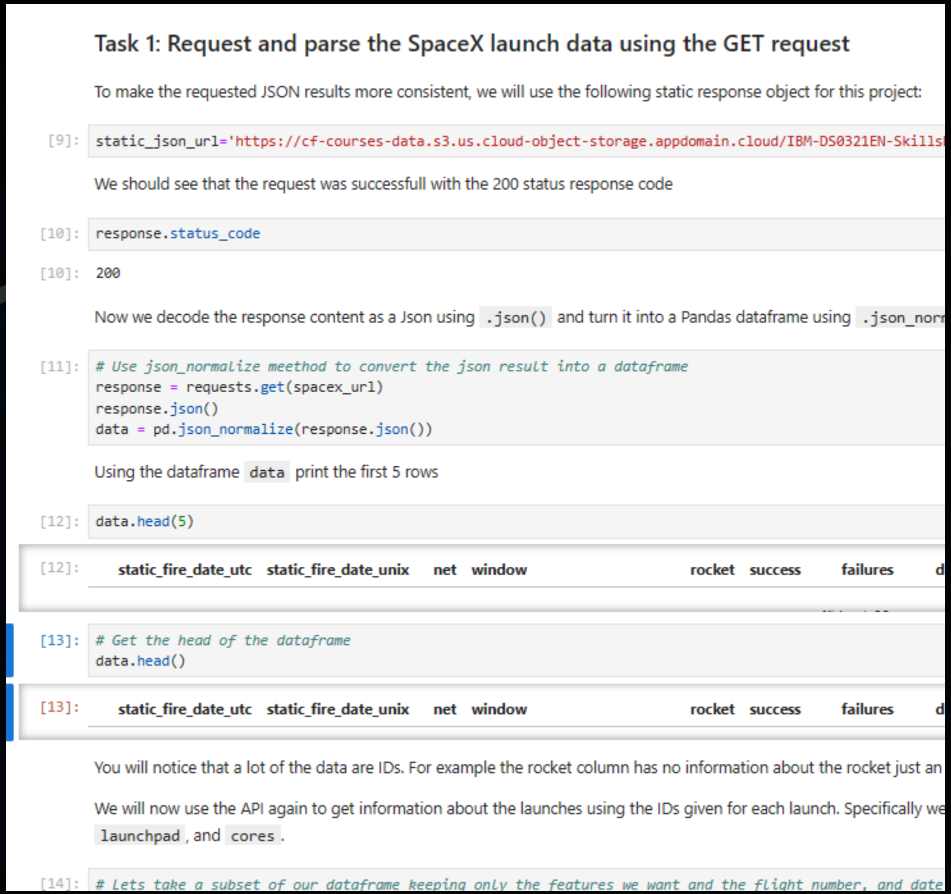
- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is : [Data Collection - SpaceX API]()



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:

```
[9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Skills
```

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

```
[10]: response.status_code
[10]: 200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_norm`

```
[11]: # Use json_normalize meethod to convert the json result into a dataframe
response = requests.get(spacex_url)
response.json()
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

```
[12]: data.head(5)
```

static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures

```
[13]: # Get the head of the dataframe
data.head()
```

static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures

You will notice that a lot of the data are IDs. For example the rocket column has no information about the rocket just an

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we `launchpad`, and `cores`.

```
[14]: # Lets take a subset of our dataframe keeping only the features we want and the flight number, and date
```

Data Collection – Web Scraping

- We applied web scrapping to webscrap Falcon 9 lauch records with BeautifulSoup.
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is : [Data Collection - Web Scraping]()

[4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&o
```

Next, request the HTML page from the above URL and get a `response` object

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.

[5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
response.status_code ==200
```

[5]: True

Create a `BeautifulSoup` object from the HTML `response`

[6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text)
```

Print the page title to verify if the `BeautifulSoup` object was created properly

[7]:
```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

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

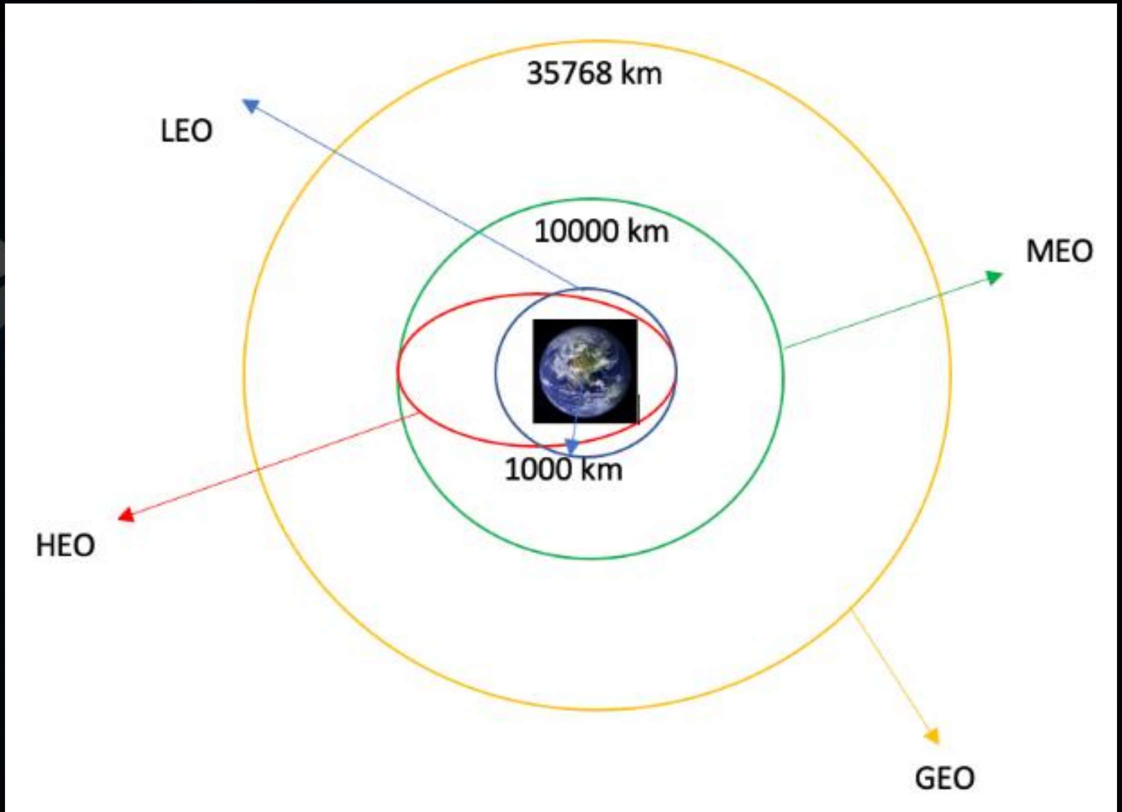
Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please ch
reference link towards the end of this lab

[8]:
```
# 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')
len(html_tables) # just trying to check the length of the table
```

