



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Mohan Natarajan
15-08-2023



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 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 compete 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

- What factors determine if the rocket will land successfully?
- The interaction among-st various features that determine the success rate of a successful landing.
- What operating conditions needs to be in place to ensure a successful landing program.

The background image shows a large industrial facility, likely a port or shipping yard. Numerous shipping containers in various colors (blue, green, red, yellow) are stacked in organized rows both inside and outside a large building with a grid-like steel frame. Some trees are visible at the bottom right.

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 Data Studio

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 using BeautifulSoup.

The objective was to extract the launch records as HTML table, parse the table and convert it into 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.
- https://github.com/MohanBI/ds_tutorial/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** jupyter-labs-spacex-data-collection-api.ipynb
- Code:**

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillUp-Data/SpaceX/SpaceX%20Launch%20Site/Spacex%20-%20��%20Data%20API.json'
```
- Output:** The response status code is 200.
- Code:**

```
# Use json_normalize method to convert the json result into a dataframe
data=pd.json_normalize(response.json())
```
- Output:** The first 5 rows of the DataFrame are displayed, showing columns like static_fire_date_utc, static_fire_date_unix, net, window, rocket, success, and failures.

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda6995f709c1eb	False	None

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas data frame.

https://github.com/MohanBI/ds_tutorial/blob/main/jupyter-labs-webscraping.ipynb

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** https://github.com/MohanBI/ds_tutorial/blob/main/jupyter-labs-webscraping.ipynb
- Header:** ds_tutorial / jupyter-labs-webscraping.ipynb
- Toolbar:** Preview, Code, Blame, 1 lines (1 loc) · 38.2 KB
- Text:** Next, request the HTML page from the above URL and get a `response` object
- Section Header:** TASK 1: Request the Falcon9 Launch Wiki page from its URL
- Text:** First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.
- Code:** In [15]:

```
# use requests.get() method with the provided static_url
response=requests.get(static_url)
# assign the response to a object
html_response=response.text
```
- Text:** Create a `BeautifulSoup` object from the HTML `response`
- Code:** In [25]:

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup=BeautifulSoup(html_response,'html5lib')
```
- Text:** Print the page title to verify if the `BeautifulSoup` object was created properly
- Code:** In [26]:

```
# Use soup.title attribute
soup.title
```
- Text:** Out[26]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
- Section Header:** TASK 2: Extract all column/variable names from the HTML table header
- Text:** Next, we want to collect all relevant column names from the HTML table header
- Text:** Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please reference link towards the end of this lab
- Code:** In [27]:

```
# 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")
```

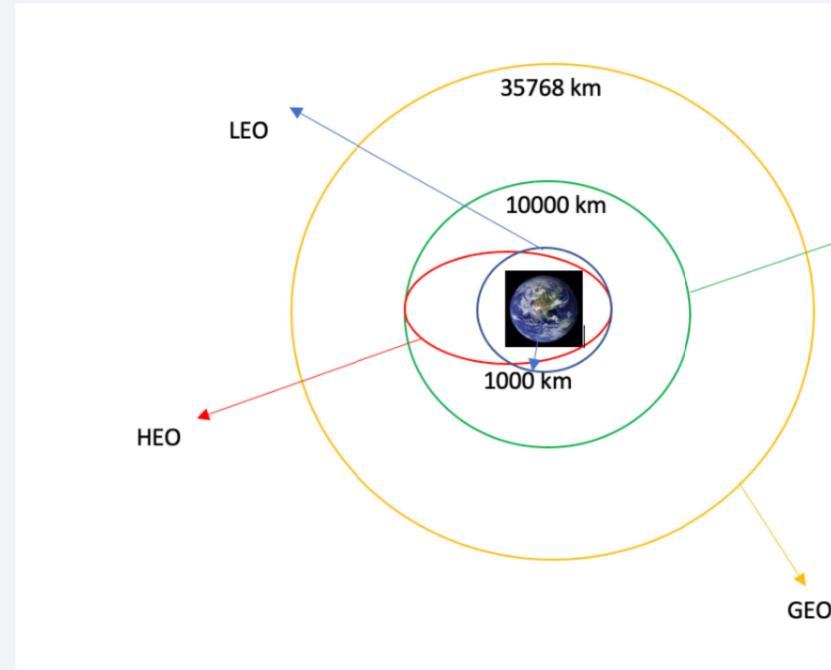
Data Wrangling

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

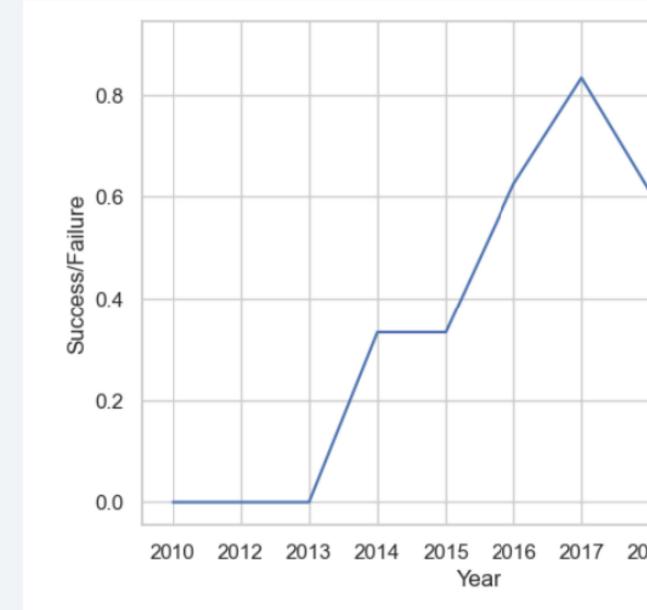
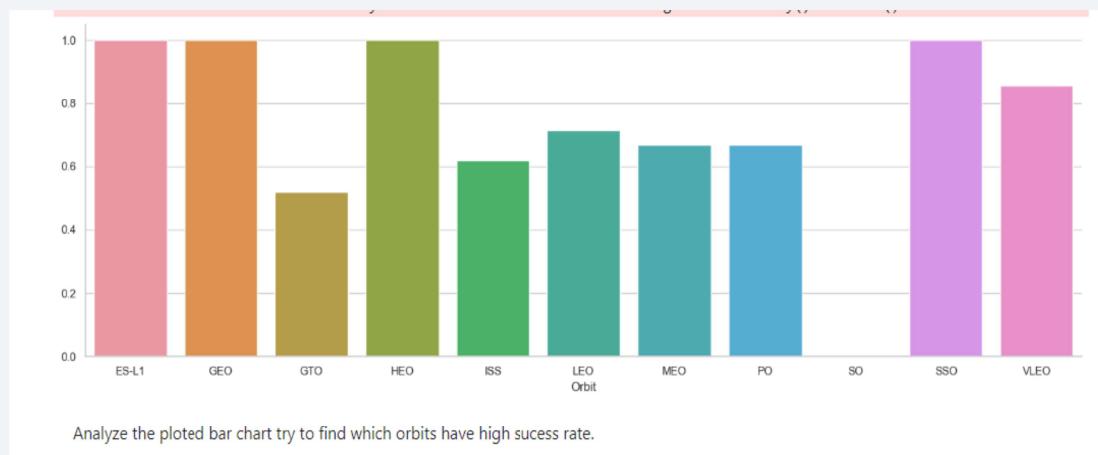
We created landing outcome label from outcome column and exported the results to csv.

