IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Daniel Abora

31st January 2025

Outline

Executive Summary

- Summary of methodologies

  - Data Collection through API

  - Data Collection with Web Scraping

  - Data Wrangling/ Data Cleaning

  - Exploratory Data Analysis with SQL

  - Exploratory Data Analysis with Data Visualization

  - Interactive Visual Analytics and Dashboard 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. The 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 amongst 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

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and nd 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

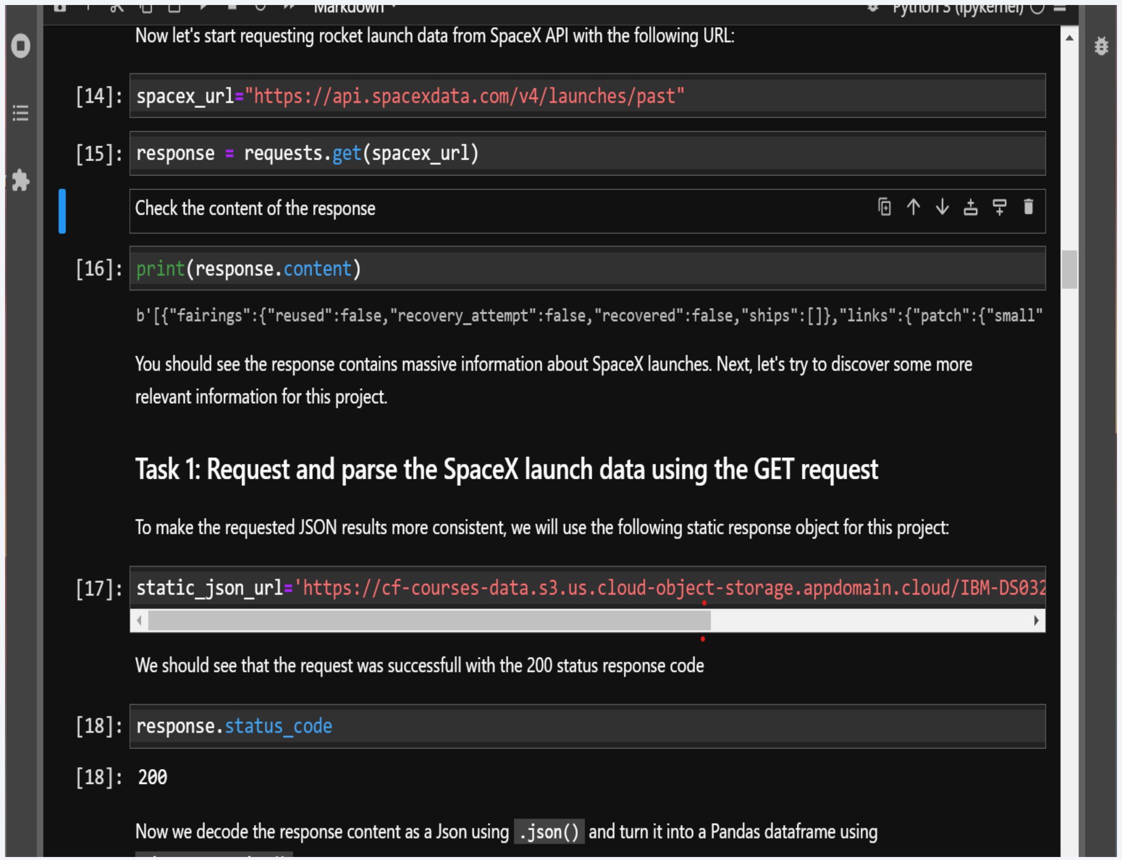
Data Collection

- Describe how data sets were collected.
  - 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 filled 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

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

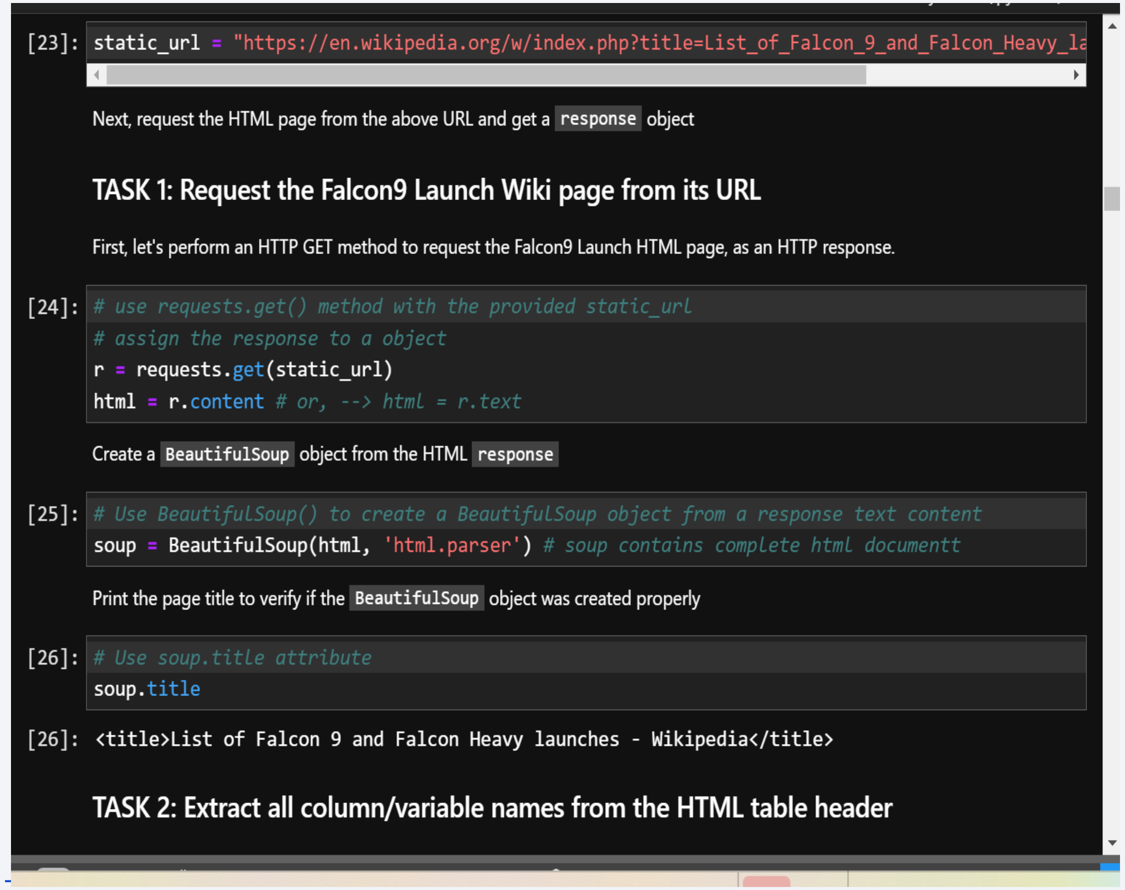
- Add the GitHub URL of the completed SpaceX API calls notebook https://github.com/Dannypist36/TechBI/blob/main/SpaceX%20Falcon%209%20first%20stage%20Landing%20Prediction%20Lab%201%20collecting%20the%20data.ipynb as an external reference and peer-review purpose



