IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Umar Odulaja
Feb 2024

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- Project background and context

  SpaceX advertises Falcon 9 rocket launches on its website, with 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 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. 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 determines if the rocket will land successfully
  - The interaction amongst various features that determine the success rate of a successful lading
  - 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
  - Find some patterns in the data and determine what would be the label for training supervised models
- 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

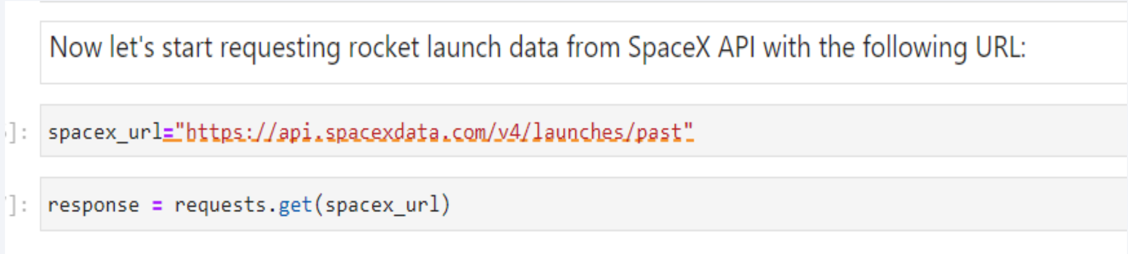
Data Collection

- Describe how data sets were collected.
  - Data collection was done using get request to the SpaceX API
  - Next, we decoded the response content as 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 the missing values with the mean where necessary
  - In addition, we performed web scraping from Wikipedia 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:

https://github.com/Abiolar11/capstone/blob/main/Data%20collection.ipynb



Now let's start requesting rocket launch data from SpaceX API with the following URL:

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

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

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

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

```
response.status_code
```

```
200
```

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

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

Data Collection - Scraping

- We applied web scraping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it to a pandas dataframe
- Here's the link to the notebook: https://github.com/Abiolar11/capstone/blob/main/Data%20Web%20Scraping.ipynb

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

```
7]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response, 'html.parser')
```