[8]: 25

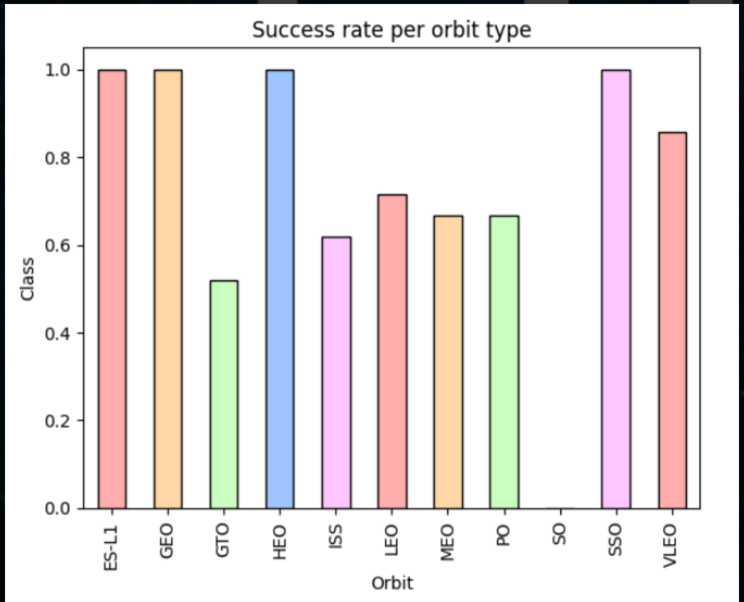
Data Wrangling

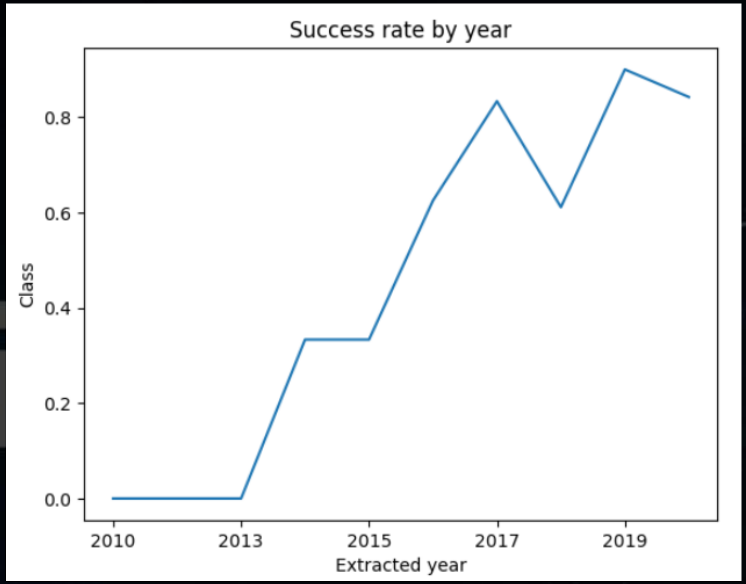
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits.
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is : [Data Wrangling]



EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.





- The link to the notebook is : [EDA with Data Visualization]

Data Collection

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:-
  - The names of unique launch sites in the space mission.-
  - The total payload mass carried by boosters launched by NASA (CRS)-
  - The average payload mass carried by booster version F9 v1.1-
  - The total number of successful and failure mission outcomes-
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is : [Data Collection sql](#)

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
    - Are launch sites near railways, highways and coastlines.
    - Do launch sites keep certain distance away from cities.

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.
- The link to the notebook is : [Plotly Dash app]

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is : [predictive analysis]

Results

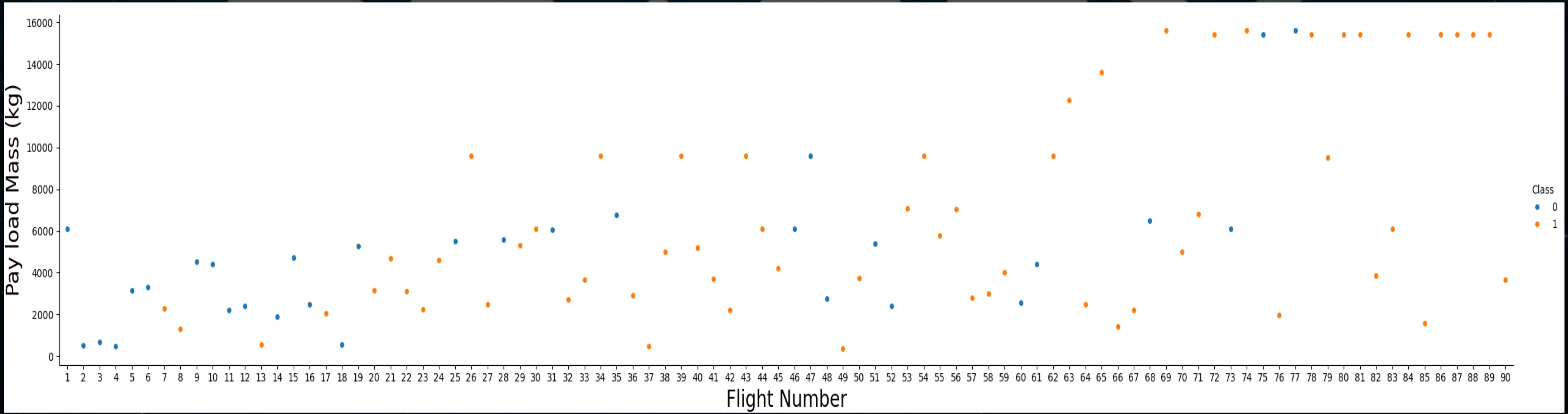
- Exploratory data analysis results.
- Interactive analytics demo in screenshots.
- Predictive analysis results

Section 2

Insight drawn from EDA

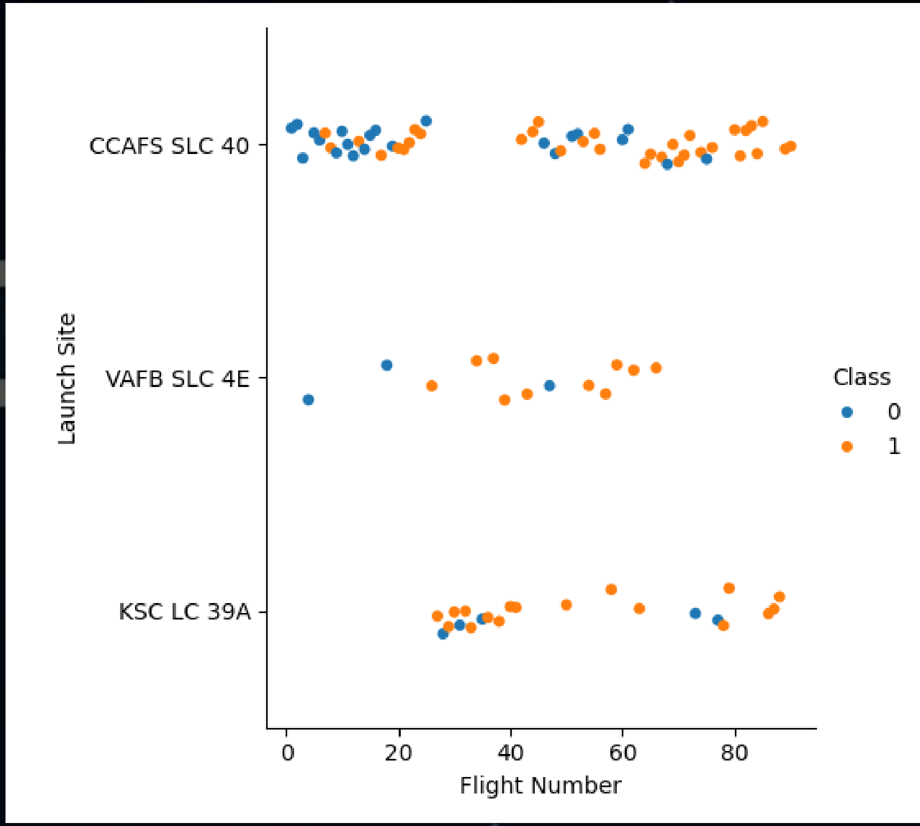
Flight Number Vs. PayLoad Mass

- We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return..



