



IBM Developer  
SKILLS NETWORK

# Winning Space Race with Data Science

Ian Walker  
17<sup>th</sup> June 2024



# Outline

---

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

# Executive Summary

---

- Summary of methodologies
  - Data Collection from API
  - Data collection via webscraping
  - Data wrangling
  - EDA (Exploratory Data Analysis) with SQL and data visualization
  - Interactive visual analytics using Folium
  - Machine Learning Prediction
- Summary of all results
  - EDA result
  - Interactive analysis (screenshots)
  - ML Predictive Analytics results

# Introduction

---

- Project background and context
  - SpaceX advertises Falcon 9 rocket launches on its website with a cost of \$62 million, while other providers charge upwards of \$165 million per launch. Much of the savings come from SpaceX's ability to reuse the first stage of the rocket. Therefore, by predicting whether the first stage will land successfully, we can estimate the cost of a launch. This information can be valuable for alternate companies looking to compete with SpaceX for rocket launch contracts. The goal of this project is to create a machine learning pipeline to predict the successful landing of the first stage.
- Problems you want to find answers
  - What factors determine a successful rocket landing?
  - How are the features that determine the success rate of landing related?
  - Are any operating conditions needed to determine if a landing will be successful?



Section 1

# Methodology

# Methodology

---

## Executive Summary

- Data collection methodology:
  - Using SpaceX Rest API
  - Via web scraping Wikipedia
- Perform data wrangling
  - By hot encoding data fields and dropping the irrelevant data columns (data transformation)
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Creating scatter and bar graphs to discover patterns,
- Perform interactive visual analytics using Folium and Plotly Dash
  - Using Plotly and Folium
- Perform predictive analysis using classification models
  - Building & Evaluating classification models

# Data Collection

---

## Data Collection – Meaning & Basic Steps

Data collection is the process of gathering and measuring information on targeted variables in an established system, enabling one to answer relevant questions and evaluate outcomes.

- **Getting Data from API or Web Page**
- **Make a dataframe from it**
- **Filter dataframes as per requirement**
- **Export to flat file**
- [GitHub Notebook](#)

# Data Collection – SpaceX API

---

- **Getting Data from API or Web Page**

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
In [7]: response = requests.get(spacex_url)
```



# Data Collection – SpaceX API

---

- **Make a dataframe**

```
]:
```

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

Using the dataframe `data` print the first 5 rows

# Data Collection – SpaceX API

- Filter DF

```
[31]: # Hint data['BoosterVersion']!= 'Falcon 1'
      # data_falcon9['BoosterVersion']!= 'Falcon 9' returns true for all rows except 'Falcon 9' and running drop, drops those rows.
      data_falcon9.drop(data_falcon9[data_falcon9['BoosterVersion']!= 'Falcon 9'].index, inplace = True)
```

- Work with missing values

```
In [34]: # Calculate the mean value of PayloadMass column
      avg_payload_mass = data_falcon9["PayloadMass"].astype("float").mean(axis=0)
      # Replace the np.nan values with its mean value
      data_falcon9["PayloadMass"].replace(np.nan, avg_payload_mass, inplace=True)
```

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

```
In [35]: data_falcon9.isnull().sum()
```

Out[36]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block
4	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1
5	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1
6	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1
7	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1
8	5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1

# Data Collection – SpaceX API

- Export to file

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

```
[39]: data_falcon9.to_csv('csvs/dataset_part_1.csv', index=False)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
1	1	2010-06-04	Falcon 9	6123.547647058824	LEO	CCSFS SLC 40	None None	1	False	False	False		1	0	B0003	-80.577366	28.5618571
2	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False		1	0	B0005	-80.577366	28.5618571
3	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False		1	0	B0007	-80.577366	28.5618571
4	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False		1	0	B1003	-120.610829	34.632093
5	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False		1	0	B1004	-80.577366	28.5618571
6	6	2014-01-06	Falcon 9	3325.0	GTO	CCSFS SLC 40	None None	1	False	False	False		1	0	B1005	-80.577366	28.5618571
7	7	2014-04-18	Falcon 9	2296.0	ISS	CCSFS SLC 40	True Ocean	1	False	False	True		1	0	B1006	-80.577366	28.5618571
8	8	2014-07-14	Falcon 9	1316.0	LEO	CCSFS SLC 40	True Ocean	1	False	False	True		1	0	B1007	-80.577366	28.5618571
9	9	2014-08-05	Falcon 9	4535.0	GTO	CCSFS SLC 40	None None	1	False	False	False		1	0	B1008	-80.577366	28.5618571
10	10	2014-09-07	Falcon 9	4428.0	GTO	CCSFS SLC 40	None None	1	False	False	False		1	0	B1011	-80.577366	28.5618571
11	11	2014-09-21	Falcon 9	2216.0	ISS	CCSFS SLC 40	False Ocean	1	False	False	False		1	0	B1010	-80.577366	28.5618571
12	12	2015-01-10	Falcon 9	2395.0	ISS	CCSFS SLC 40	False ASDS	1	True	False	True	5e9e3032383ecb761634e7cb	1	0	B1012	-80.577366	28.5618571
13	13	2015-02-11	Falcon 9	570.0	ES-L1	CCSFS SLC 40	True Ocean	1	True	False	True		1	0	B1013	-80.577366	28.5618571
14	14	2015-04-14	Falcon 9	1898.0	ISS	CCSFS SLC 40	False ASDS	1	True	False	True	5e9e3032383ecb761634e7cb	1	0	B1015	-80.577366	28.5618571
15	15	2015-04-27	Falcon 9	4707.0	GTO	CCSFS SLC 40	None None	1	False	False	False		1	0	B1016	-80.577366	28.5618571
16	16	2015-06-28	Falcon 9	2477.0	ISS	CCSFS SLC 40	None ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	1	0	B1018	-80.577366	28.5618571
17	17	2015-12-22	Falcon 9	2034.0	LEO	CCSFS SLC 40	True RTLS	1	True	False	True	5e9e3032383ecb1267234e77	1	0	B1019	-80.577366	28.5618571

# Data Collection - Scraping

- Web scraping was performed using BeautifulSoup
- This was then parsed and turned into a DF
- [GitHub URL](#)

```
In [8]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html5lib')
```

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

```
In [10]: # Use soup.title attribute
tag_title = soup.title
tag_string_tag_title = tag_title.string
tag_string_tag_title
```

After you have filled in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

```
In [20]: df=pd.DataFrame(launch_dict)
```

```
In [21]: df.head()
```

```
Out[21]:
```