[https://github.com/MohanBI/ds_tutorial/
blob/main/labs-jupyter-spacex-Data%20
wrangling.ipynb](https://github.com/MohanBI/ds_tutorial/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb)



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.



<https://github.com/MohanBI/Orbital/blob/main/jupyter-labs-edaz.ipynb>

EDA with SQL

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

https://github.com/MohanBI/ds_tutorial/blob/main/jupyter-labs-eda-sql-coursera_sqllite3.ipynb

Build an Interactive Map with Folium

We marked all launch sites, and added map objects such as markers, circles, and polygons 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 respectively, 0 for failure, and 1 for success.

Using the color-labeled marker clusters, we identified which launch sites have a relatively high success rate.

We calculated the distances between a launch site to its proximities. We are interested in some questions 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.

https://github.com/MohanBI/ds_tutorial/blob/main/spacex_dashboard.py

Predictive Analysis (Classification)

We loaded the data using numpy and pandas, transformed the data, split the data into training and testing.

We built different machine learning models and tune different hyperparameters using GridSearchCV.

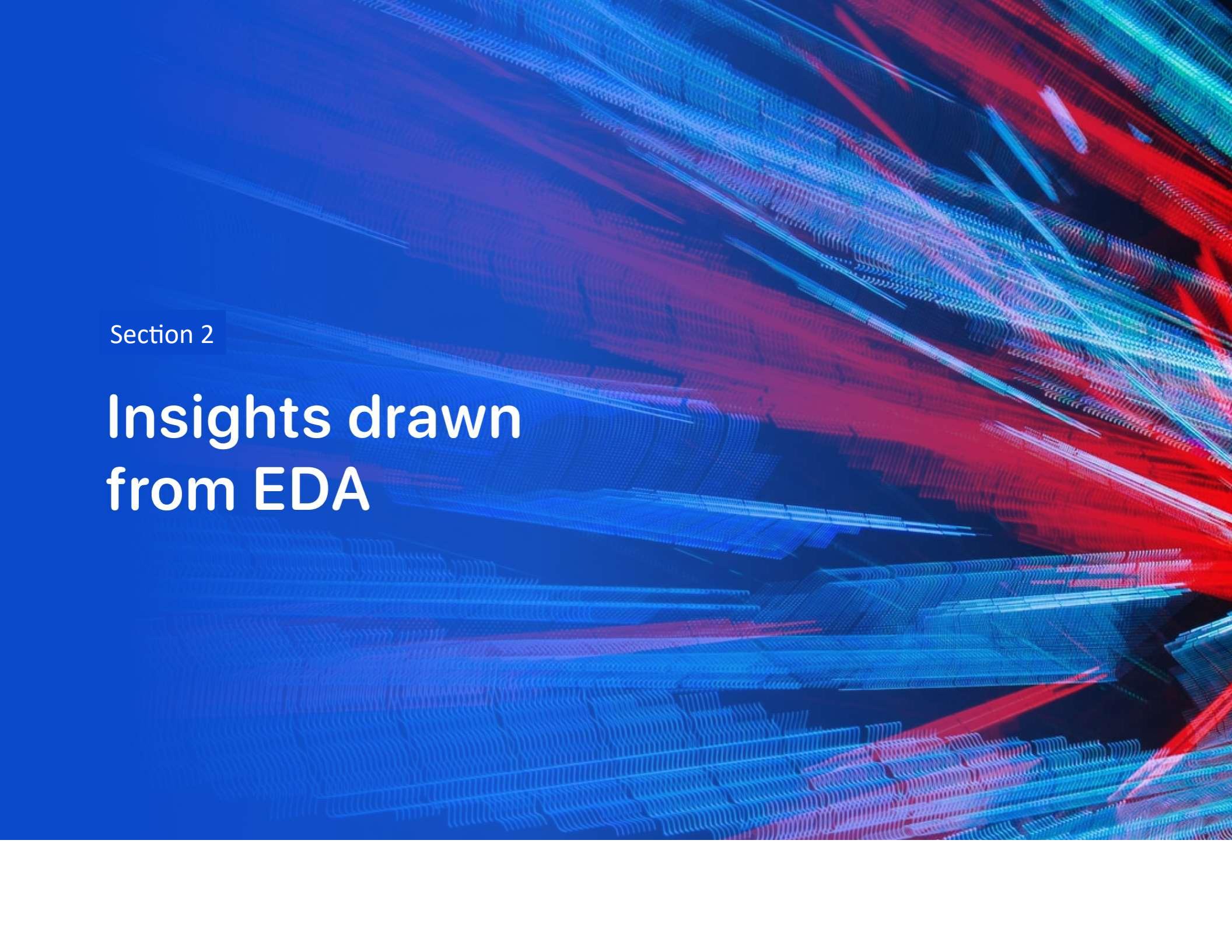
We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.