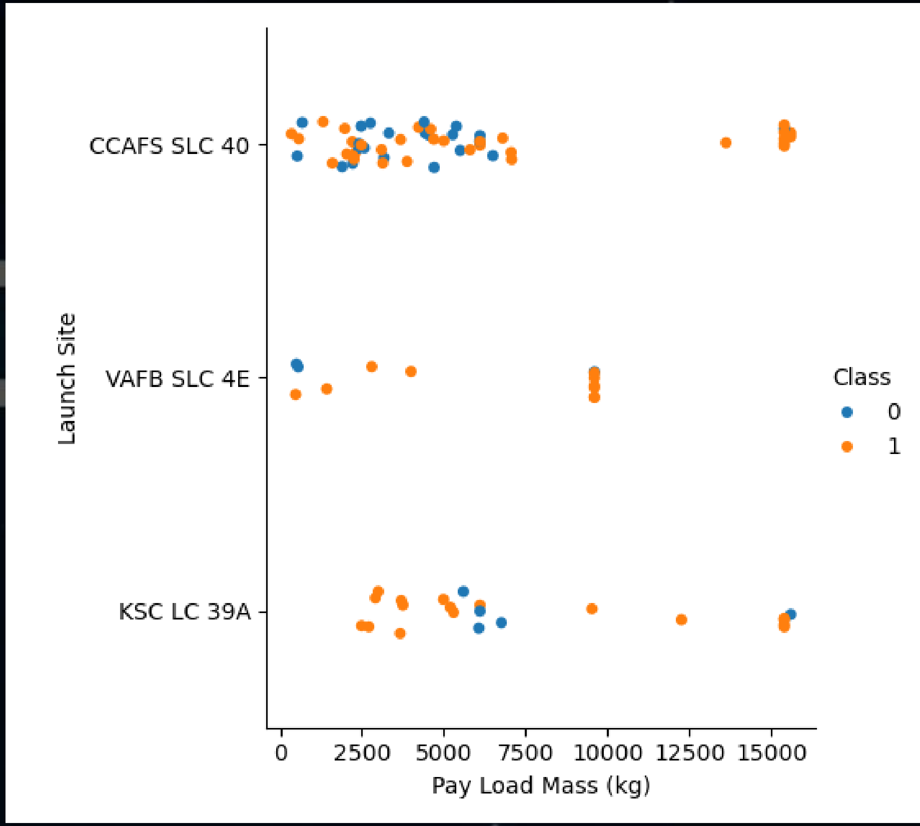
Flight Number Vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



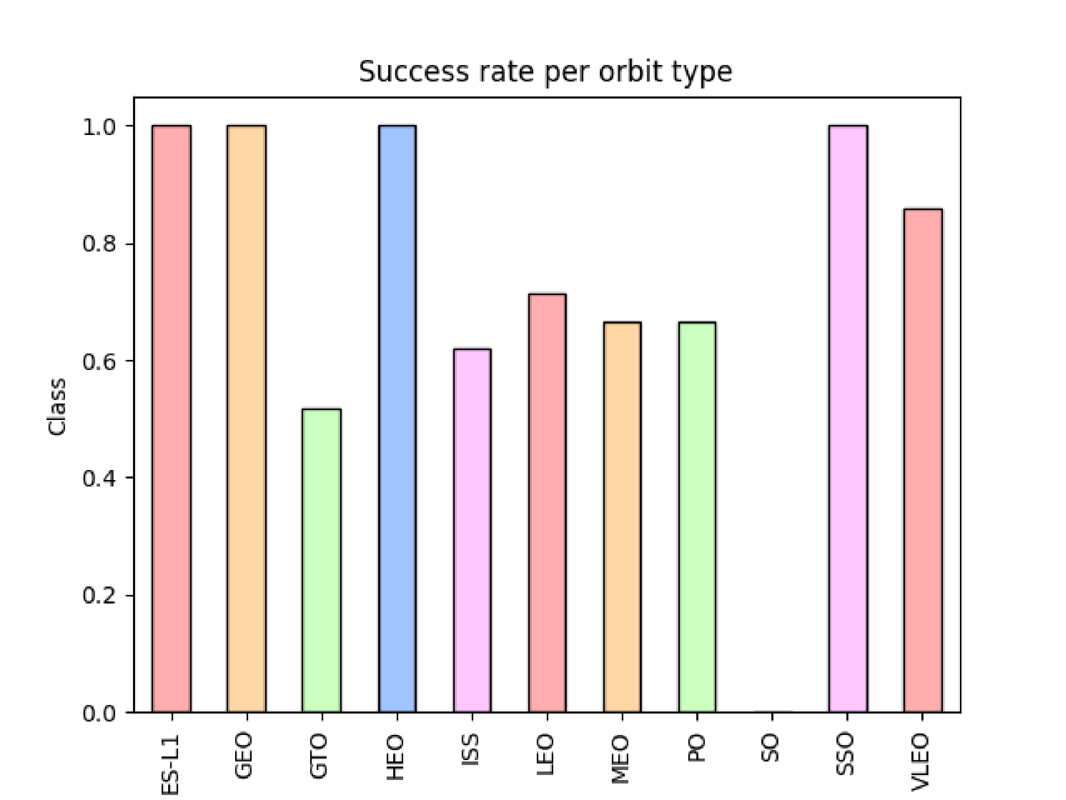
Payload Mass Vs. Launch Site

- The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.