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

```
In [22]: df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

# Data Wrangling

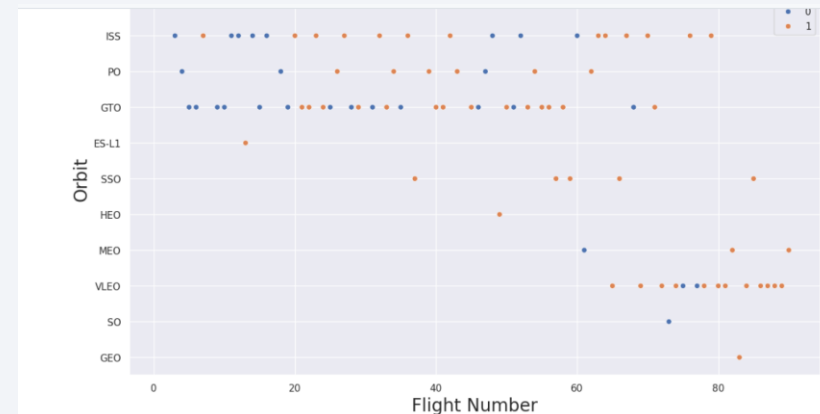
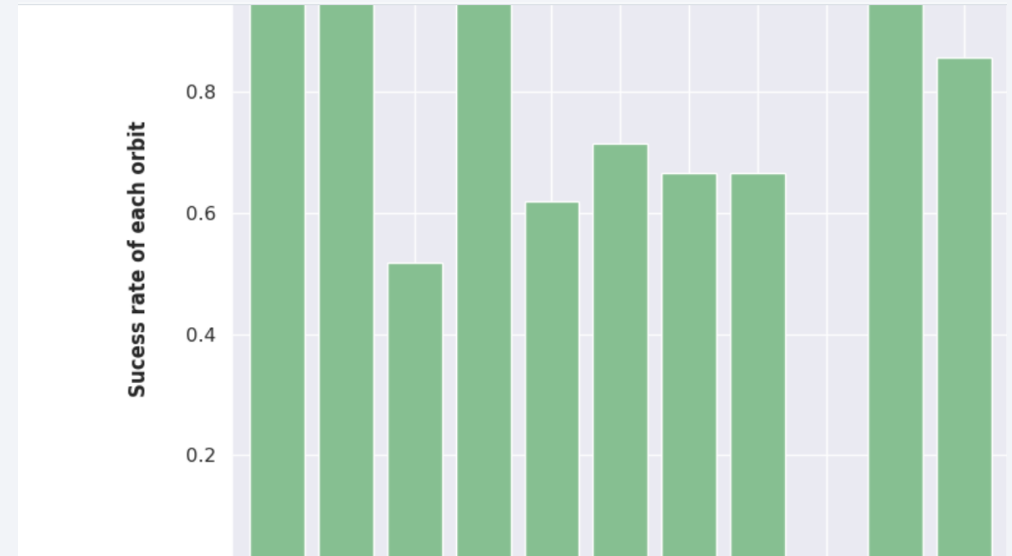
---

- EDA was performed and the training labels were determined.
- The number of launches for each site was calculated, and the number of occurrences of each orbit.
- A label was also created for landing outcome and this was exported to CSV.
- [GitHub URL](#)



# EDA with Data Visualization

- Data Visualization performed was:
- The relationship between flight number and launch site
- Payload and launch site, success rate of each orbit
- Flight number and orbit type
- Yearly trend of launch success.
- [GitHub URL](#)



# EDA with SQL

---

- The SpaceX dataset was loaded into a database directly inside Jupyter notebook.
- The following SQL queries were written and performed on the data:
  - **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 failed mission outcomes**
  - **The failed landing outcomes in drone ship, their booster version and launch site names**