We found the best performing classification model.

https://github.com/MohanBI/ds_tutorial/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

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

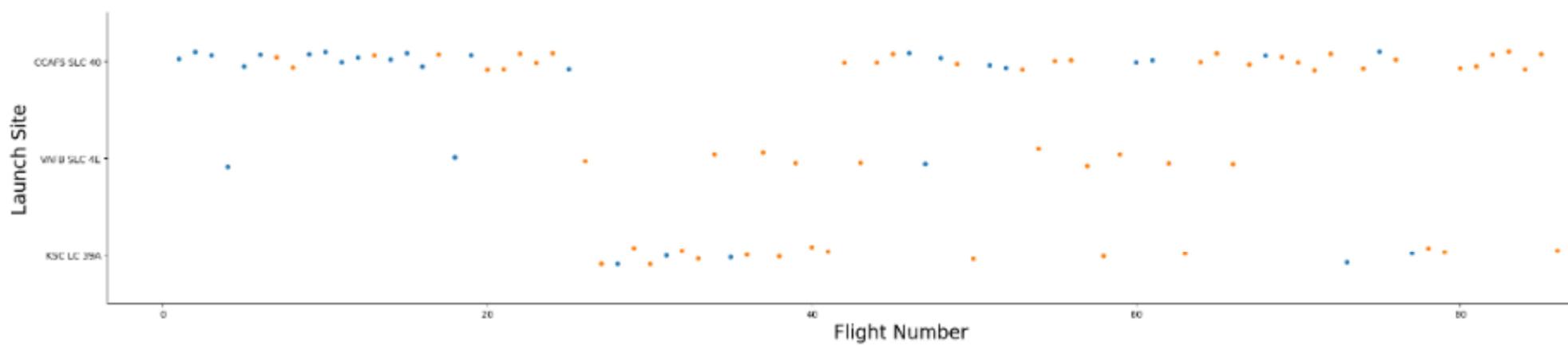
The background of the slide features a dynamic, abstract pattern of wavy, horizontal lines in shades of blue, red, and purple. These lines create a sense of depth and motion, resembling data streams or architectural structures. They are set against a dark, solid blue background.