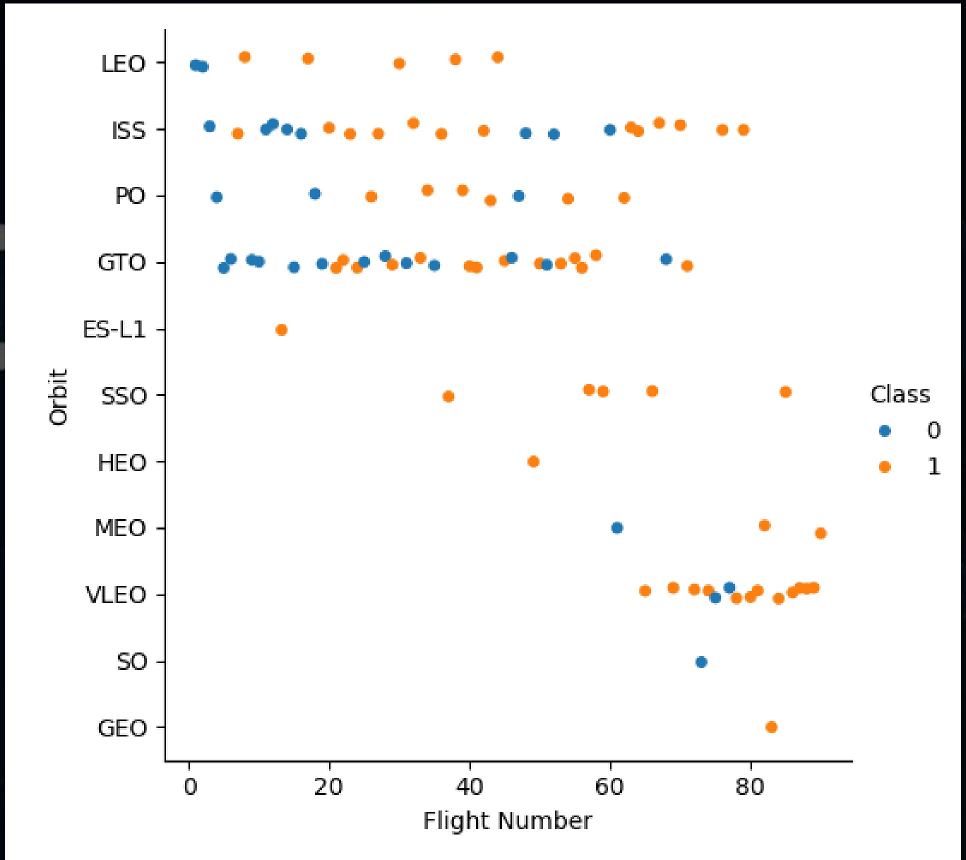
Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



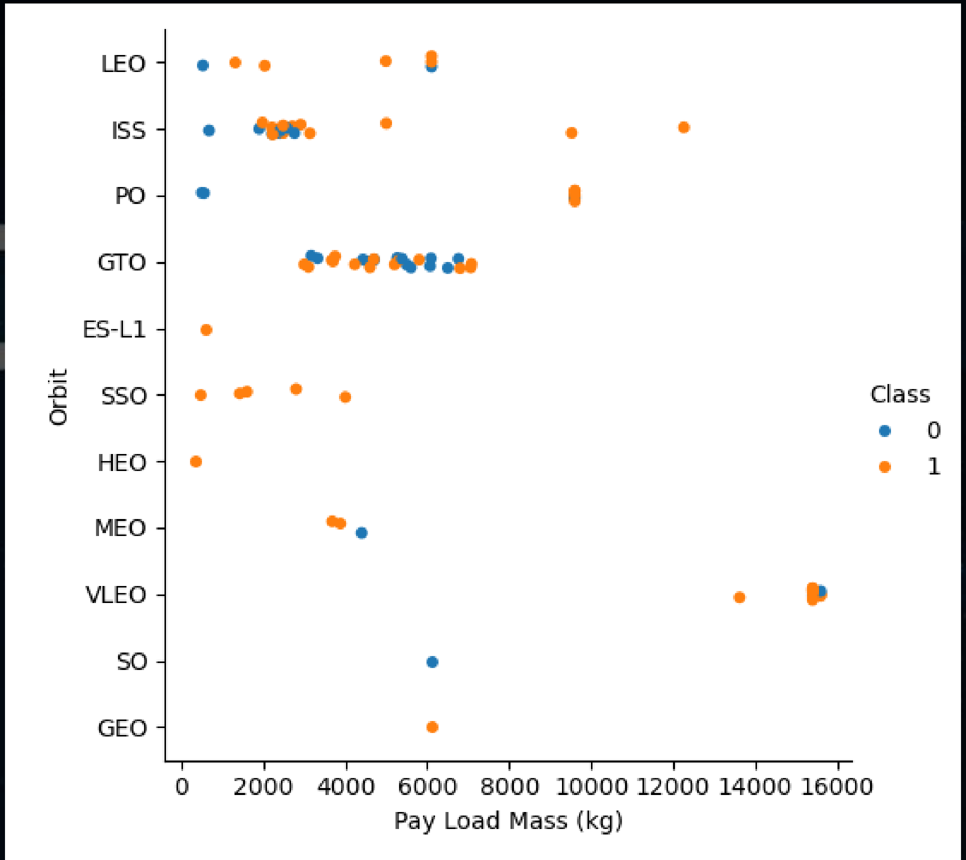
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe that 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.



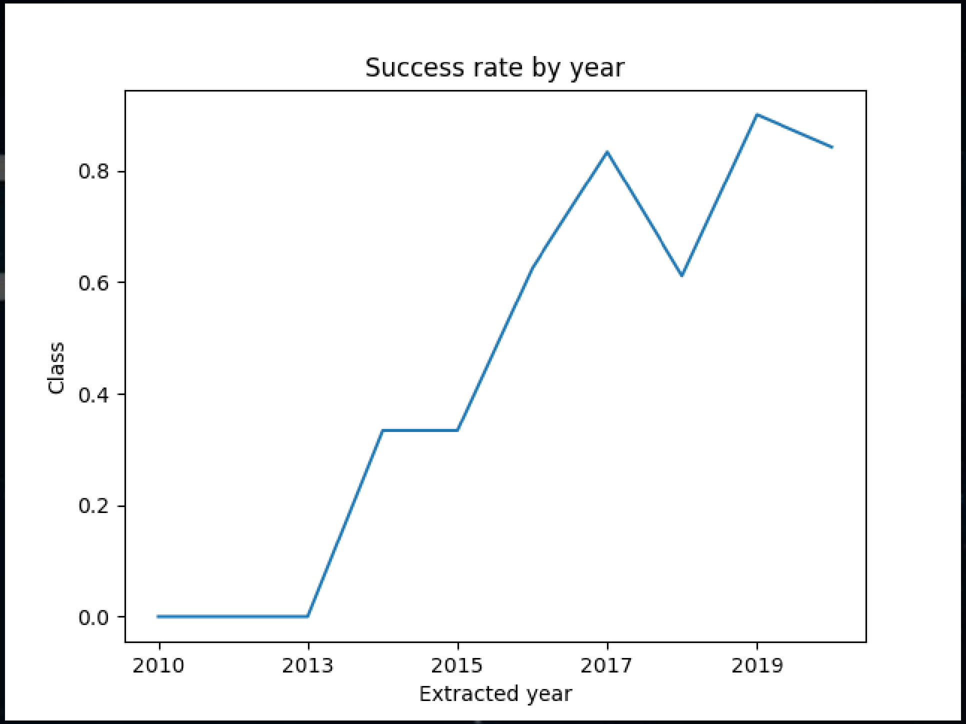
Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- 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

```
[9]: sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1
 * sqlite:///my_data1.db
Done.
```

[9]:

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

Launch Site Names Begin with ‘CCA’

Task 2

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

[10]:
```
sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```

```
 * sqlite:///my_data1.db
Done.
```

[10]:

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

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

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)

```
[11]: sql SELECT SUM (PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER='NASA (CRS)'
```

```
 * sqlite:///my_data1.db
Done.
```

[11]:

SUM (PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4.

Task 4

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

[32]:
```
%%sql
select avg(PAYLOAD_MASS__KG_) as "Average Payload F9V1.1"
from SPACEXTABLE
where Booster_Version like '%F9 v1.1'
```