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 dataframe.Add the
- GitHub URL of the completed web scraping notebook:

https://github.com/Dannypist36/TechBI/blob/main/Lab%20Notebook%202%20-%20Data%20Collection%20with%20Web%20Scraping.ipynb

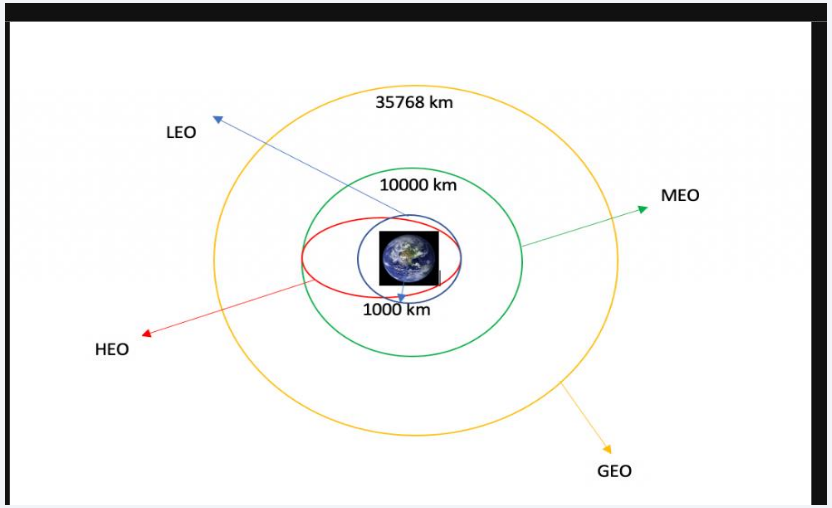


Data Wrangling

- We performed exploratory data analysisand 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

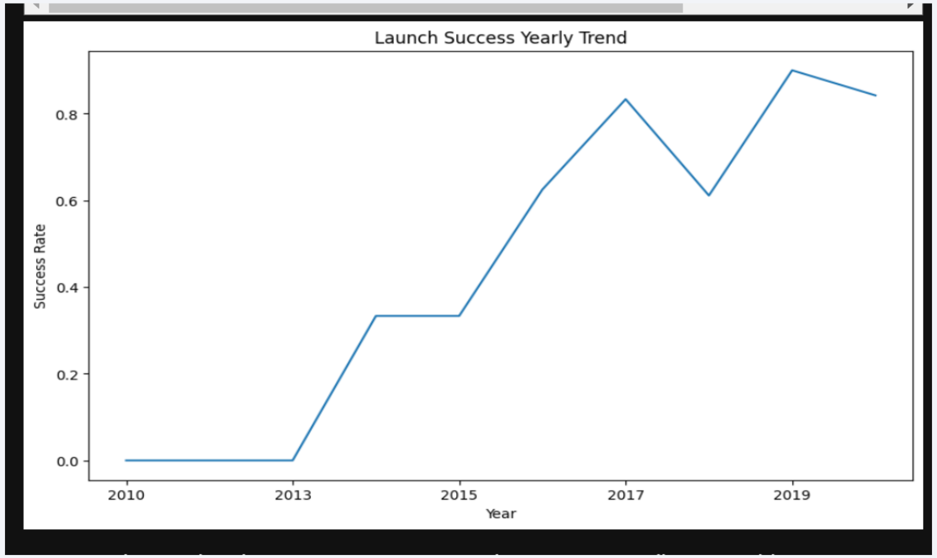
  outcome column and exported the results to csv.
- GitHub URL of the completed data wrangling related notebooks:

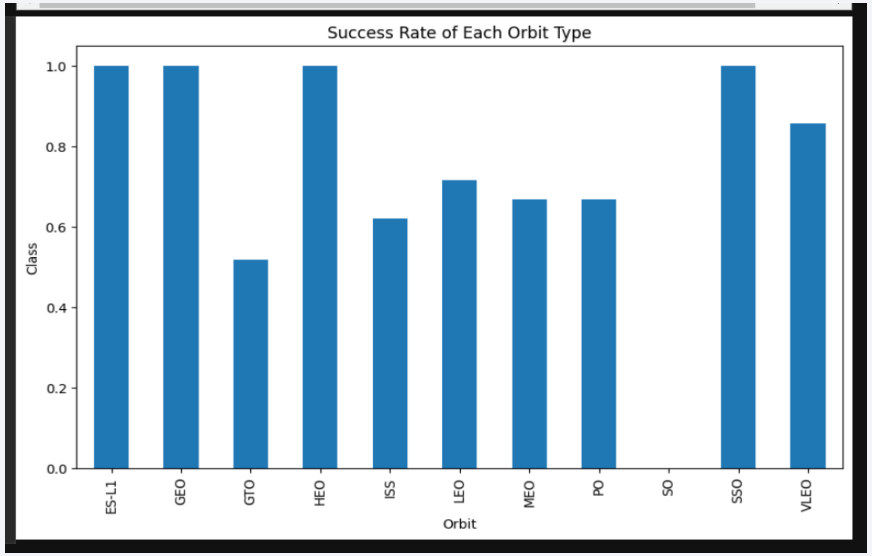
https://github.com/Dannypist36/TechBI/blob/main/labs-jupyter-spacex-Data%20wrangling-v2.ipynb



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





The link to the notebook is:
https://github.com/Dannypist36/TechBI/blob/main/EDA%20with%20Visualization%20Lab%20jupyter-labs-eda-dataviz-v2.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 names.

• The link to the notebook is:

https://github.com/Dannypist36/TechBI/blob/main/Hands-on%20Lab%20Complete%20the%20EDA%20with%20SQL.ipynb

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

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:

https://github.com/Dannypist36/TechBI/blob/main/Lab%20Notebook%207%20-%20Machine%20Learning%20Predictions.ipynb

Results

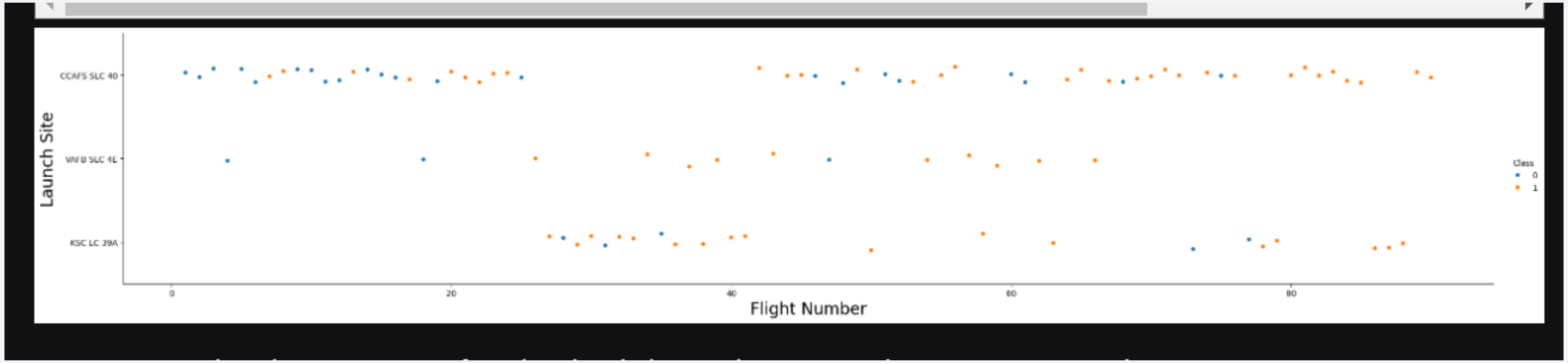
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