Section 2

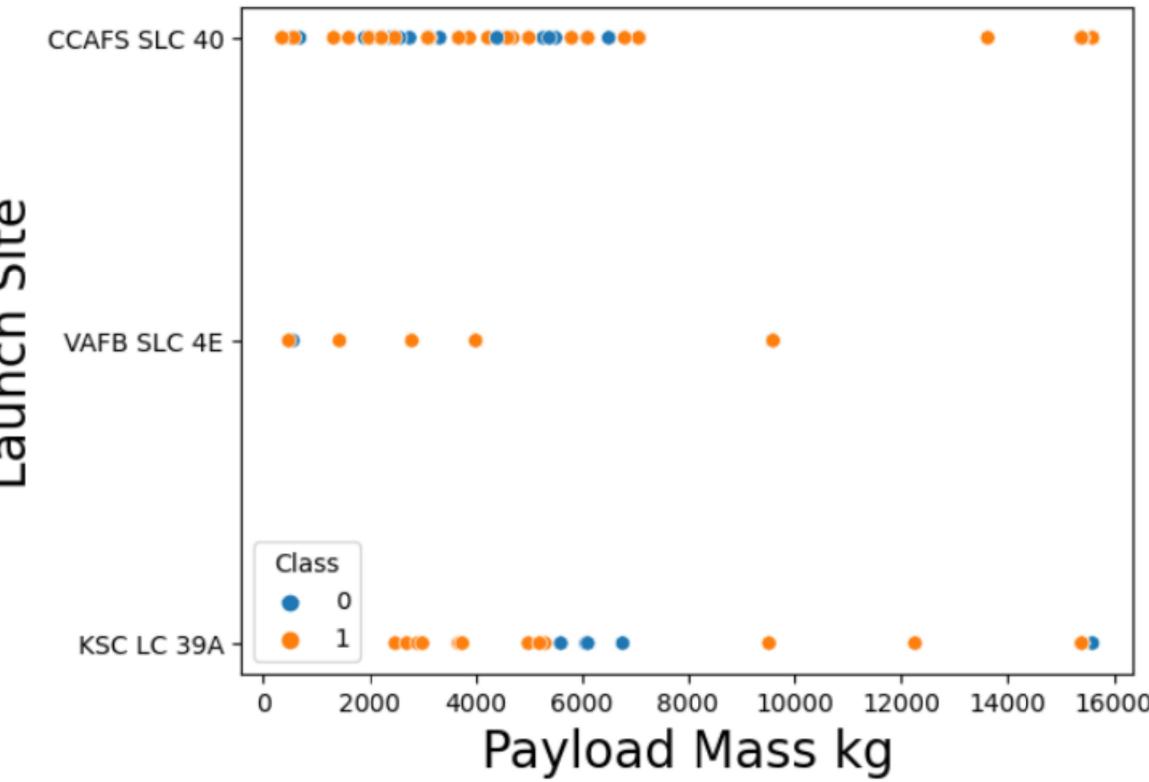
Insights drawn from EDA

Flight Number vs. Launch Site

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

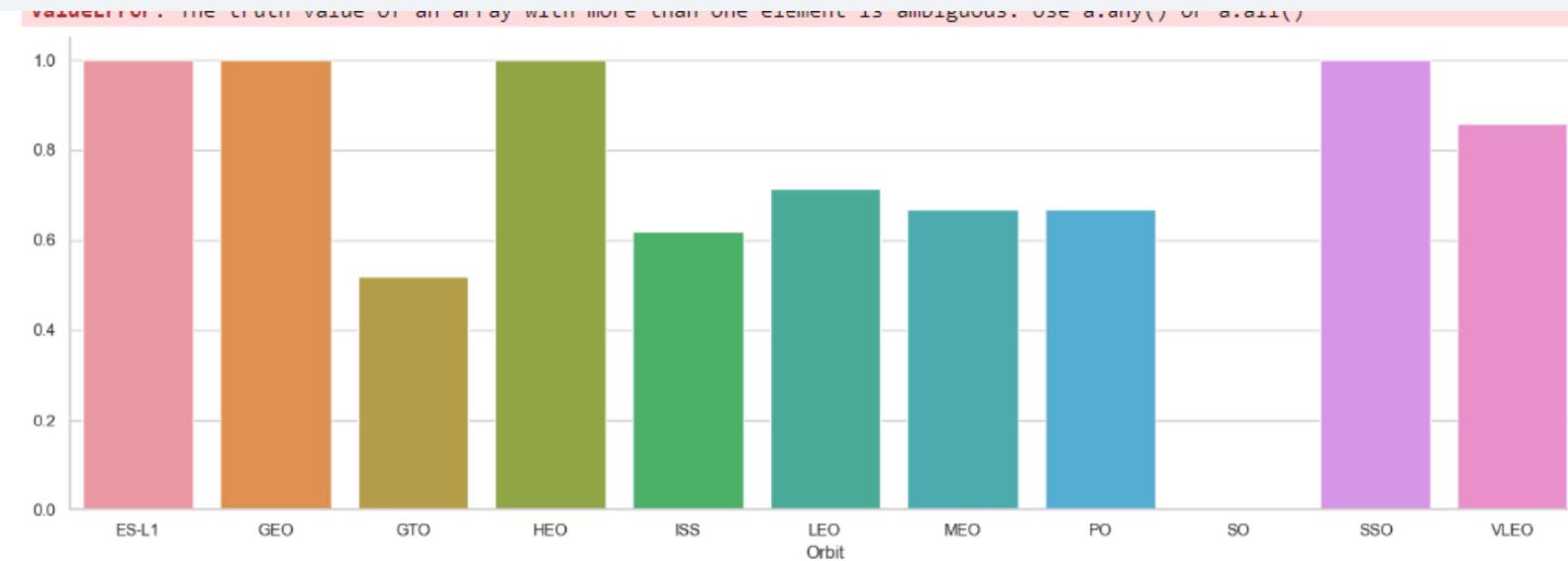


Payload vs. Launch Site



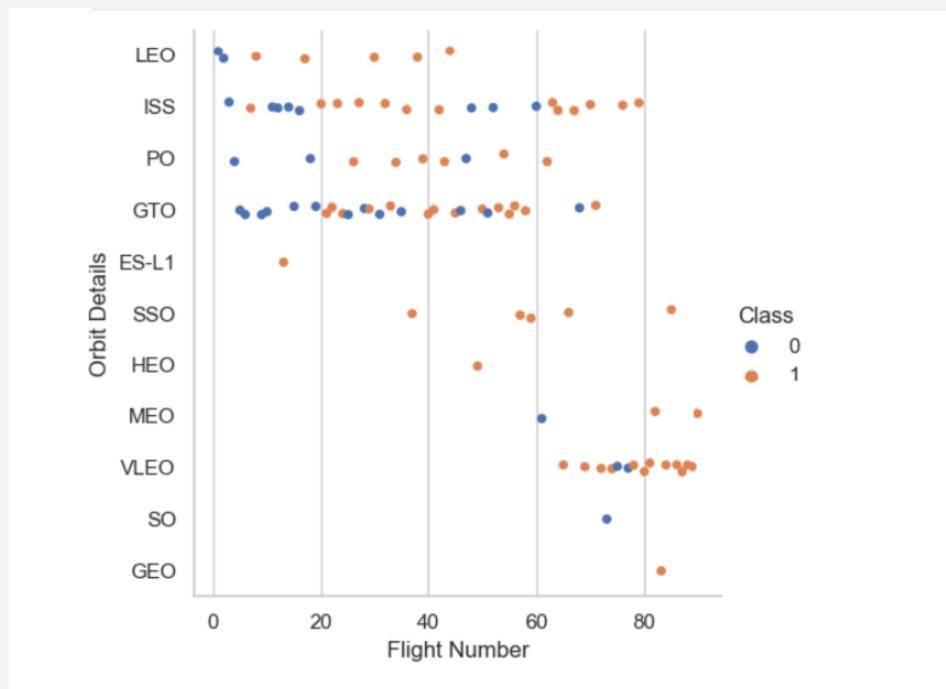
Success Rate vs. Orbit Type

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



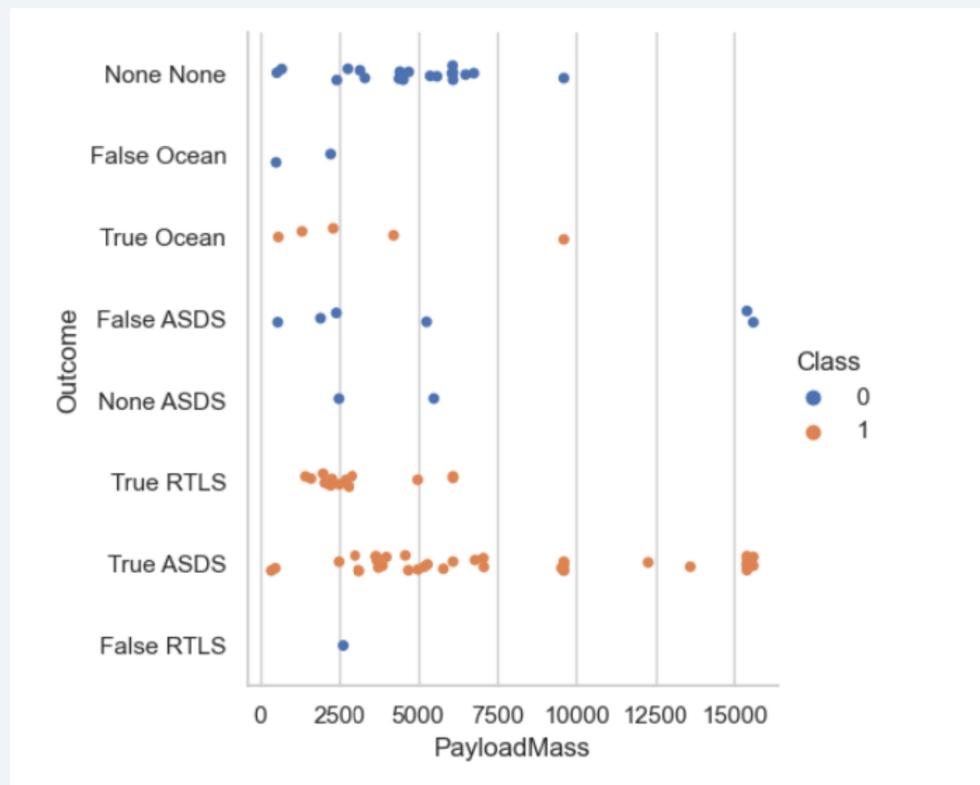
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe that in LEO orbit, success is related to the number of flights whereas in the GTO there is no relationship between flight number and the orbit.