```
 * sqlite:///my_data1.db
Done.
```

[32]:

Average Payload F9V1.1
2928.4

First Successful Ground landing Date

- We observed that the dates 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.

Hint:Use min function

[58]:
```
%%sql
select MIN(Date) AS First_Sucessfull_Landing
from SPACEXTBL
WHERE Landing_Outcome = "Success (ground pad)";
```

```
 * sqlite:///my_data1.db
Done.
```

[58]:

First_Sucessfull_Landing
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

[31]:
```
%%sql
select Booster_Version
from SPACEXTABLE
where Landing_Outcome = 'Success (drone ship)' and (PAYLOAD_MASS__KG_ between 4000 and 6000)
```

```
 * sqlite:///my_data1.db
Done.
```

[31]:

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 wildcard like '%' to filter for WHERE Mission Outcome was a success or a failure.

Task 7

List the total number of successful and failure mission outcomes

[60]:
```
%%sql
SELECT Mission_Outcome, COUNT(*) as Total_Count
FROM SPACEXTBL
GROUP BY Mission_Outcome;
```

```
 * sqlite:///my_data1.db
Done.
```

[60]:

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

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

```
[52]: %%sql
select Booster_Version, PAYLOAD_MASS__KG_
from SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = ( SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

```
 * sqlite:///my_data1.db
Done.
```

[52]:

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

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

[53]:
```
%%sql
select substr(Date, 6,2) as month , Landing_Outcome , Booster_Version,Launch_Site
from SPACEXTABLE
where substr(Date,0,5)='2015' and Landing_Outcome = 'Failure (drone ship)'
```

```
 * sqlite:///my_data1.db
Done.
```

[53]:

month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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.

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground
03-20, in descending order.

[63]:
```
%%sql
SELECT Landing_Outcome, COUNT(*) AS OutcomeCount, RANK() OVER (ORDER BY COUN
FROM SPACEXTBL
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY OutcomeCount DESC;
```

```
 * sqlite:///my_data1.db
Done.
```

[63]:

Landing_Outcome	OutcomeCount	Rank
No attempt	10	1
Success (drone ship)	5	2
Failure (drone ship)	5	2
Success (ground pad)	3	4
Controlled (ocean)	3	4
Uncontrolled (ocean)	2	6
Failure (parachute)	2	6
Precluded (drone ship)	1	8