[See the notebook on GitHub for more information.](#)

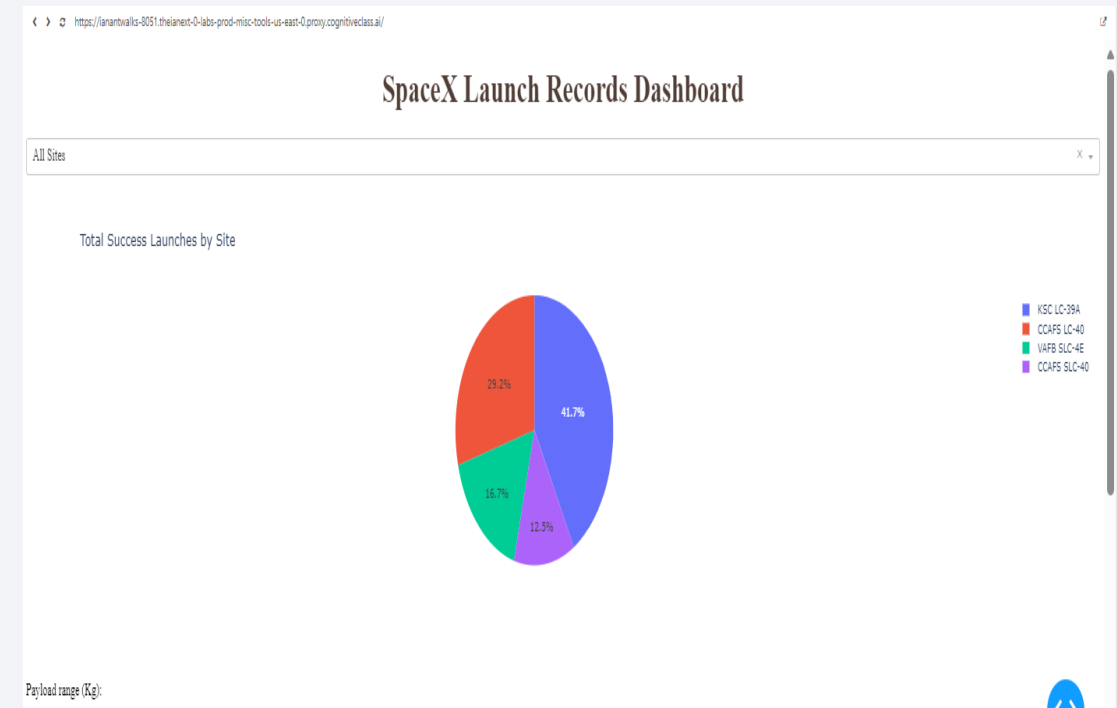
# Build an Interactive Map with Folium

---

- We marked all launch sites on an interactive Folium map. This map included markers, circles and lines to measure the success or failure of each launch for each launch site.
- Classes 0 and 1 were created to signify launch outcomes (e.g. class 0 was failure and class 1 was a success).
- Colour-coded markers were also added to marked clusters to help identify which launch sites had a high rate of success.
- Distances were calculated between a launch site and area proximities such as roads and rail. This data was then used to answer questions on the relationship between launch sites and their distance between them and cities and their overall location.
- [GitHub URL](#)

# Build a Dashboard with Plotly Dash

- An interactive dashboard was created with Plotly.
- For this, pie charts were created showing the number of launchers per site.