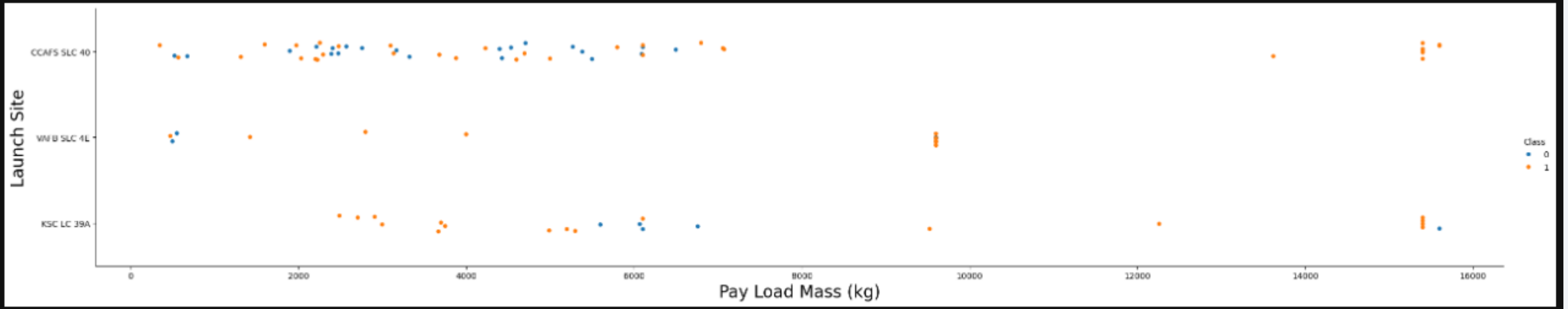
Flight Number vs. Launch Site

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



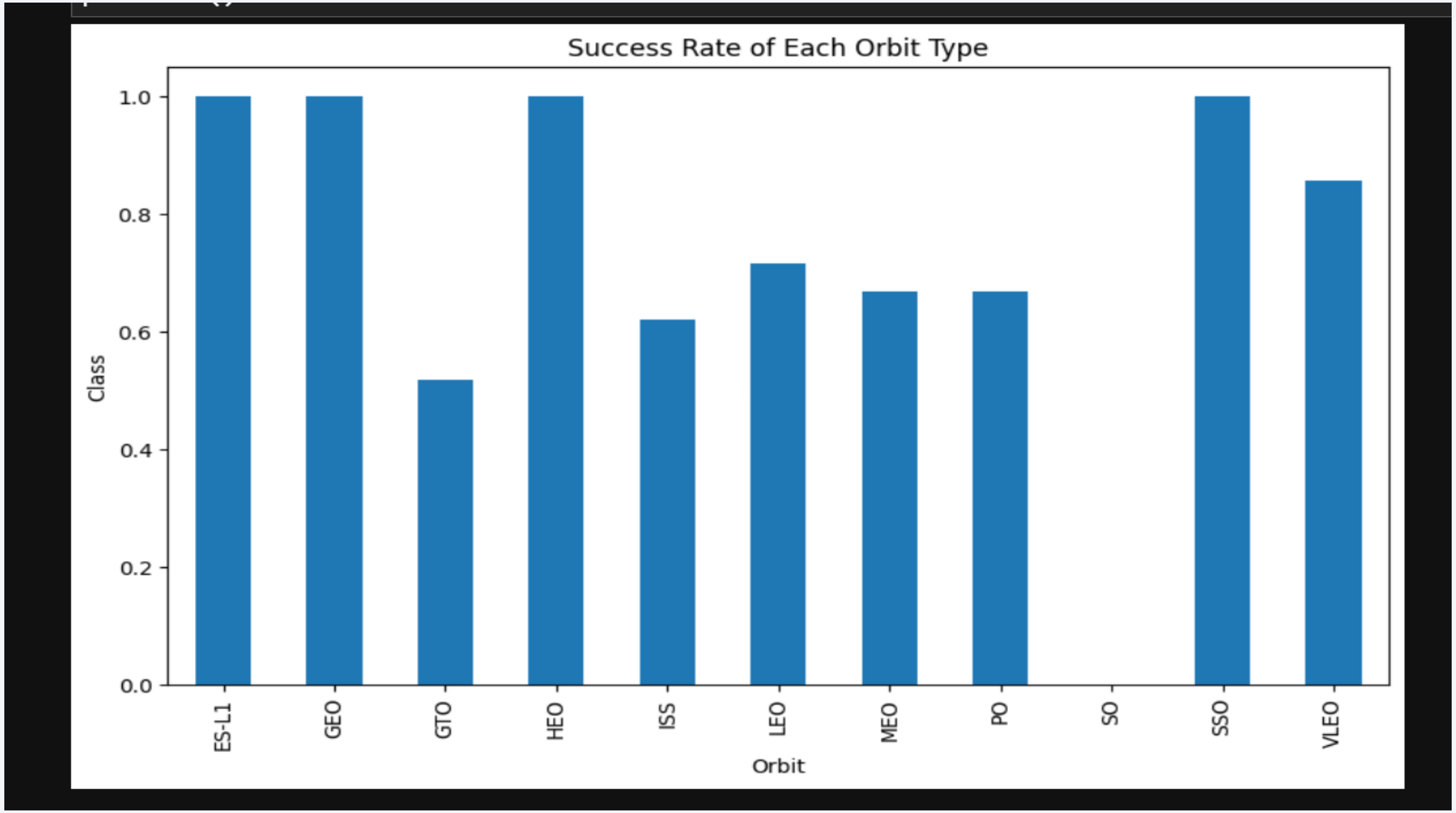
Payload vs. Launch Site

```python
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Pay Load Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



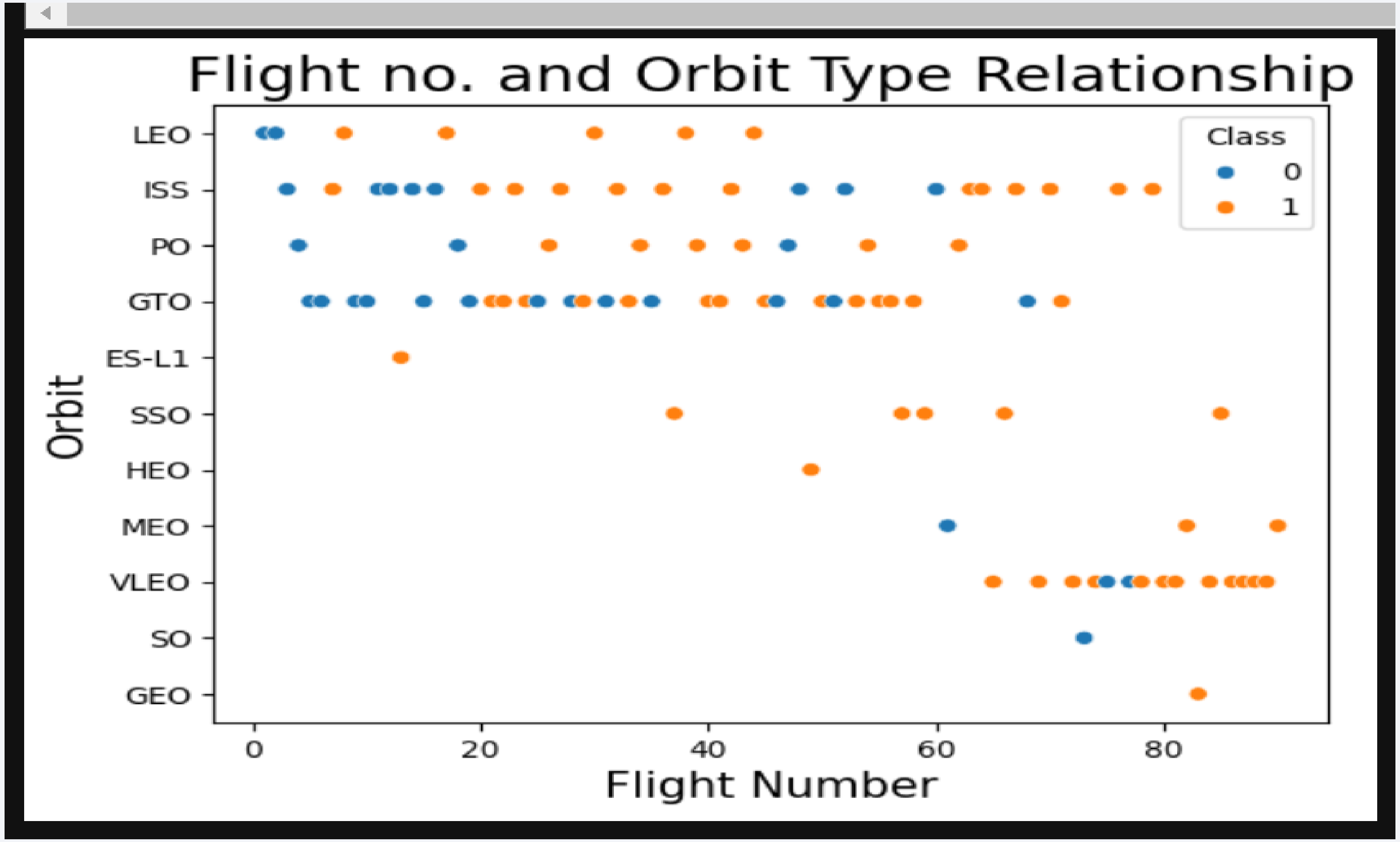