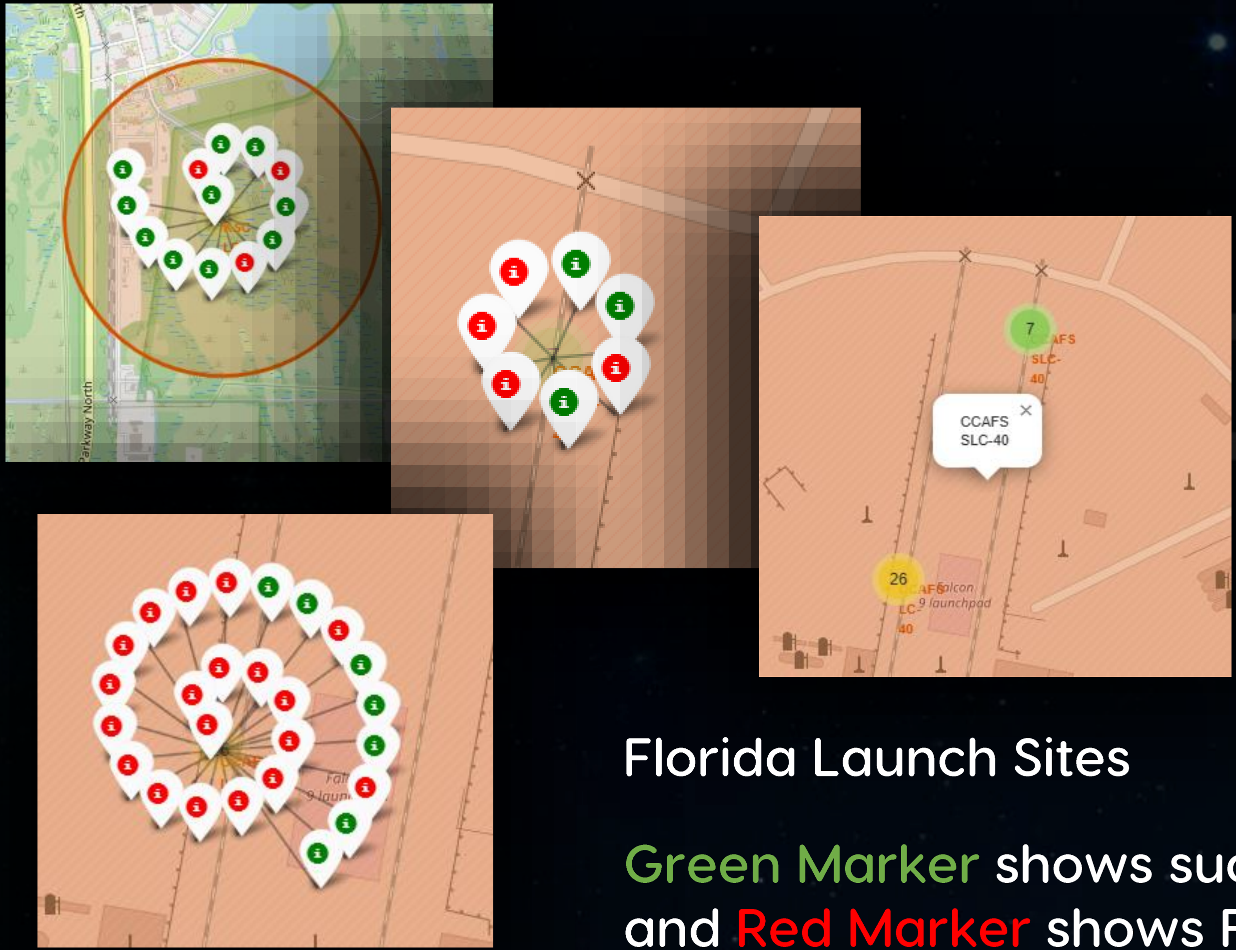
Section 3

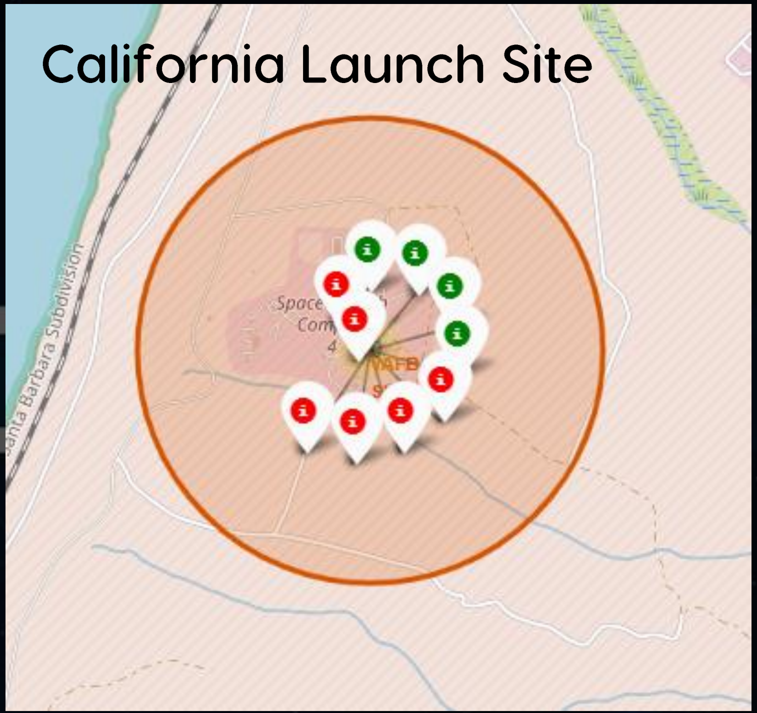
Launch Sites Proximities Analysis

All launch sites global map markers

We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

Markers showing launch sites with color labels

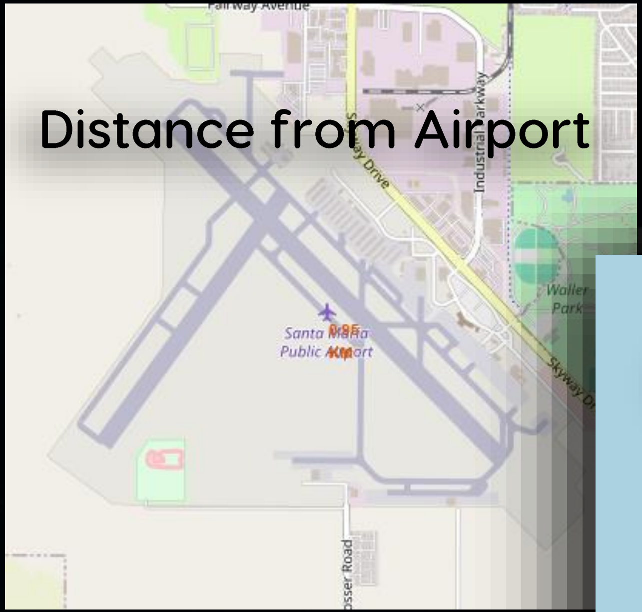


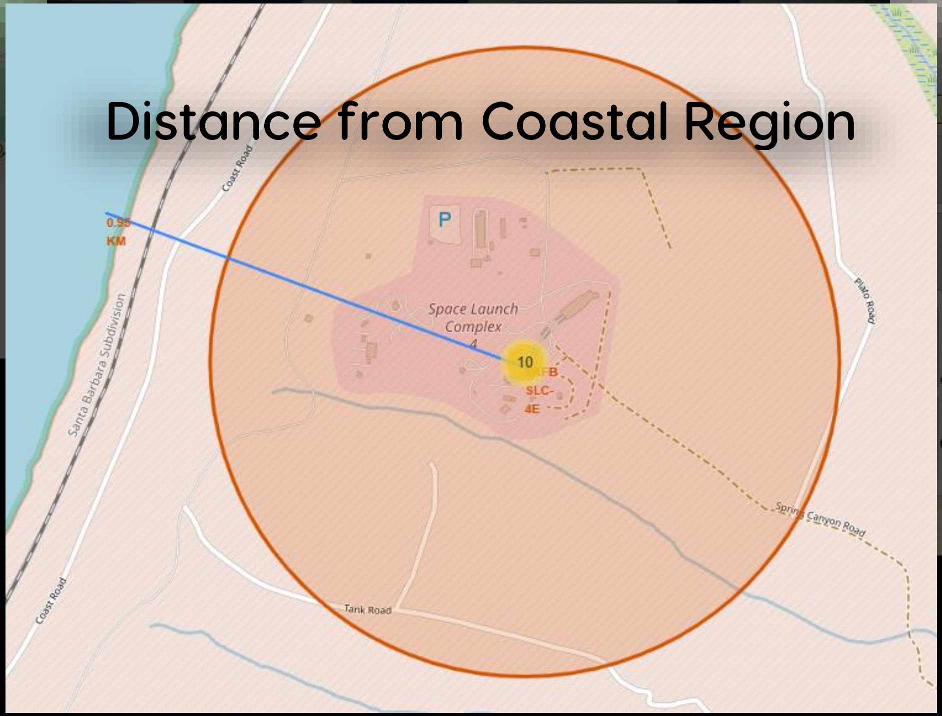


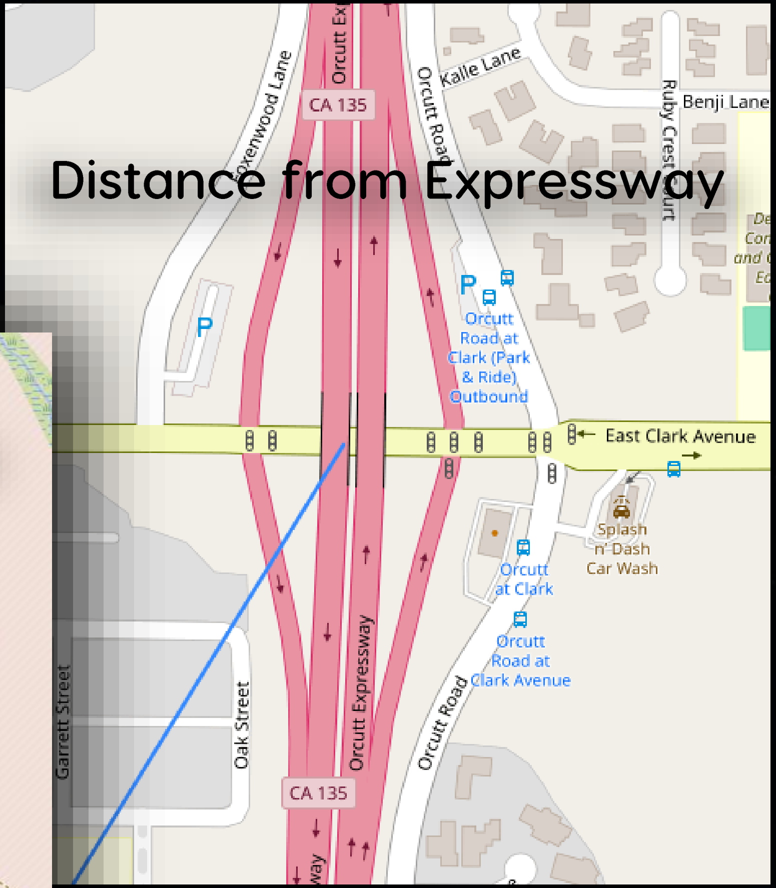
Florida Launch Sites

Green Marker shows successful Launches
and Red Marker shows Failures.