- Scatter graphs were also created to show relationships between Outcome and Payload Mass (Kg) for different booster version.
- [GitHub URL for .py program](#)



# Predictive Analysis (Classification)

---

- Data was loaded using NumPy and pandas. The data was then transformed and split into two sets, training and testing.
- Using GridSearchCV it allowed for building different machine learning models and for tuning of different hyperparameters.
- The metric used for the model was accuracy, this was improved using feature engineering and algorithm tuning.
- We also used this to discover which classification model performed best.
- [GitHub URL](#)



# 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 lower half of the image. The overall effect is dynamic and technological.

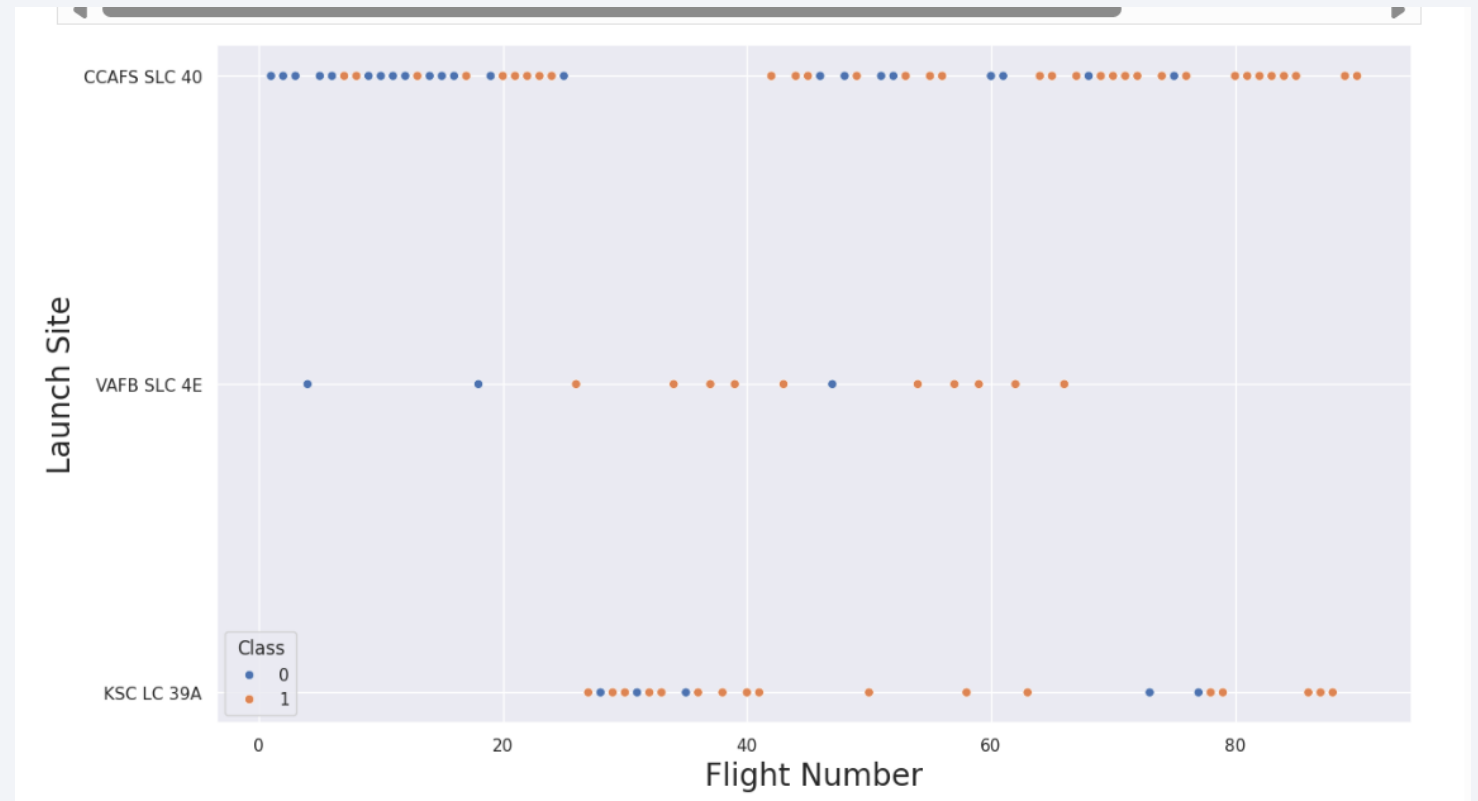
Section 2

# Insights drawn from EDA



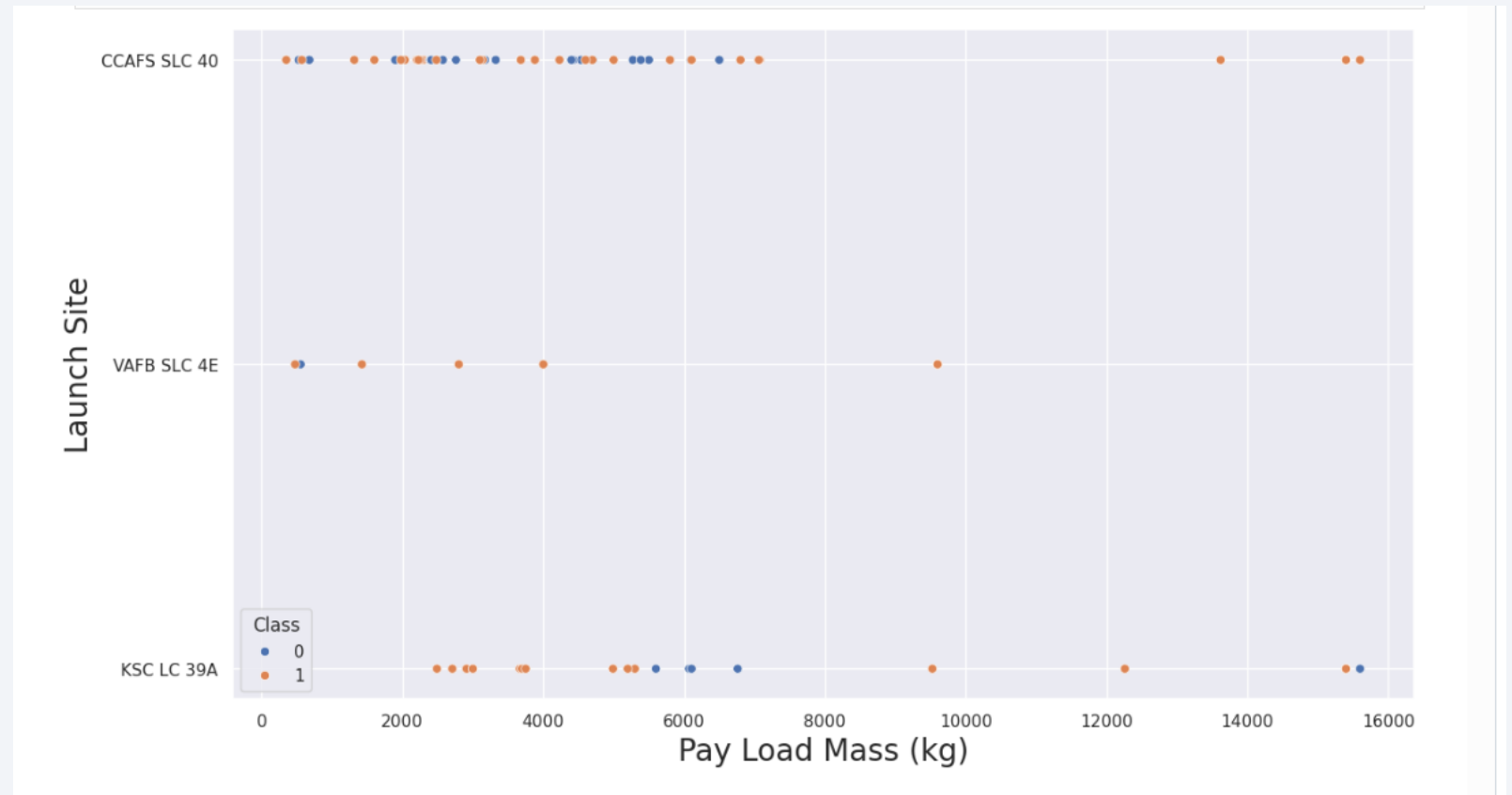
# Flight Number vs. Launch Site

- Based on the graph, it was determined that higher amount of launches a flight had at a launch site then the greater the launch success rate for that particular site.