Success Rate vs. Orbit Type

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



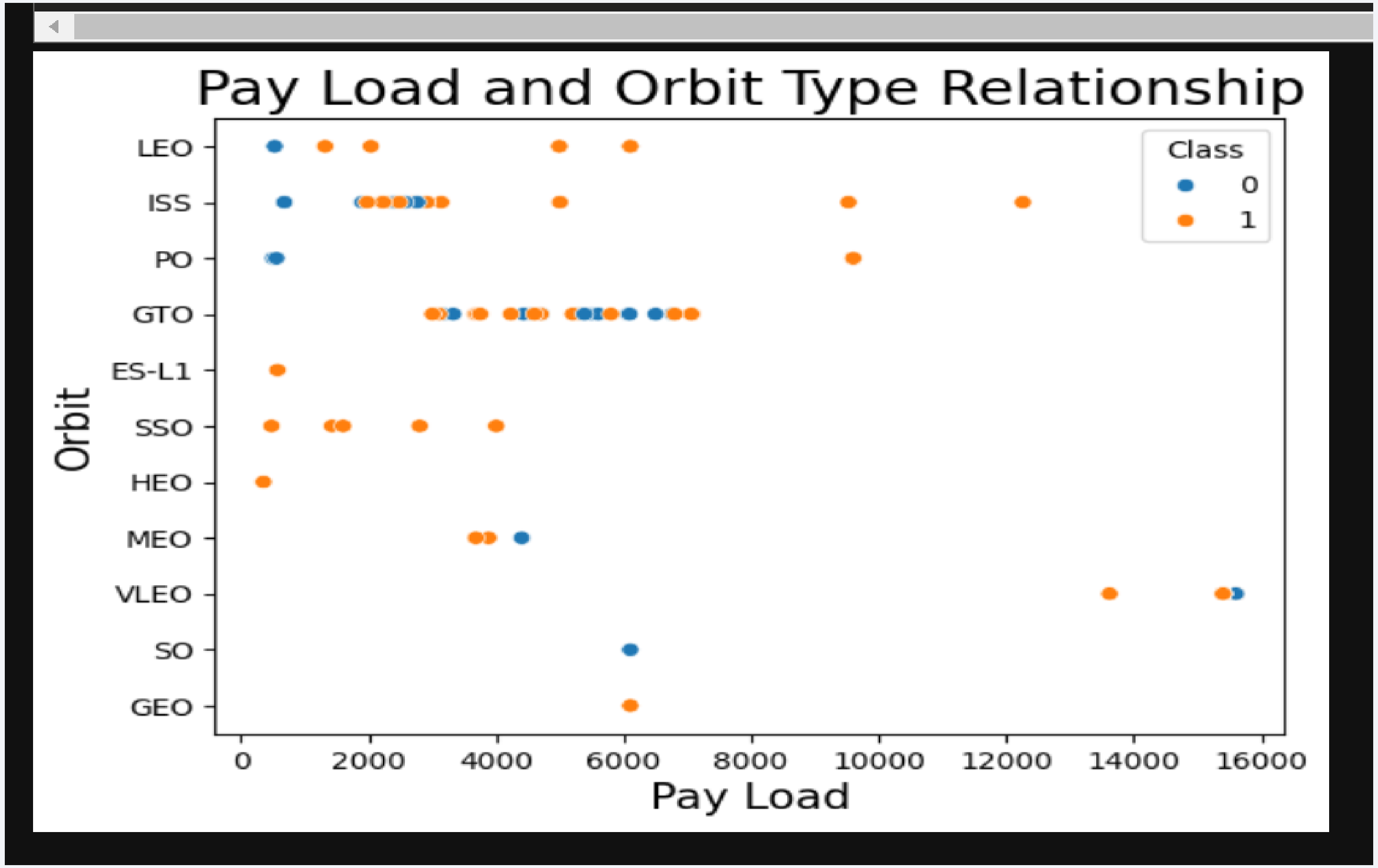
Flight Number vs. Orbit Type

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.



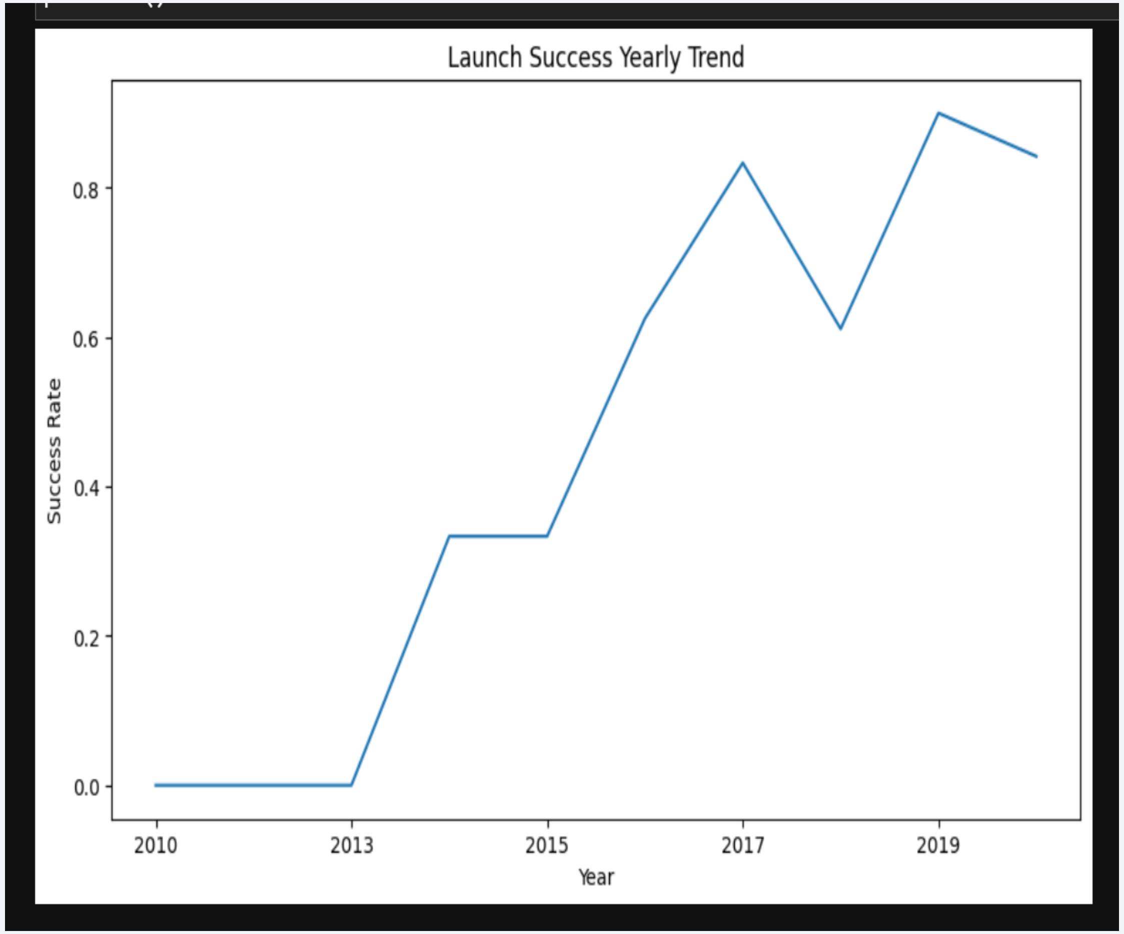
Payload vs. Orbit Type

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