Launch Site Distance to landmarks



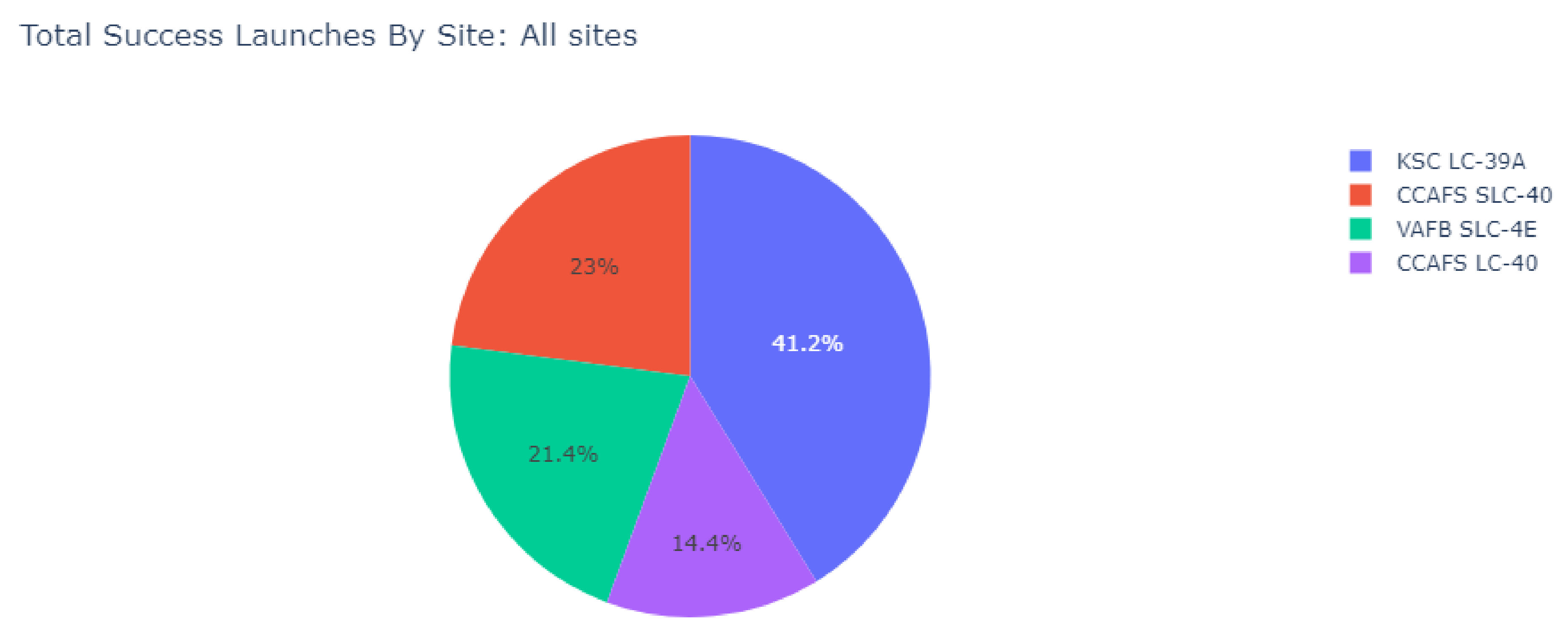




Section 4

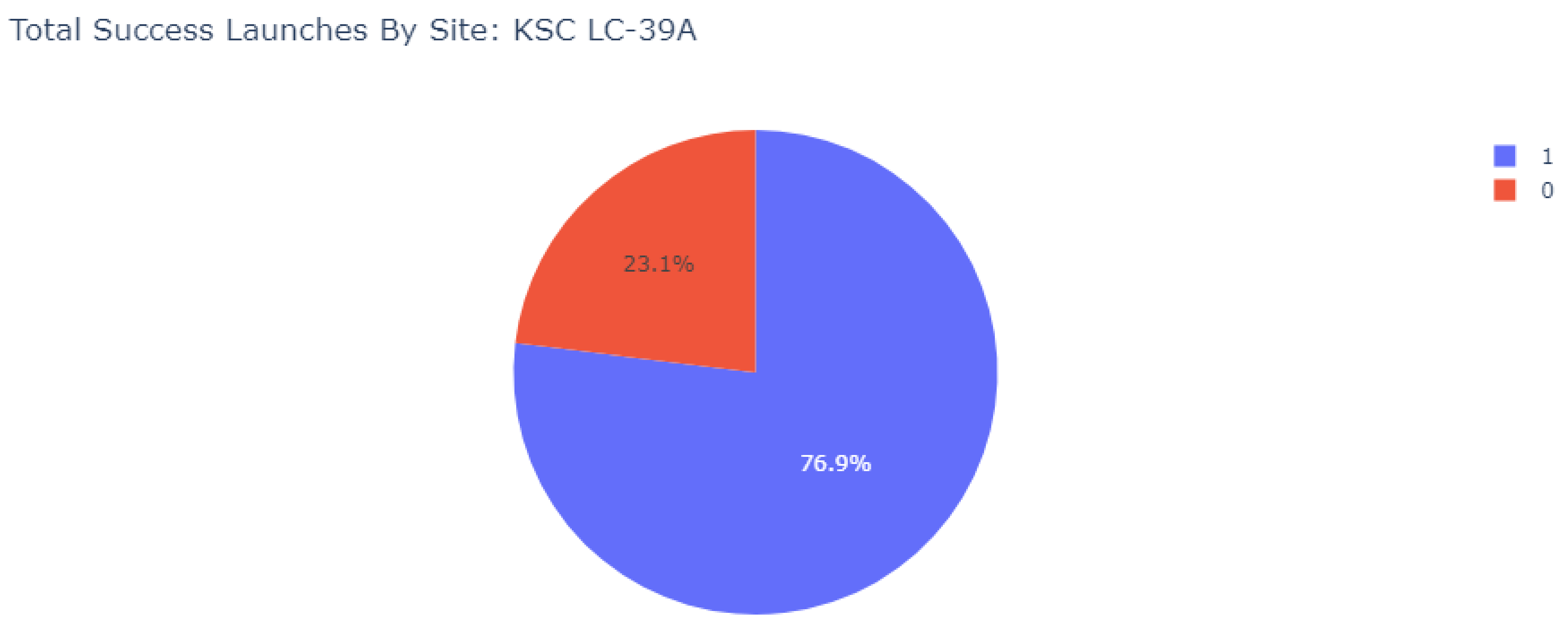
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



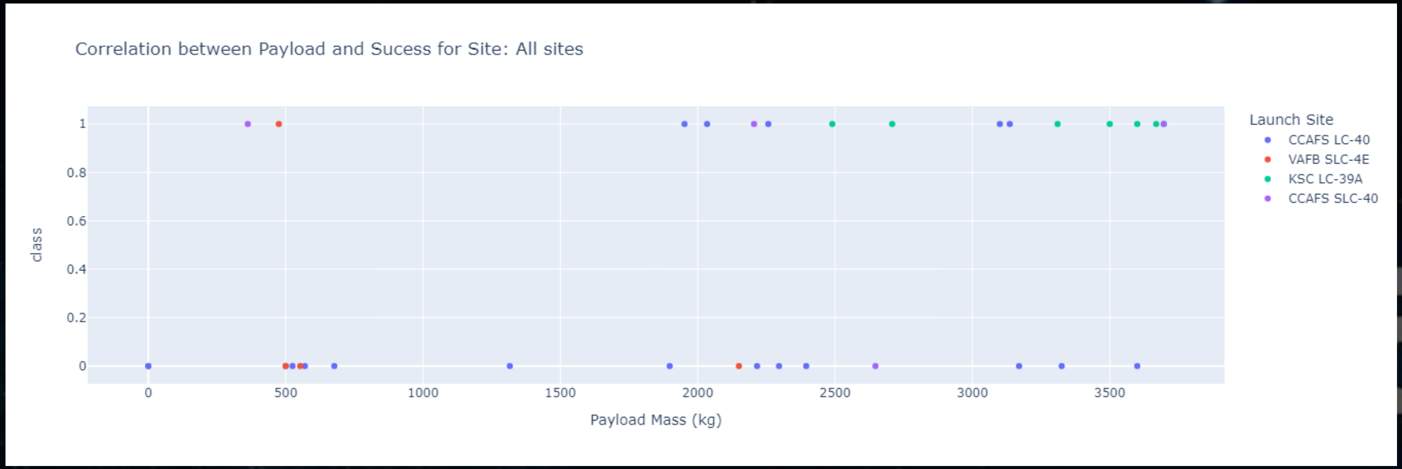
KSC LC-39A had the most successful launches comparatively among all of them.