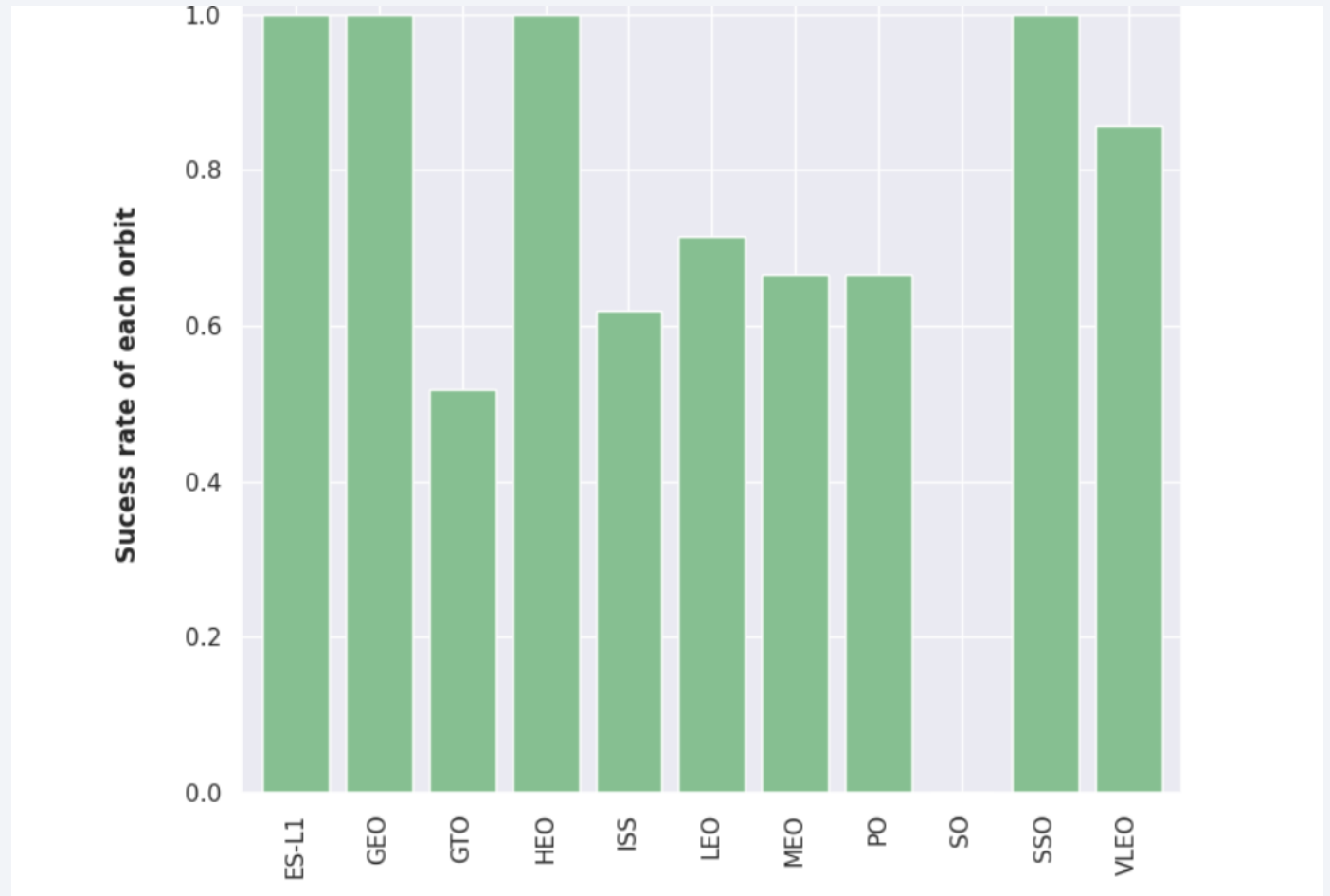
# Payload vs. Launch Site

- Based on the graph, the greater the payload mass then the higher the success rate for the rocket (e.g. CCAFS SLC 40).



# Success Rate vs. Orbit Type

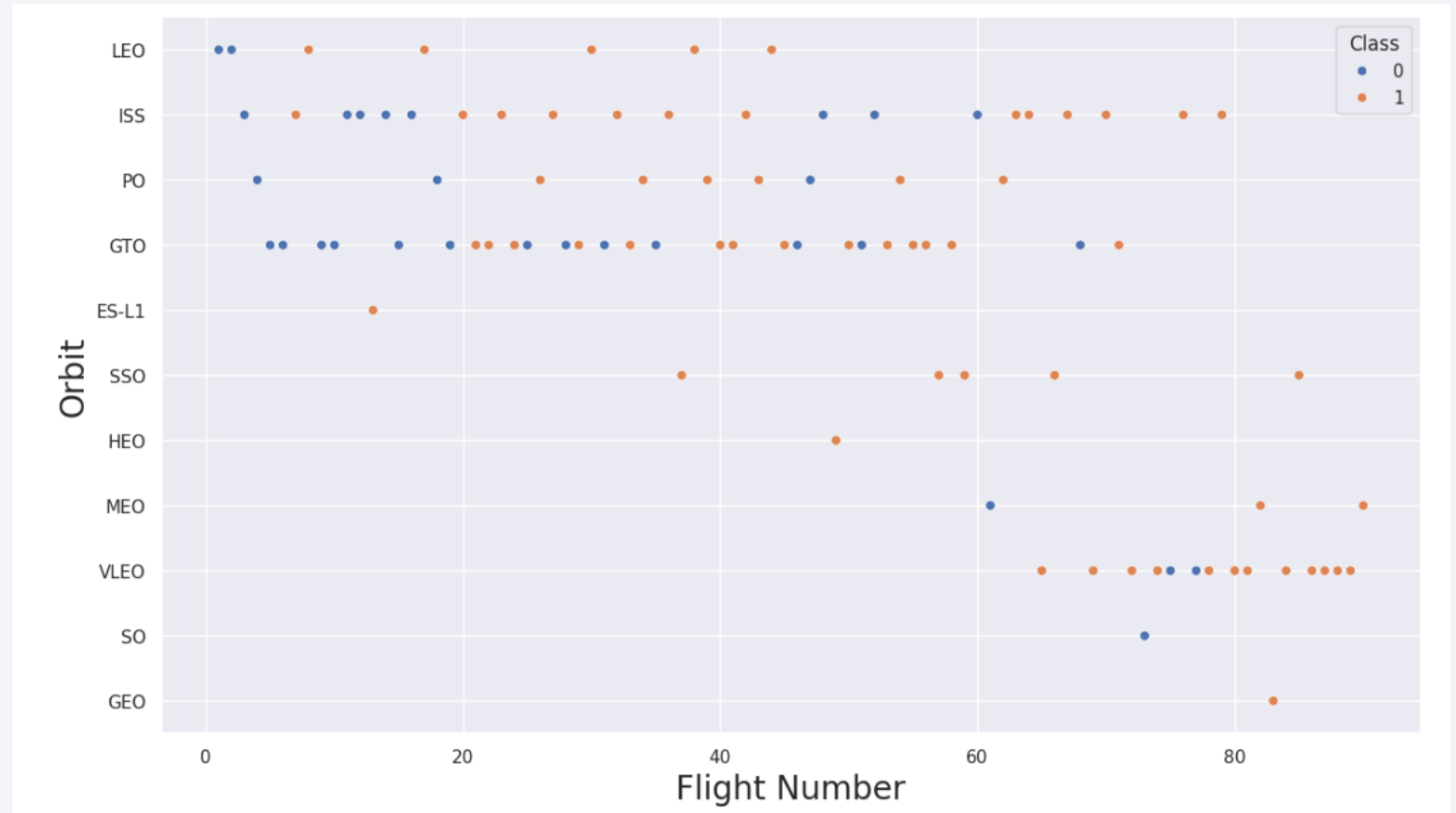
- Based on the graph, we can see that the most successful orbit types were ES-L1, GEO, HEO, SSO





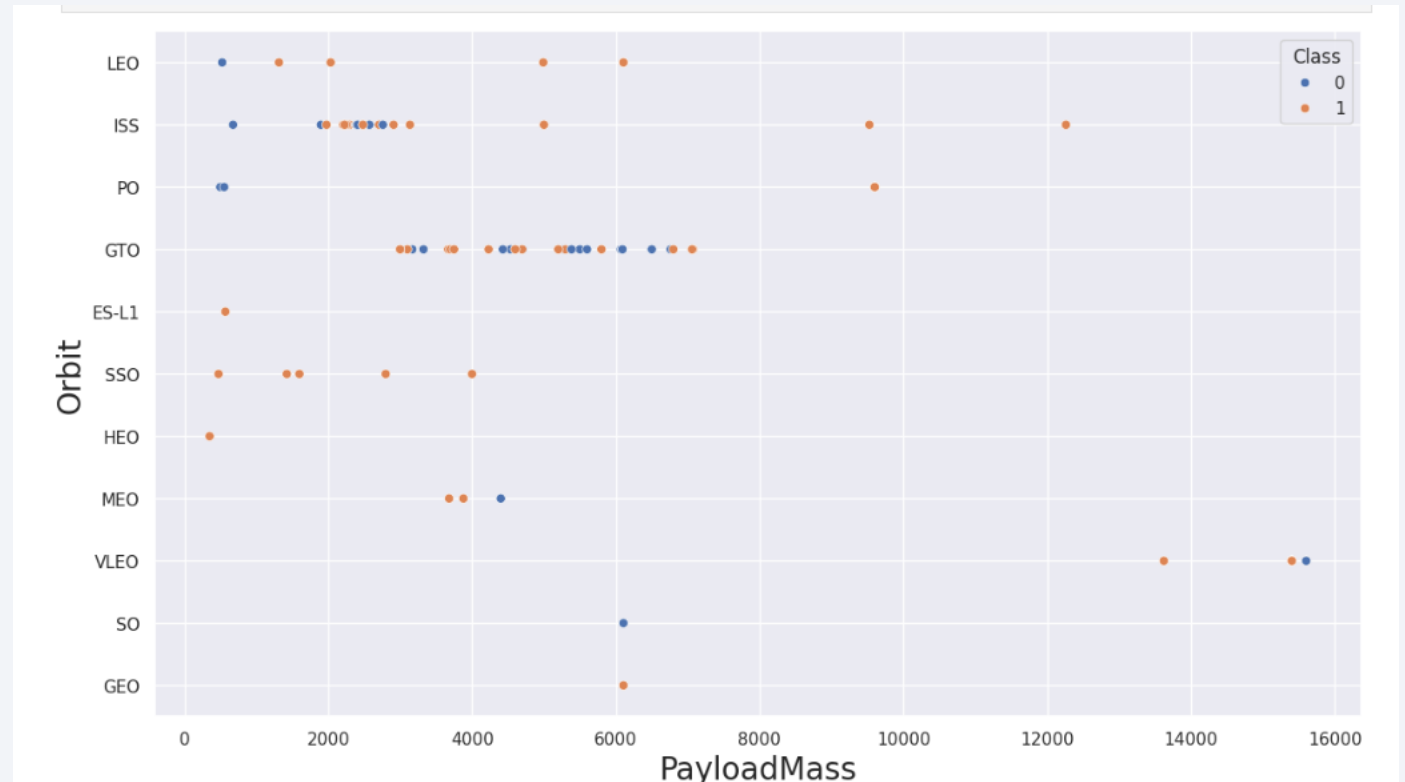
# Flight Number vs. Orbit Type

- Based on the graph, we can see that, for LEO orbit type, the success is related to the number of flights.
- For other orbit types, e.g. GTO, there is no such relationship.



# Payload vs. Orbit Type

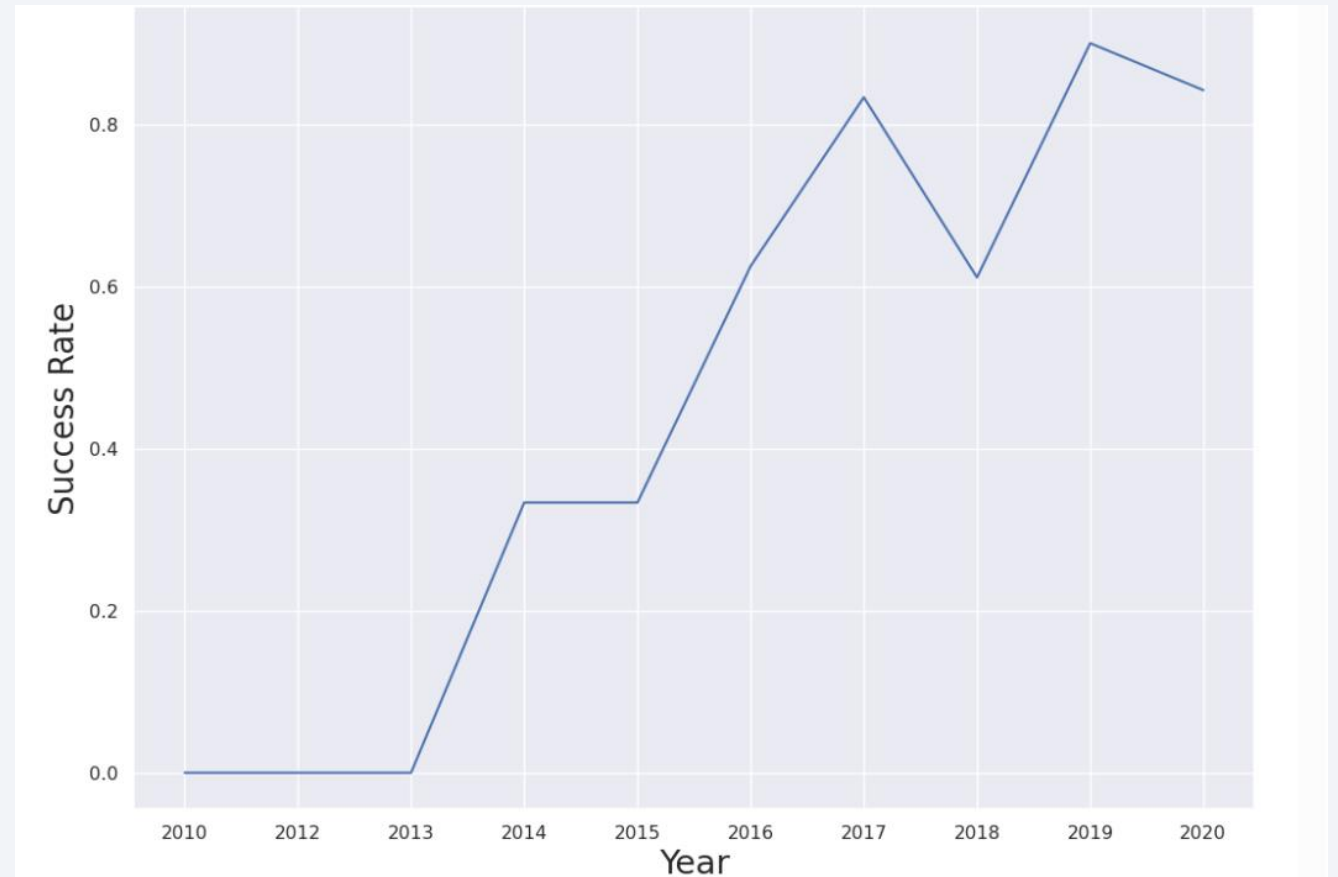
- From the graph we can see that when the PayloadMass (KG) is heavier then orbit types such as PO, LEO and ISS are more successful.



# Launch Success Yearly Trend

---

- From the graph we can see that between the years 2013 and 2020 the launch success rate continued to grow.



# All Launch Site Names

---

- This SQL query used **DISTINCT** to show only unique launch site data.