```
[20]: sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1;
```

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

[20]:

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

Launch Site Names Begin with 'CCA'

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

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

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

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

[21]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Lan
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Fail
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	Fail
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	

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)

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

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

[22]:

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

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

```
[23]: %sql SELECT AVG(PAYLOAD_MASS__KG_) \
        FROM SPACEXTBL \
        WHERE BOOSTER_VERSION = 'F9 v1.1';

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

[23]:

AVG(PAYLOAD_MASS__KG_)
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

```
[36]: sql SELECT MIN(DATE) AS FIRST_SUCCESS_LANDING FROM SPACEXTBL WHERE Landing_Outcome = 'Success

 * sqlite:///my_data1.db
Done.

[36]: FIRST_SUCCESS_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

```
[34]: SPACEXTBL WHERE PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 AND Landing_Outcome = 'Success (drone

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

[34]:

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 filter for WHERE MissionOutcome was a success or a failure.

List the total number of successful and failure mission outcomes

```
[18]: %sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME
```

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

[18]:

missionoutcomes
1
98
1
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.

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

```
[19]: ersion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

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

[19]:

boosterversion
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 to filter for failed landing out comes in drone ship, their booster versions, and launch site names for year 2015

Note: SQLLite 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.

```
[20]: %sql SELECT substr(Date, 6,2),MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL wher
```

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

[20]:

substr(Date, 6,2)	Mission_Outcome	Booster_Version	Launch_Site
01	Success	F9 v1.1 B1012	CCAFS LC-40
02	Success	F9 v1.1 B1013	CCAFS LC-40
03	Success	F9 v1.1 B1014	CCAFS LC-40
04	Success	F9 v1.1 B1015	CCAFS LC-40
04	Success	F9 v1.1 B1016	CCAFS LC-40
06	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
12	Success	F9 FT B1019	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 BYclause 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 pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

[21]:
```
%sql select Landing_Outcome, count(*) as count_outcomes \
from SPACEXTABLE\
where Date between '2010-06-04' and '2017-03-20'\
group by Landing_Outcome\
order by count_outcomes Desc;
```

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

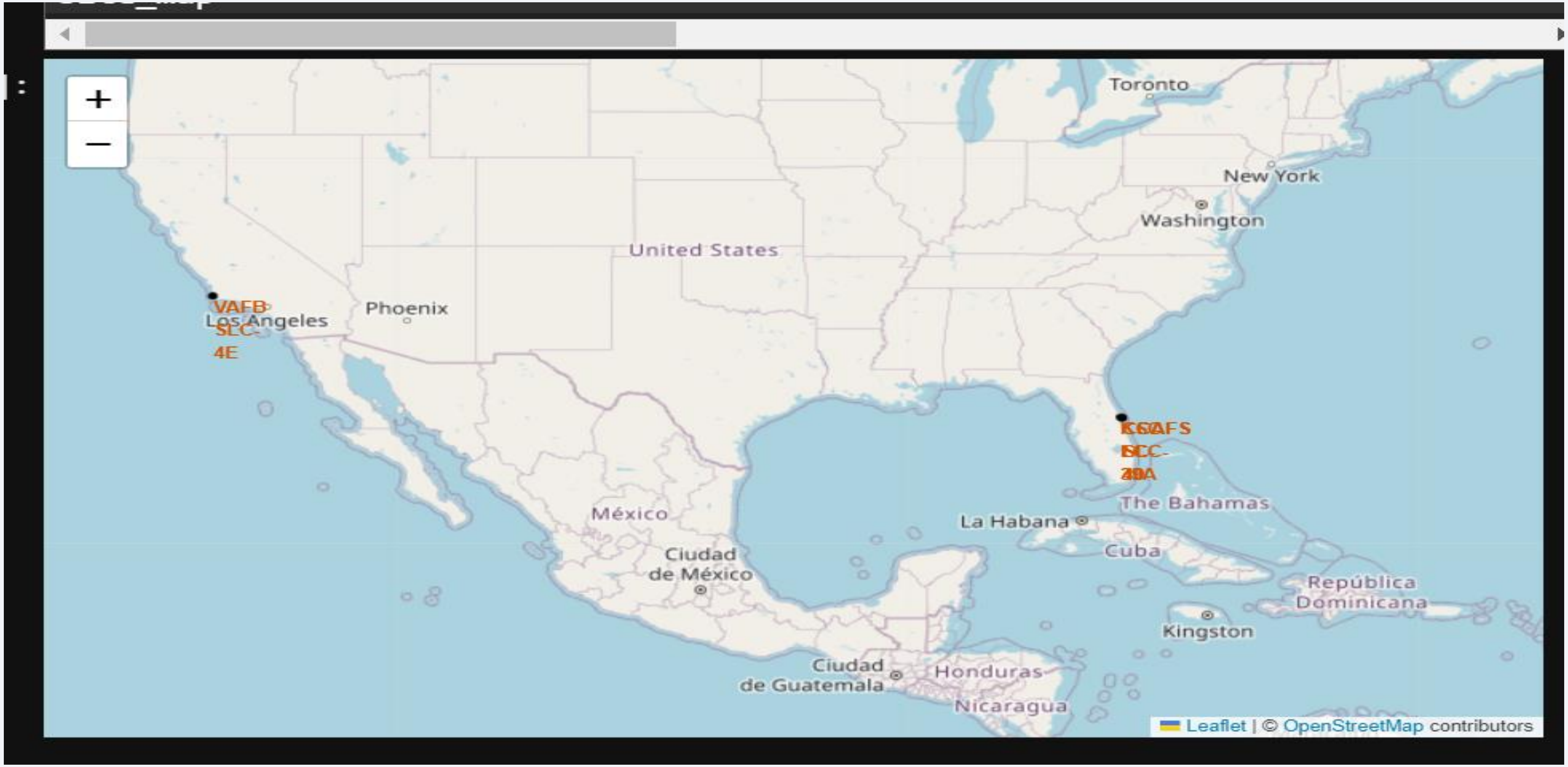
[21]:

Landing_Outcome	count_outcomes
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

Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>



<Folium Map Screenshot 2>

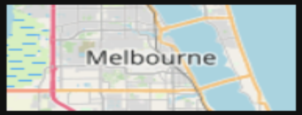
<Folium Map Screenshot 3>