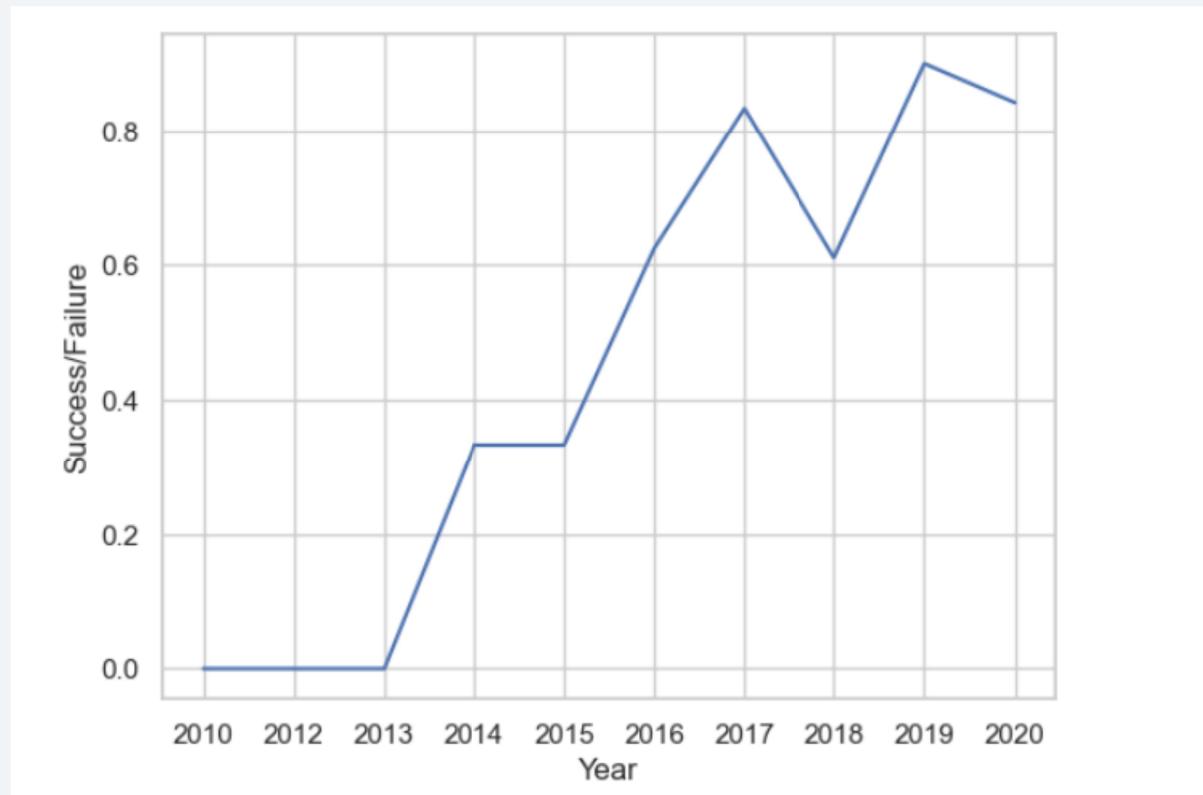
Payload vs. Orbit Type

- 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.

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

In [7]: `%sql select distinct Launch_Site from SPACEXTABLE`

* sqlite:///my_data1.db
Done.

Out[7]: [Launch_Site](#)

CCAFS LC-40

VAFB SLC-4E

KSC LC-39A

CCAFS SLC-40

Launch Site Names Begin with 'CCA'

Task 2

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

In [13]:

```
%sql select * from SPACEXTABLE where Launch_Site like 'CCA%' limit 5
```

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

Out[13]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	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 [22]: %sql select sum("PAYLOAD_MASS__KG_") as Total from SPACEXTABLE where customer='NASA (CRS)'  
* sqlite:///my_data1.db  
Done.  
Out[22]: Total  
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

```
In [27]: %sql select avg(Payload_Mass__KG_) as average_payload_mass from SPACEXTABLE where booster_Version='F9 v1.1'  
* sqlite:///my_data1.db  
Done.  
Out[27]: average_payload_mass  
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

In [31]:

```
%sql select min(date) as first_successful_landing from SPACEXTABLE where mission_outcome='Success'
```

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

Out[31]: **first_successful_landing**

2010-04-06

Successful Drone Ship Landing with Payload between 4000 and 6000

We used the **WHERE** clause to filter for boosters which have successfully landed 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

In [33]: `%sql select Booster_version from SPACEXTABLE where payload_mass_kg_ > 4000 and payload_mass_kg_ < 6000 and Landing_outcome`

* sqlite:///my_data1.db

Done.

Out[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 wildcard like '%' to filter for **WHERE** MissionOutcome was a success or failure.

Task 7

List the total number of successful and failure mission outcomes

In [38]:

```
%sql select mission_outcome,count(*) as total from SPACEXTABLE group by mission_outcome  
* sqlite:///my_data1.db  
Done.
```

Out[38]:

Mission_Outcome	total
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

```
In [45]: %sql select Booster_version from SPACEXTABLE where payload_mass_kg_ = (select max(payload_mass_kg_) from SPACEXTABLE)
```

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

```
Out[45]: Booster_Version
```

```
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

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.

```
In [56]: %sql select substr(Date,4,1) as Month,Landing_outcome,booster_version,launch_site from SPACEXTABLE where landing_outcome like '%Failure%' and substr(Date,7,4)='2015'  
* sqlite:///my_data1.db  
Done.  
Out[56]: Month Landing_Outcome Booster_Version Launch_Site  
5 Failure (drone ship) F9 v1.1 B1012 CCAFS LC-40  
5 Failure (drone ship) F9 v1.1 B1015 CCAFS LC-40
```

Rank Landing Outcomes Between 2010-06-04 and 2017

- 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 landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 20, in descending order.

In [71]:

```
%sql select Date,Landing_outcome,rank() over(partition by landing_outcome order by date desc) as rank  
* sqlite:///my_data1.db  
Done.
```

Out[71]:

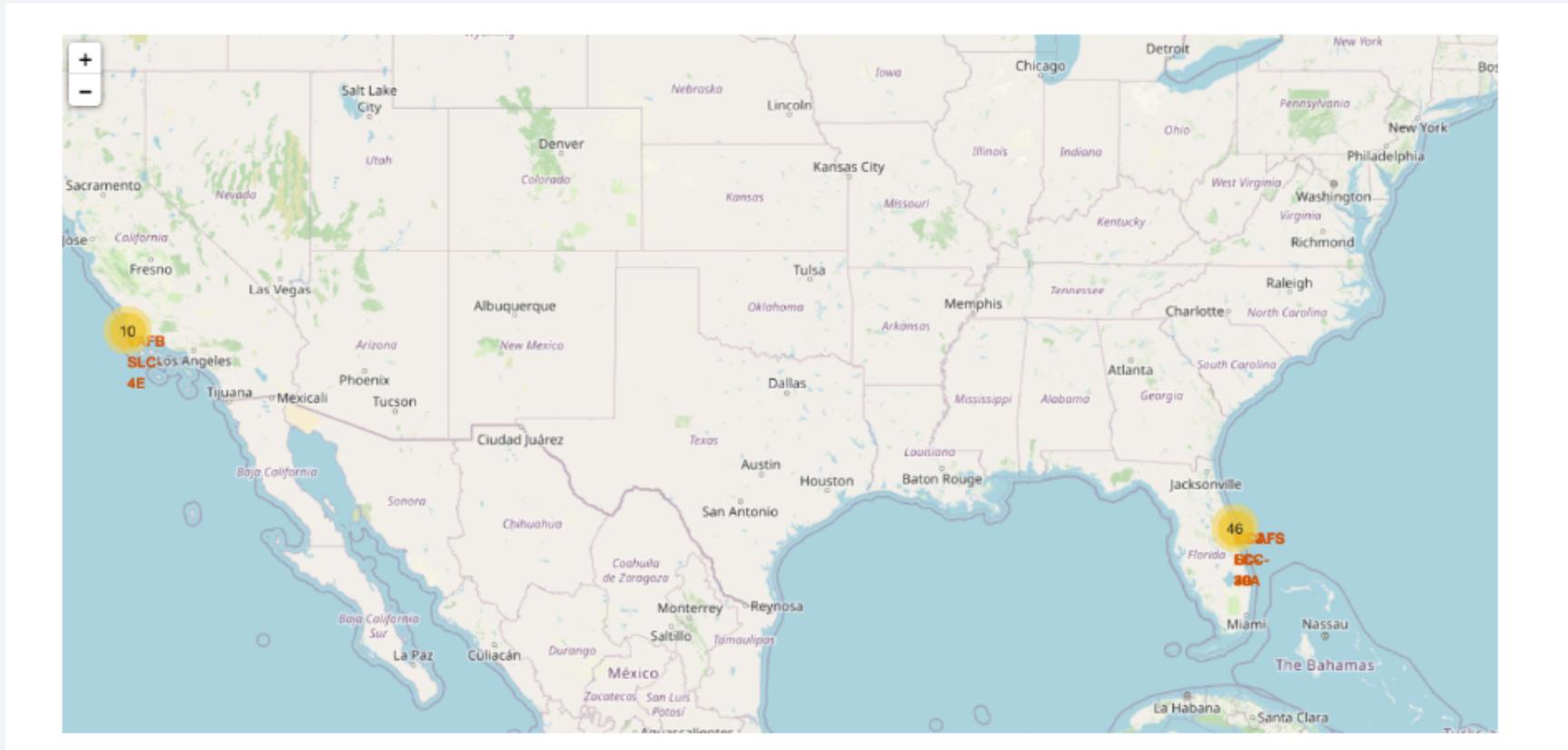
	Date	Landing_Outcome	rank
	2016-06-15	Failure (drone ship)	1
	2016-04-03	Failure (drone ship)	2
	2016-01-17	Failure (drone ship)	3
	2015-10-01	Failure (drone ship)	4
	2015-04-14	Failure (drone ship)	5
	2017-03-06	Success (ground pad)	1
	2017-02-19	Success (ground pad)	2
	2017-01-05	Success (ground pad)	3
	2016-07-18	Success (ground pad)	4
	2015-12-22	Success (ground pad)	5

The background of the slide is a photograph taken from space at night. It shows the curvature of the Earth against the dark void of space. City lights are visible as numerous small white and yellow dots, primarily concentrated in coastal and urban areas. The atmosphere appears as a thin blue layer, and the clouds below are dark and textured.