```
In [9]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTABLE;
```

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

```
Done.
```

```
Out[9]: Launch_Sites
```

```
CCAFS LC-40
```

```
VAFB SLC-4E
```

```
KSC LC-39A
```

```
CCAFS SLC-40
```

# Launch Site Names Begin with 'CCA'

- The following query was used, with results shown.

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

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

\* sqlite:///my\_data1.db  
Done.

Out[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



# Total Payload Mass

---

- The following query was used, with results shown.

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

```
[11]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)" FROM SPACEXTABLE WHERE CUSTOMER = 'NASA (CRS)';
```

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

```
Out[11]: Total Payload Mass by NASA (CRS)
```

45596
-------

# Average Payload Mass by F9 v1.1

---

- The following query was used, with results shown.

## Task 4

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

```
46]: %sql SELECT AVG(PAYLOAD_MASS_KG_) AS "Average Payload Mass by Booster Version F9 v1.1" FROM SPACEXTABLE \
      WHERE Booster_Version LIKE 'F9 v1.1';
```

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

```
Done.
```

```
46]: Average Payload Mass by Booster Version F9 v1.1
```

```
2928.4
```

# First Successful Ground Landing Date

---

- The following query was used, with results shown.

**\* Note: Not sure about this.**

## Task 5

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

*Hint: Use min function*

```
In [57]: %sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad" FROM SPACEXTABLE \
        WHERE 'Landing _Outcome' LIKE 'Success (ground pad)';
```

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

```
Out[57]: First Succesful Landing Outcome in Ground Pad
```

```
None
```

# Successful Drone Ship Landing with Payload between 4000 and 6000

---

- The following query was used, with results shown.

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 SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

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

Done.

Booster_Version
-----------------

F9 FT B1022
-------------

F9 FT B1026
-------------

F9 FT B1021.2
---------------

F9 FT B1031.2
---------------

# Total Number of Successful and Failure Mission Outcomes

- The following query was used, with results shown.

## Task 7

List the total number of successful and failure mission outcomes

```
In [27]: %sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEXTABLE WHERE MISSION_OUTCOME LIKE 'Success%';  
* sqlite:///my_data1.db  
Done.
```

```
Out[27]: Successful Mission  
100
```

```
In [17]: %sql SELECT COUNT(MISSION_OUTCOME) AS "Failure Mission" FROM SPACEXTABLE WHERE MISSION_OUTCOME LIKE 'Failure%';  
* sqlite:///my_data1.db  
Done.
```

```
Out[17]: Failure Mission  
1
```

# Boosters Carried Maximum Payload

- The following query was used, with results shown.

## Task 8

List the names of the booster\_versions which have carried the maximum payload mass. Use a subquery

```
In [20]: %sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEXTABLE \
WHERE PAYLOAD_MASS_KG_ =(SELECT MAX(PAYLOAD_MASS_KG_) FROM SPACEXTABLE);
```

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

```
Out[20]: Booster Versions which carried the Maximum Payload Mass
```

F9 B5 B1048.4

F9 B5 B1049.4

F9 B5 B1051.3

F9 B5 B1056.4

F9 B5 B1048.5

F9 B5 B1051.4

F9 B5 B1049.5

F9 B5 B1060.2

F9 B5 B1058.3

F9 B5 B1051.6

F9 B5 B1060.3

F9 B5 B1049.7

# 2015 Launch Records

---

- The following query was used, with results shown.

This used a combination of **WHERE**, **LIKE**, **AND** and **BETWEEN** to filter the required data.

## 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: SQLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.**

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTABLE WHERE DATE LIKE '2015-%' AND \
Landing_Outcome = 'Failure (drone ship)';
```

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

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

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

- The following query was used, with results shown.

This used a combination of **COUNT** (landing outcomes), **WHERE** (to filter **BETWEEN** dates).

- A **GROUP BY** clause was applied which was had an **ORDER BY** clause added to get the landing outcome data in descending order.

## Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
] : %sql SELECT Landing_Outcome, COUNT(*) as count \
      FROM SPACEXTABLE \
      WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' \
      GROUP BY Landing_Outcome \
      ORDER BY count DESC
```

\* sqlite:///my\_data1.db  
Done.