```
[28]: # Create a marker with distance to a closest city, railway, highway relative
      # Draw a line between the marker to the launch site
      closest_highway = 28.56335, -80.57085
      closest_railroad = 28.57206, -80.58525
      closest_city = 28.10473, -80.64531

[29]: distance_highway = calculate_distance(launch_site_lat, launch_site_lon, close
      print('distance_highway =',distance_highway, ' km')
      distance_railroad = calculate_distance(launch_site_lat, launch_site_lon, clos
      print('distance_railroad =',distance_railroad, ' km')
      distance_city = calculate_distance(launch_site_lat, launch_site_lon, closest_
      print('distance_city =',distance_city, ' km')

      distance_highway = 0.5834695366934144  km
      distance_railroad = 1.2845344718142522  km
      distance_city = 51.43416999517233  km
```

A railway map symbol may look like this:

A highway map symbol may look like this:

A city map symbol may look like this:

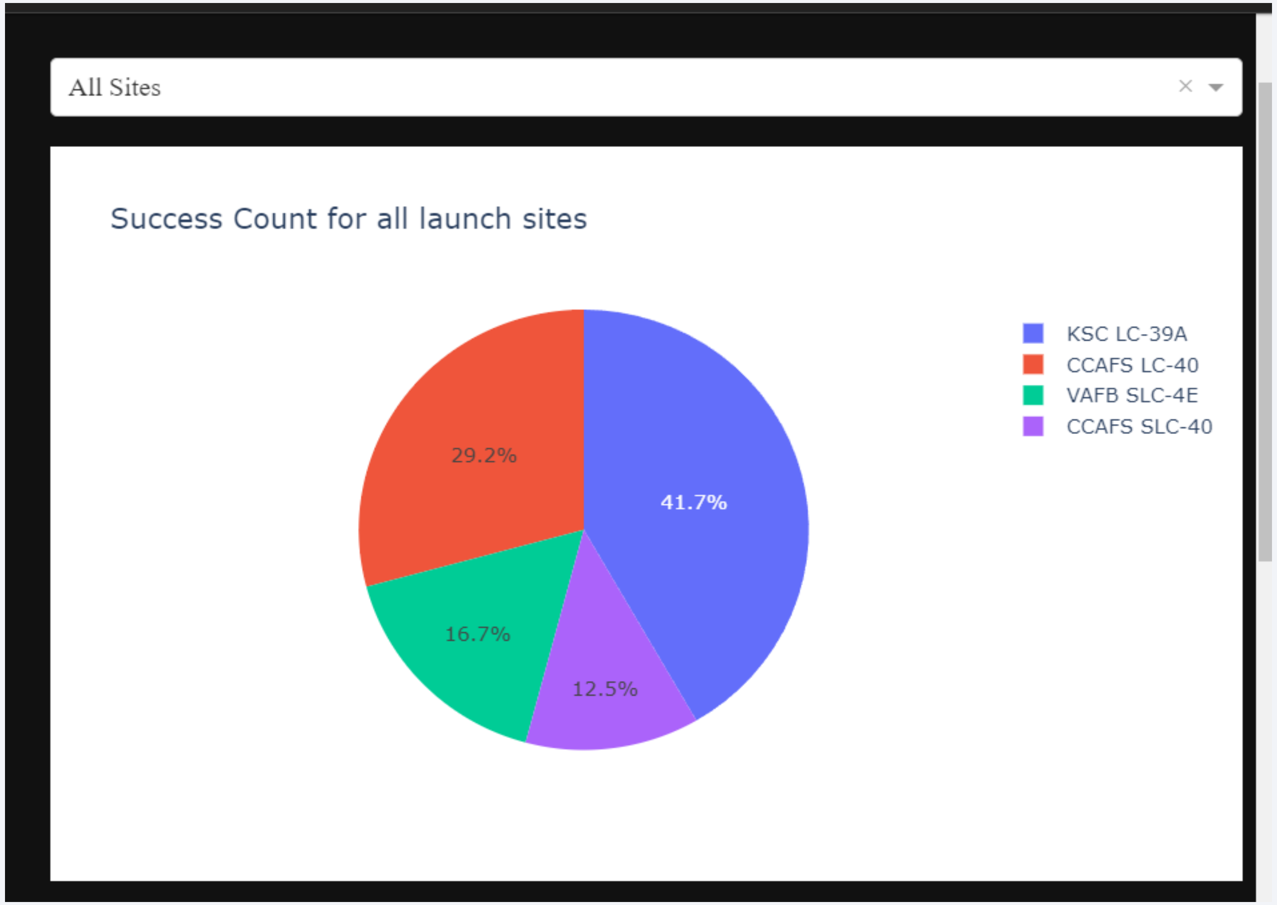


Section 4

Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

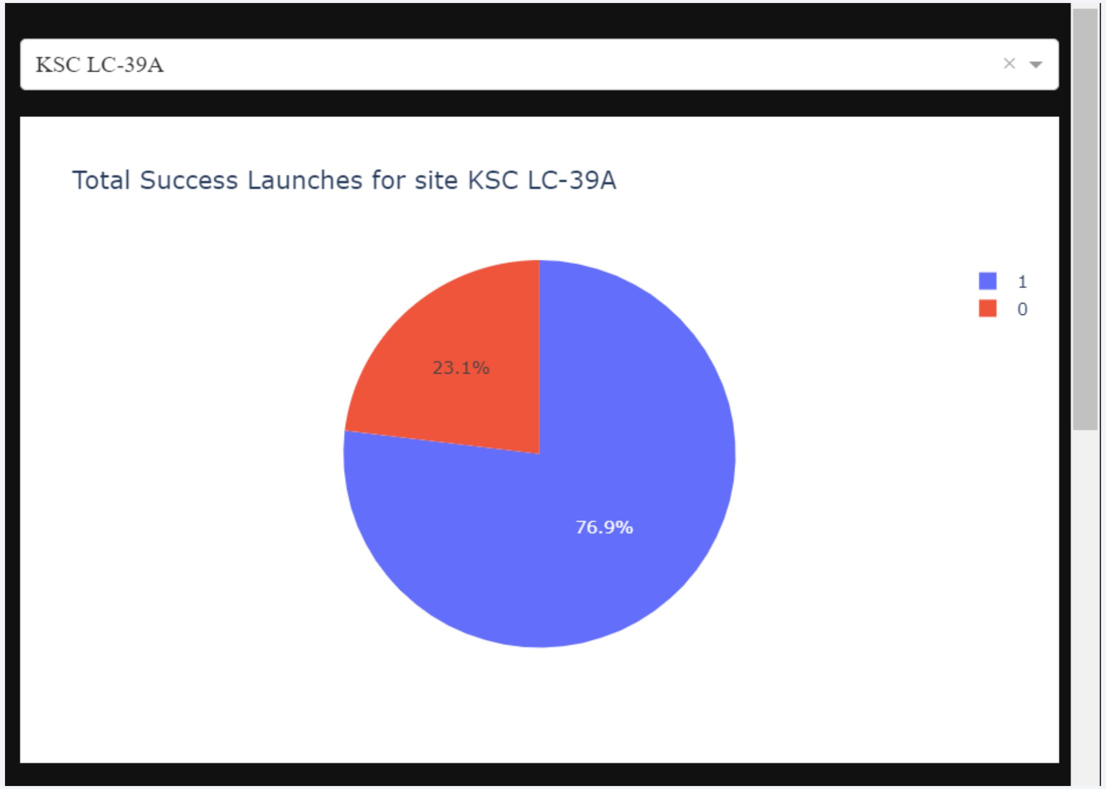
We can see that KSCLC-39A had the most successful launches from all the sites



<Dashboard Screenshot 2>

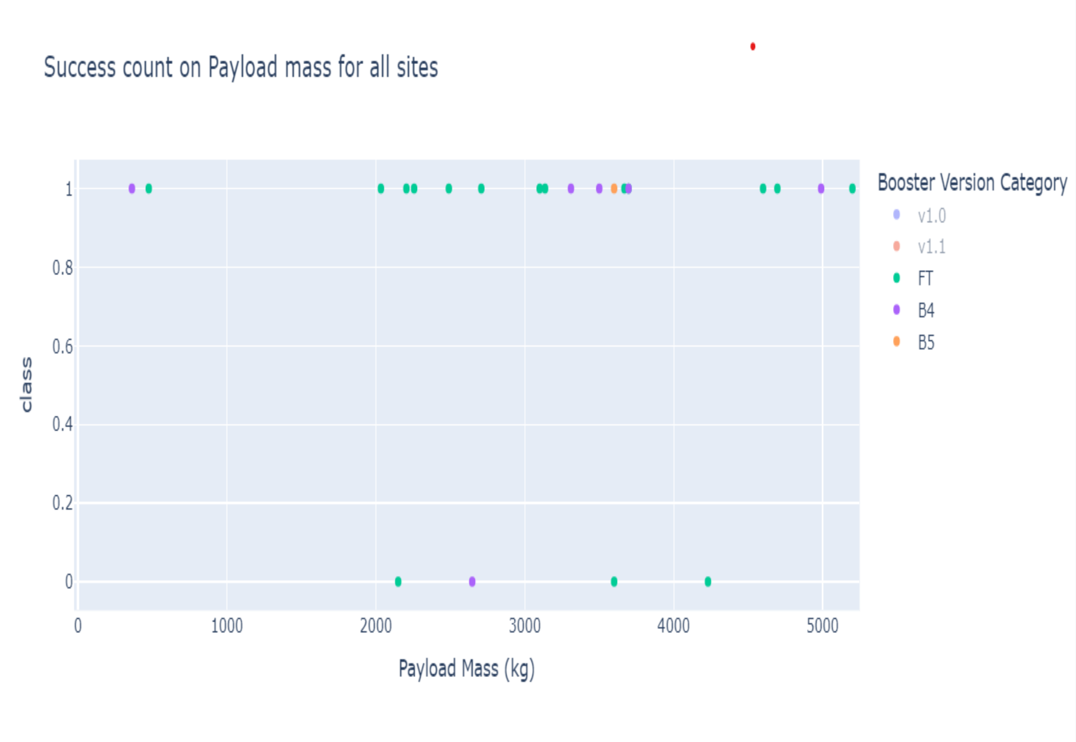