Section 3

Launch Sites Proximities Analysis

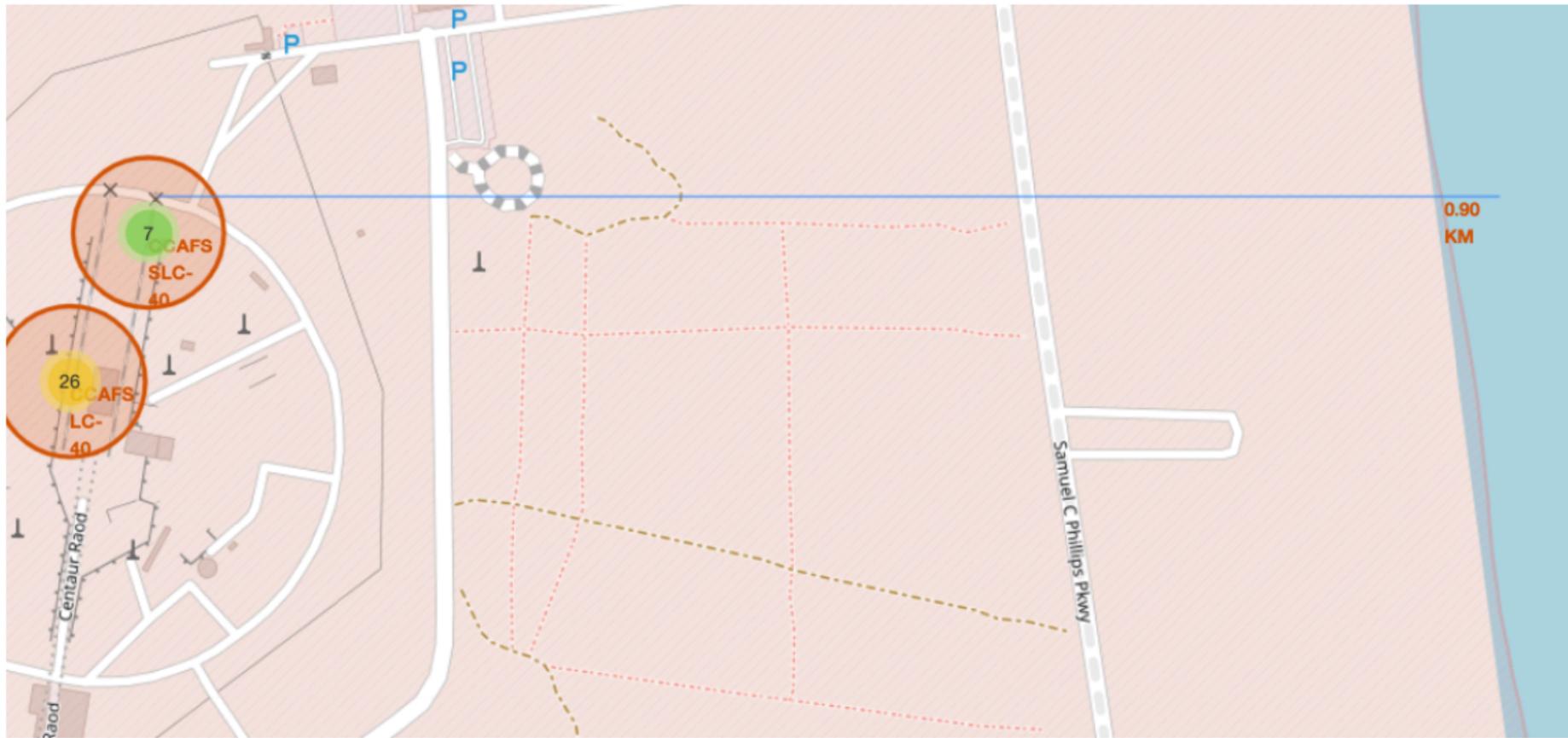
All launch sites global map markers



Markers showing launch sites with color labeling



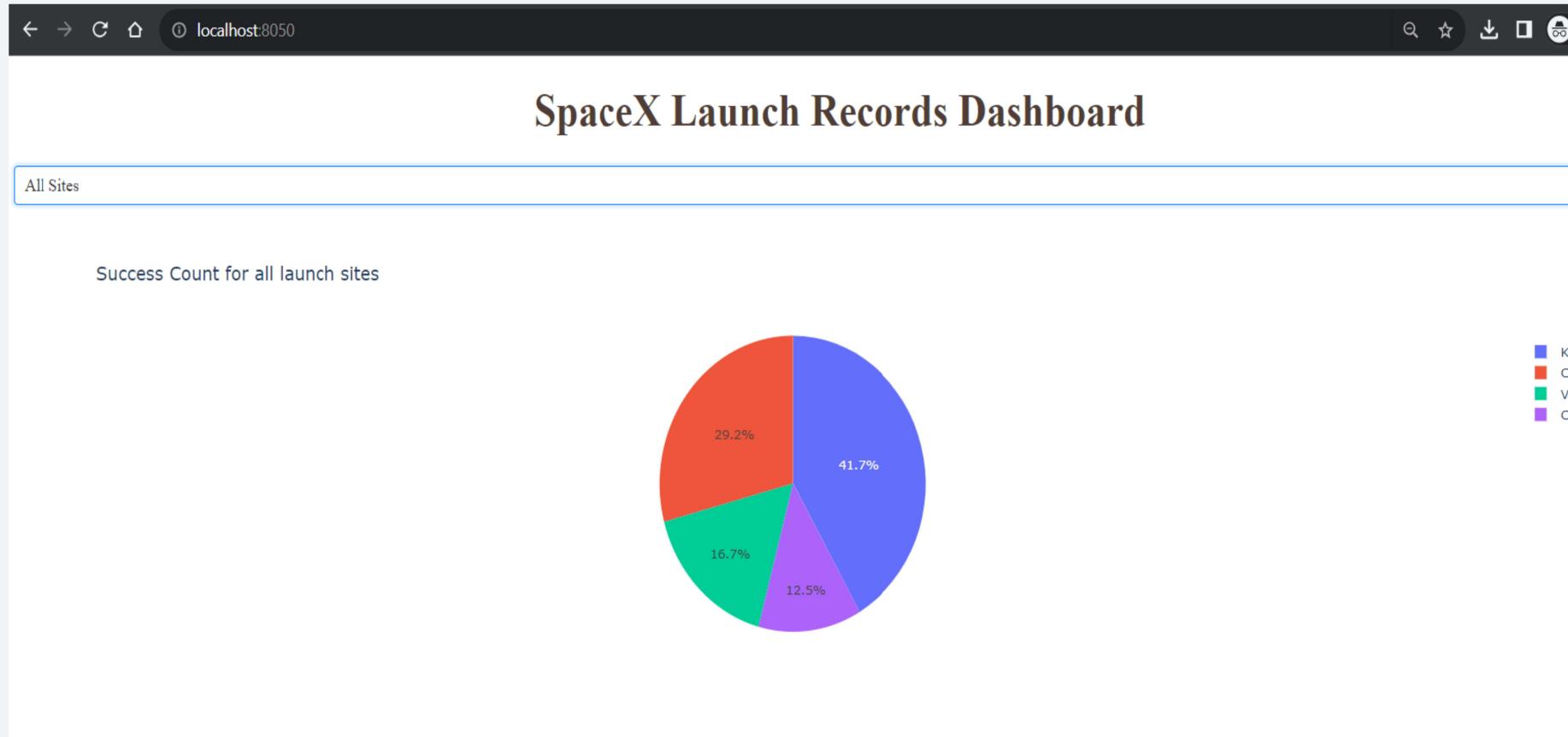
Launch Site distance to landmarks



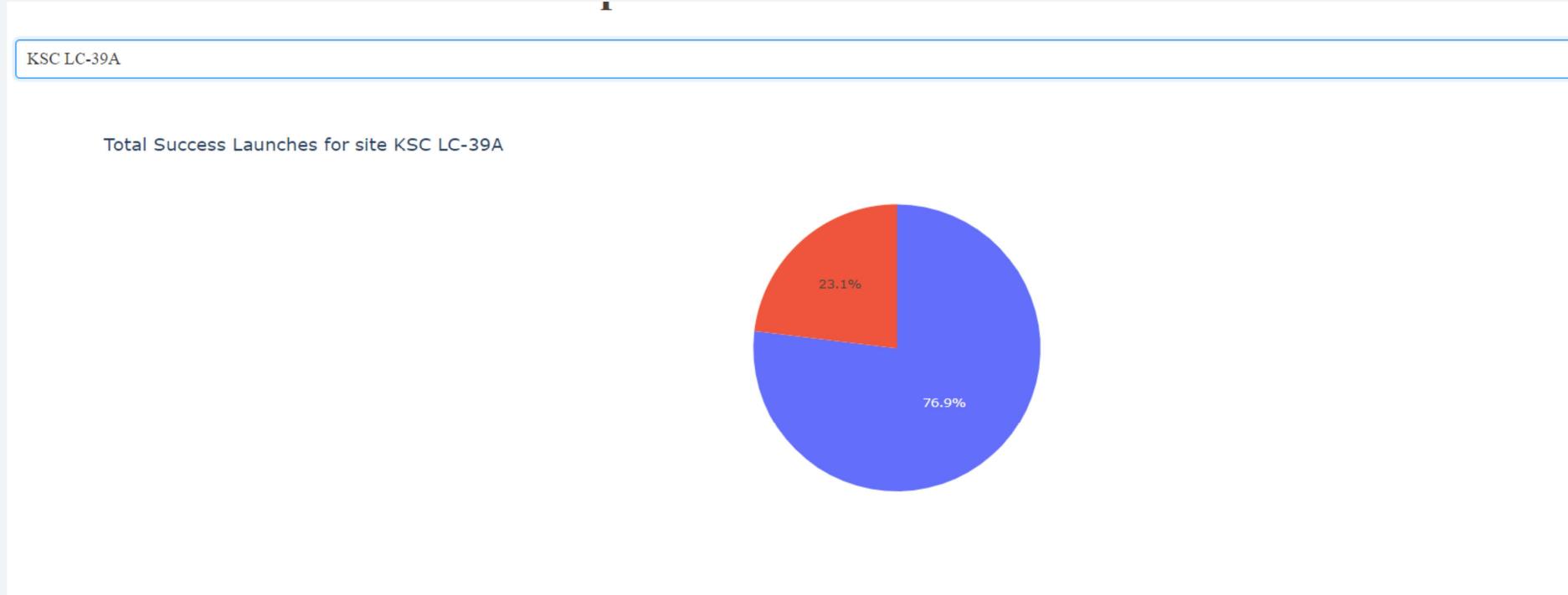
Section 4

Build a Dashboard with Plotly Dash

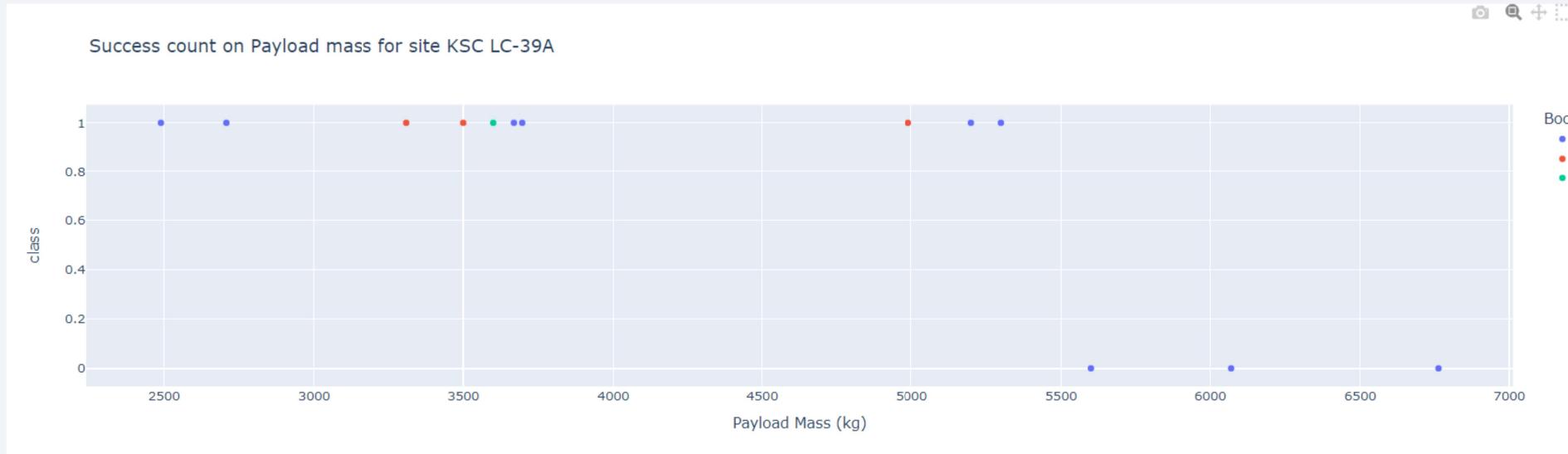
Pie chart showing the success percentage achieved by each launch site



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



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



Section 5

Predictive Analysis (Classification)

Classification Accuracy

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

TASK 12

Find the method performs best:

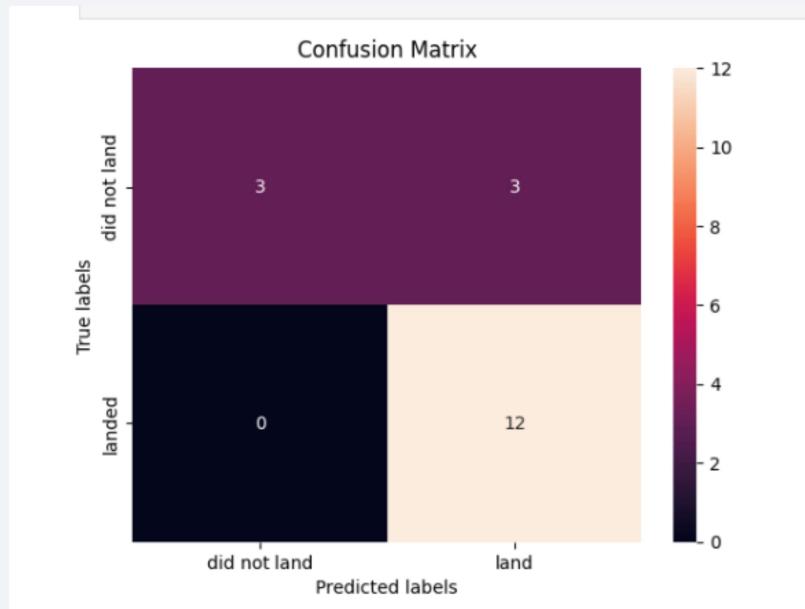
```
1 [47]: print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
print('Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
print('Accuracy for K nearest neighbors method:', 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: 0.7777777777777778
Accuracy for K nearest neighbors method: 0.8333333333333334
```

Authors

Confusion Matrix

- A confusion matrix is a table that is often used to describe the performance of a classification model on a set of data for which the true values are known. It is a fundamental tool in the field of machine learning and provides insights into the performance of a classifier, highlighting the types of errors it makes.



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.

Appendix

https://github.com/MohanBI/ds_tutorial/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

https://github.com/MohanBI/ds_tutorial/blob/main/jupyter-labs-eda-dataviz.ipynb

https://github.com/MohanBI/ds_tutorial/blob/main/jupyter-labs-eda-sql-coursera_sqlite.ipynb

https://github.com/MohanBI/ds_tutorial/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

https://github.com/MohanBI/ds_tutorial/blob/main/jupyter-labs-webscraping.ipynb

https://github.com/MohanBI/ds_tutorial/blob/main/lab_jupyter_launch_site_location.ipynb

https://github.com/MohanBI/ds_tutorial/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

Thank you!