```
] :
```

Landing_Outcome	count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1



A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

Section 3

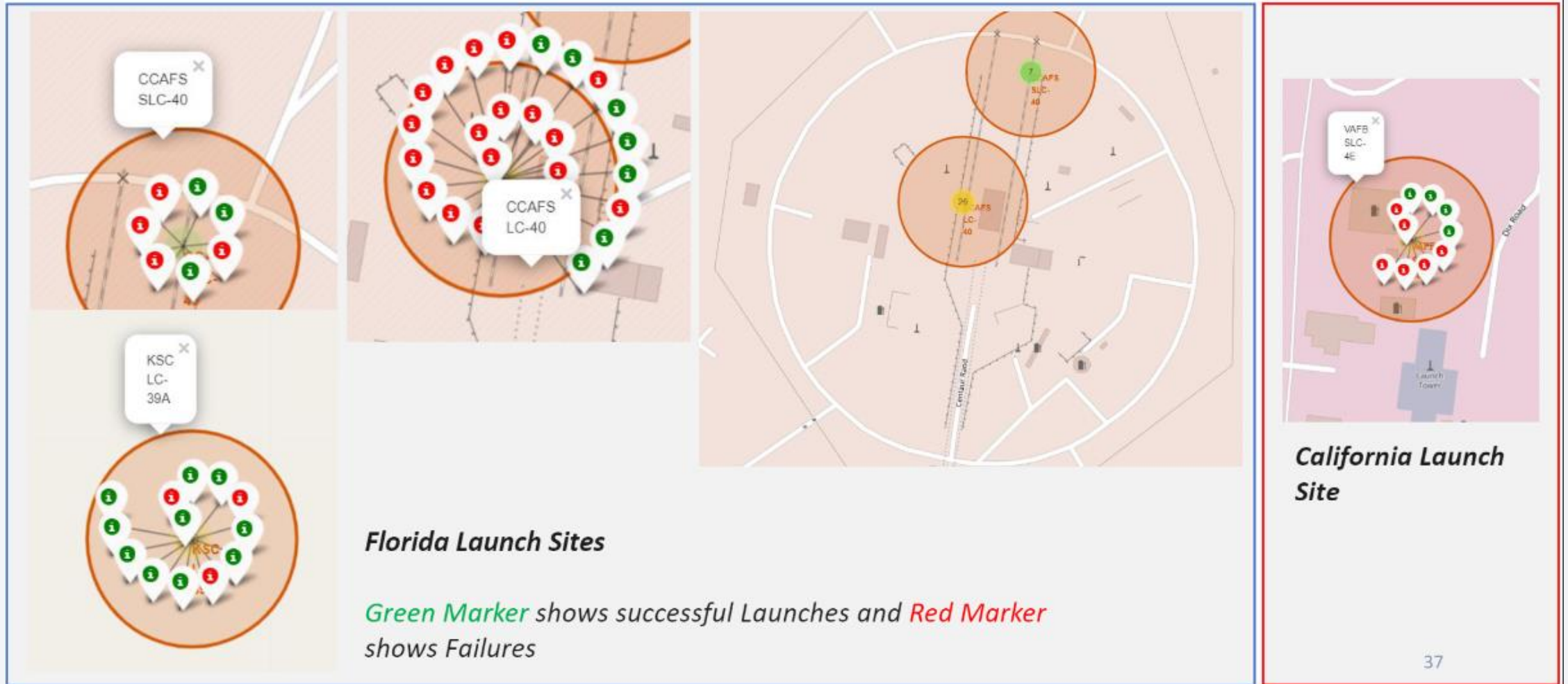
# Launch Sites Proximities Analysis

# Global markers of all launch sites in the USA

---



# Launch sites with colour labels













Section 4

# Build a Dashboard with Plotly Dash

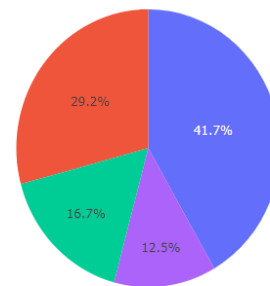
# Pie Chart showing success % per launch site

- We can see that KSC LC-39A has the most successful launches.

## SpaceX Launch Records Dashboard

All Sites

Total Success Launches by Site



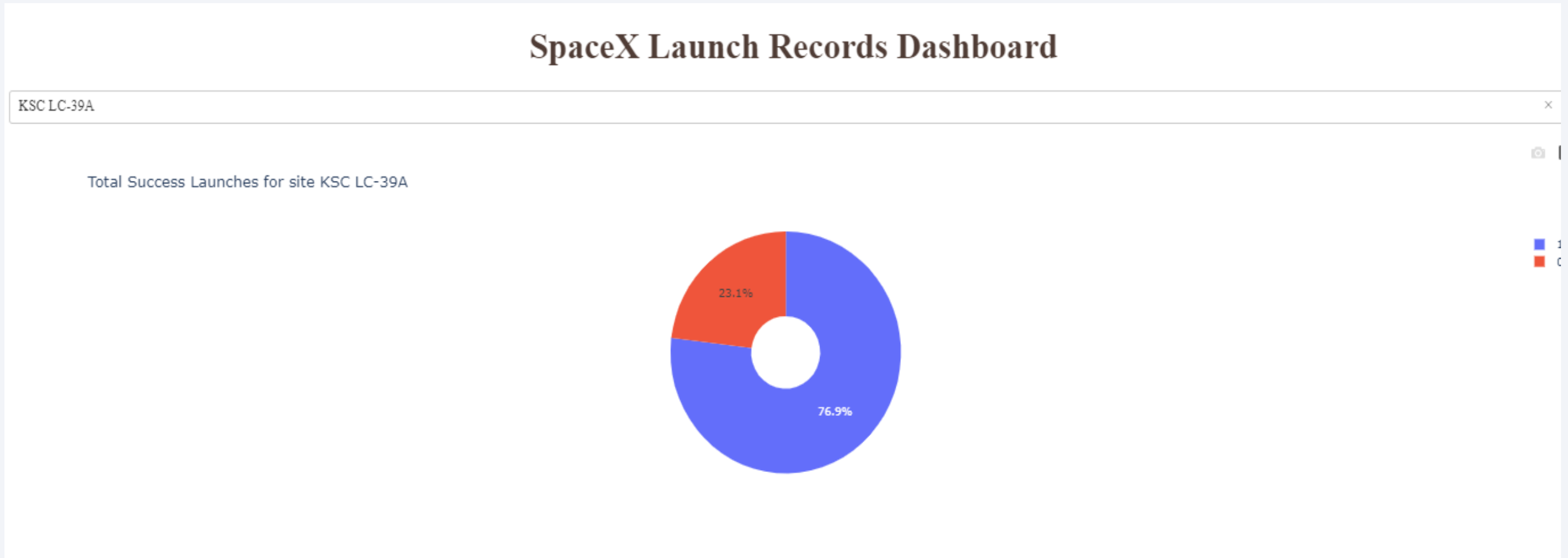
■ KSC LC-39A  
■ CCAFS LC-40  
■ VAFB SLC-4E  
■ CCAFS SLC-40

Payload range (Kg):

# Pie chart showing launch site with highest success ratio

---

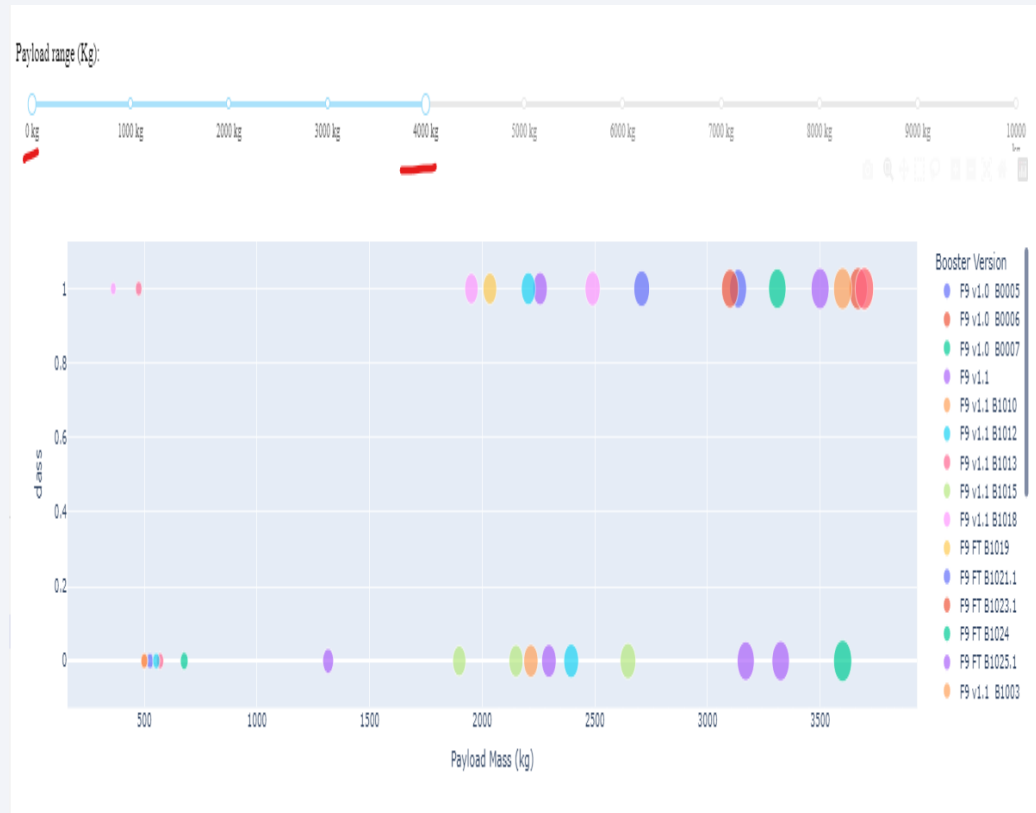
- We can see that KSC LC-39A has the highest launch success ratio (76.9%).





# Scatter plot of Payload vs Launch Outcome for All Sites

- We can see that success for low weighted payloads (0kg – 4000kg) is higher than heavy weighted payloads (4000kg – 10000kg).



Section 5

# Predictive Analysis (Classification)

# Classification Accuracy – Decision Tree is the most accurate

Find the method performs best:

```
In [47]: algorithms = {'KNN':knn_cv.best_score_, 'Decision Tree':tree_cv.best_score_, 'Logistic Regression':logreg_cv.best_score_,  
best_algorithm = max(algorithms, key=lambda x: algorithms[x])  
  
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 0.875

```
In [ ]: Here is best performing method
```

```
In [48]: algo_df = pd.DataFrame.from_dict(algorithms, orient='index', columns=['Accuracy'])
```

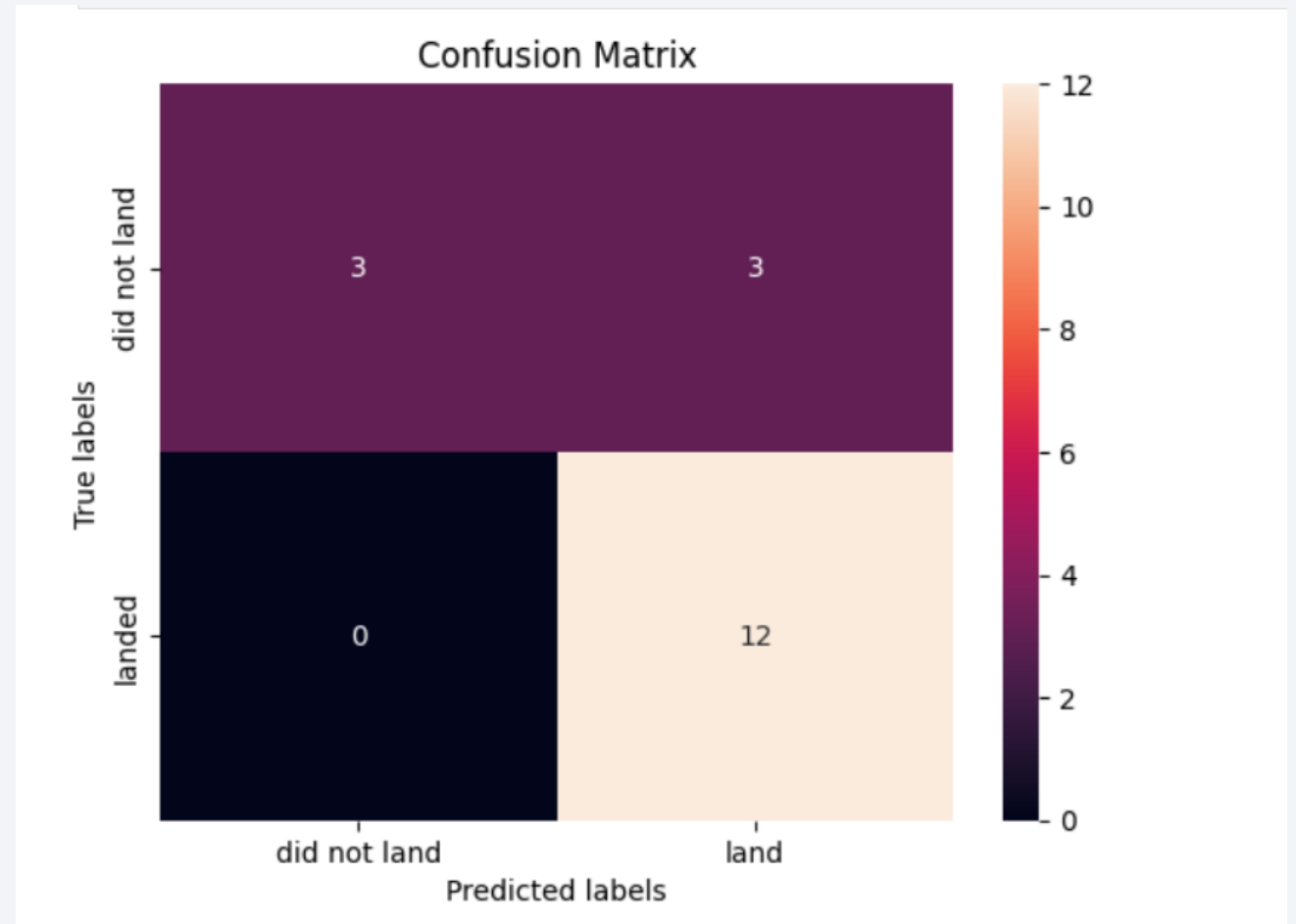
```
In [49]: algo_df.head()
```

```
Out[49]:
```

	Accuracy
KNN	0.848214
Decision Tree	0.875000
Logistic Regression	0.846429
SVM	0.848214

# Confusion Matrix

- Based on the confusion matrix for the decision tree classifier it shows that the classifier can correctly distinguish between different classes.
- The only issue is false positives. With some landing failures being marked as successful.



# Conclusions

---

From this exercise, we can conclude that:

- The larger the flight amount at a particular launch site then then the greater the rate of success.
- From 2013 to 2020 the rate of successful launch rates increased.
- The Orbits with the highest success rate were ES-L1, GEO, HEO, SSO and VLEO
- KSC-L39A was the most successful launch site overall with the most number of successful launches.
- The Decision tree classifier was the best machine learning model for this task.

# Appendix

---

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project



Thank you!