Pie chart showing the Launch site with the highest launch success ratio



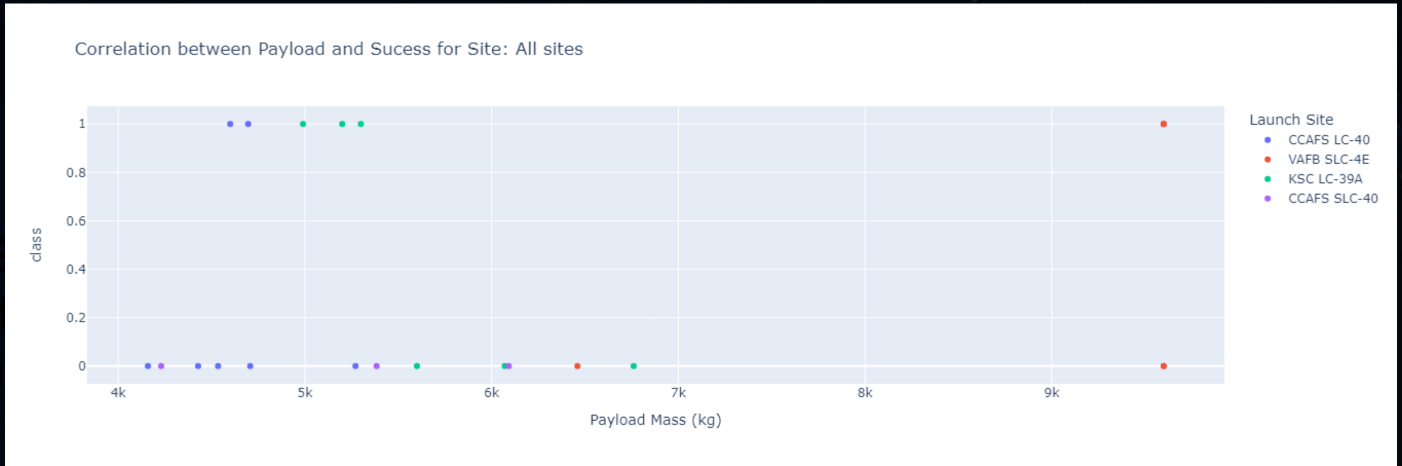
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate.

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider



Low Weighted Payload
0kg – 4000kg.

Heavy Weighted Payload
4000kg – 10000kg.



Section 5

Predictive Analysis (Classification)

Classification Accuracy

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

TASK 8

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
[29]: parameters = {'criterion': ['gini', 'entropy'],
           'splitter': ['best', 'random'],
           'max_depth': [2*n for n in range(1,10)],
           'max_features': ['auto', 'sqrt'],
           'min_samples_leaf': [1, 2, 4],
           'min_samples_split': [2, 5, 10]}

      tree = DecisionTreeClassifier(random_state=RANDOM_STATE)
```

```
[31]: tree_cv = GridSearchCV(tree , param_grid = parameters , cv = 10)
      print(tree_cv.fit(X_train , Y_train))
```

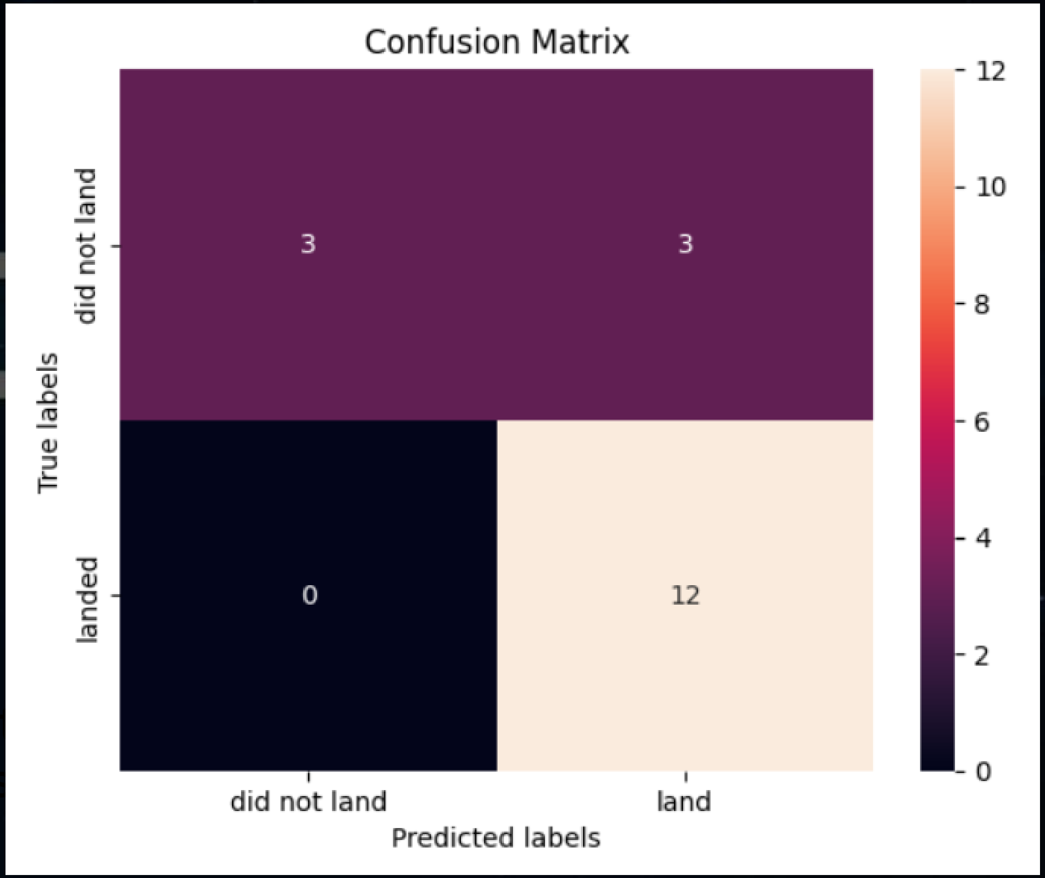
```
 0.83571429 0.75178571 0.80714286 0.76428571 0.8375     0.79464286
 0.83392857 0.74107143 0.79107143 0.75535714 0.80892857 0.78214286
 0.81964286 0.78392857 0.81964286 0.78392857 0.84821429 0.78392857
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
 0.82142857 0.7375     0.79285714 0.77857143 0.8375     0.79464286
```

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10,
'splitter': 'best'}
accuracy : 0.8767857142857143
```

Confusion Matrix

- The confusion matrix for the decision tree 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.



Conclusions

We can conclude that:

- 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

Thank you !