Pie chart showing the launch site with the highest launch success ratio.

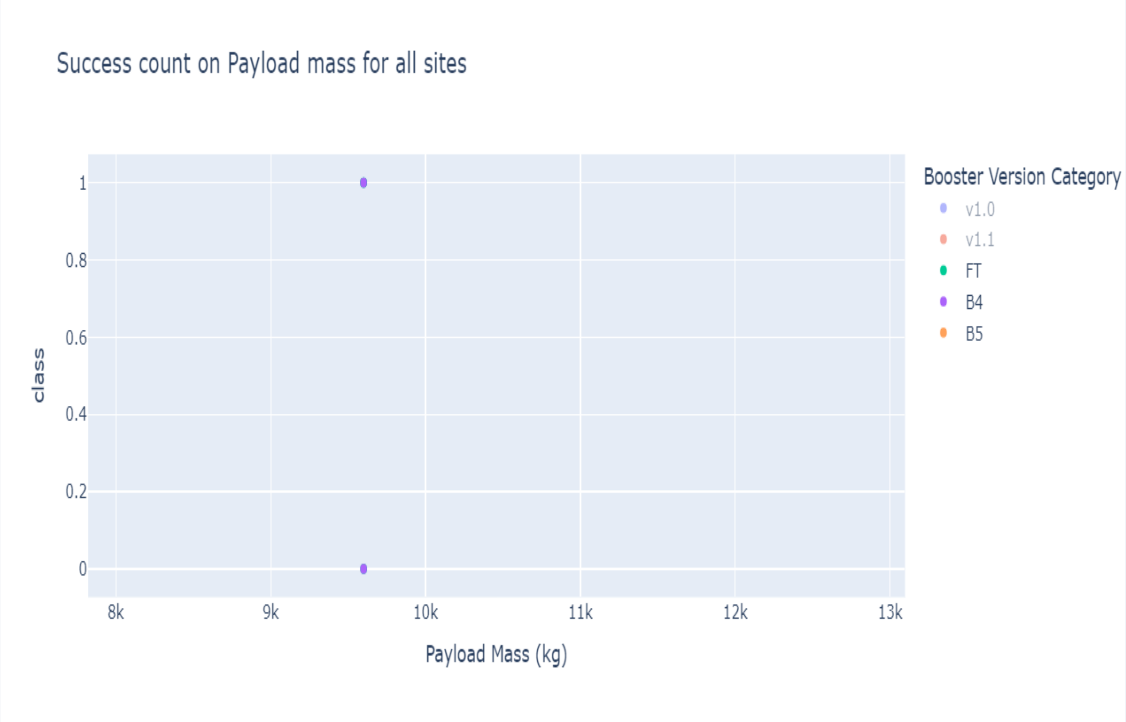
KSC LC-39A achieved a 74.9% success while getting a 23.1% failure rate



<Dashboard Screenshot 3>

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

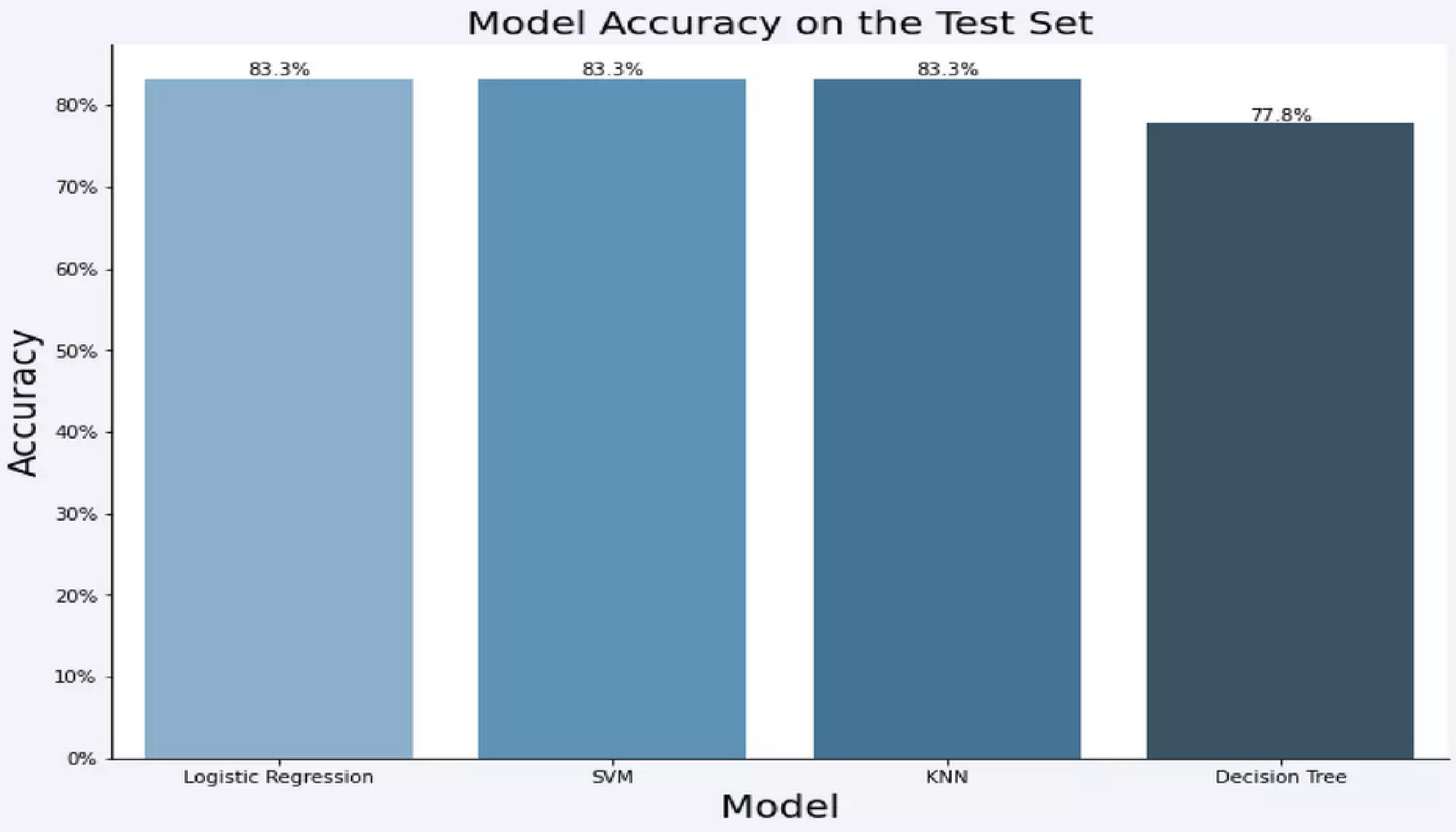




Section 5

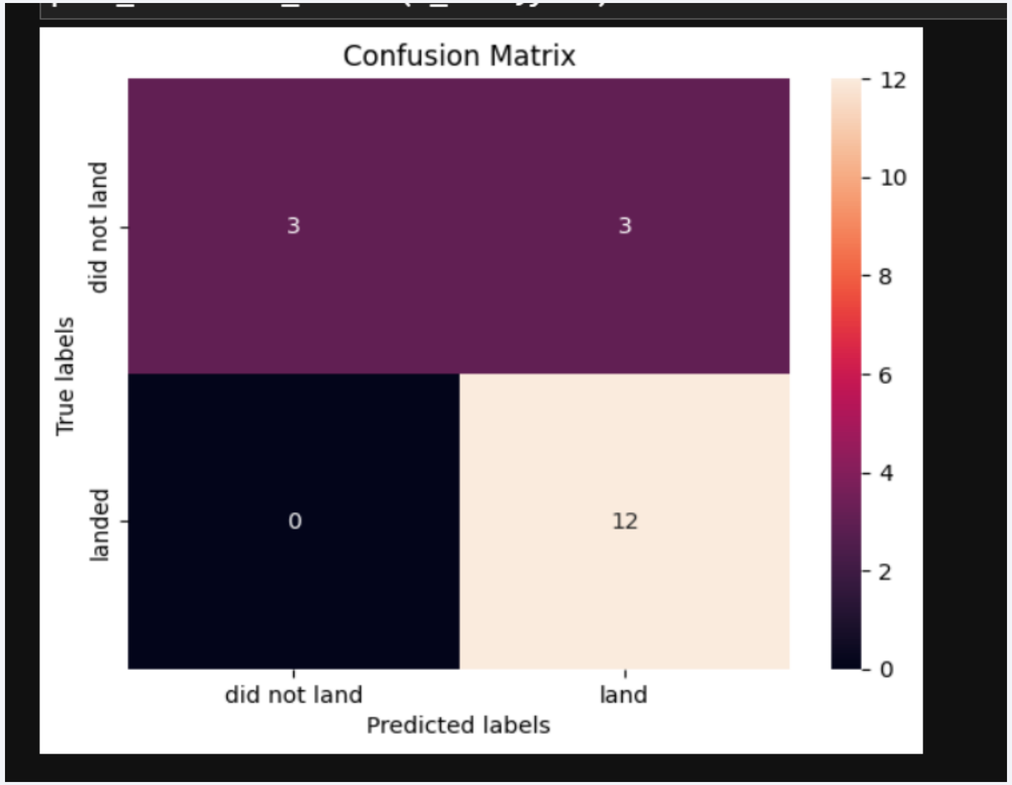
Predictive Analysis (Classification)

Classification Accuracy



Confusion Matrix

The confusion matrix for the decision tree classifier shows that the 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

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!