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

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

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

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_fro`
column name one by one

```
16]: column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_n
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
        name = extract_column_from_header(temp[x])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

Check the extracted column names

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 orbit
- We created landing outcome label from outcome column and ecported the results in csv
- Here’s the link to the notebook:

https://github.com/Abiolar11/capstone/blob/main/Data%20Wrangling%20-%20Lab%202.ipynb

EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch site, payload and launh site, success rate of each orbit type, flight number and orbit type, and the launch success yearly trend
- Here's the link to the notebook:

https://github.com/Abiolar11/capstone/blob/main/Data%20Visualization.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
- Here's the link to the notebook:

https://github.com/Abiolar11/capstone/blob/main/EDA%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 of failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to calls 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
- Here's the link to the notebook
  - https://github.com/Abiolar11/capstone/blob/main/Visual%20Analytics%20Folium.ipynb

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 site
- We plotted scatter grap showing the relationship with outcome and payload mass(kg) for the different booster version.
- Here's the link to the notebook:
  - https://github.com/Abiolar11/capstone/blob/main/spacex_dash_app.py

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 tuned 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
- Here's the link to the notebook
    - https://github.com/Abiolar11/capstone/blob/main/Predictive%20Analysis.ipynb

Results

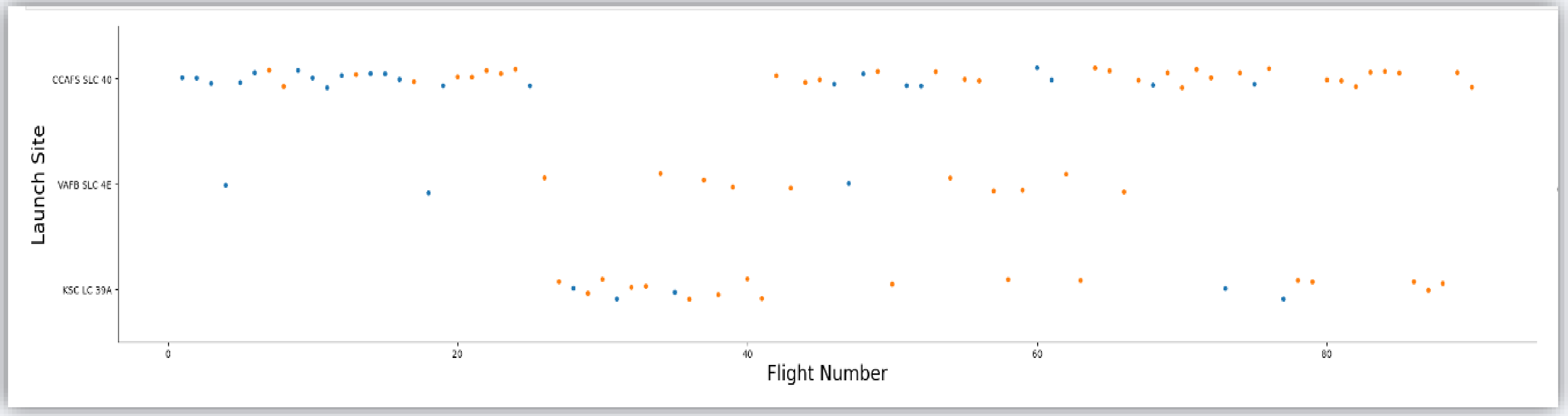
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

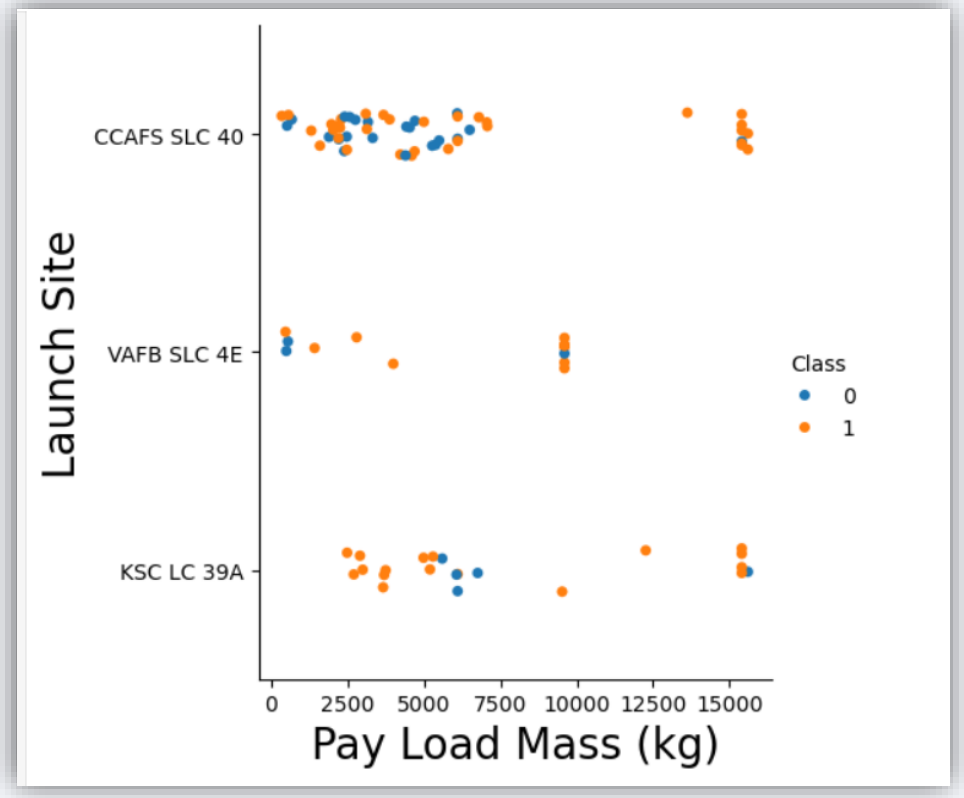
Flight Number vs. Launch Site

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



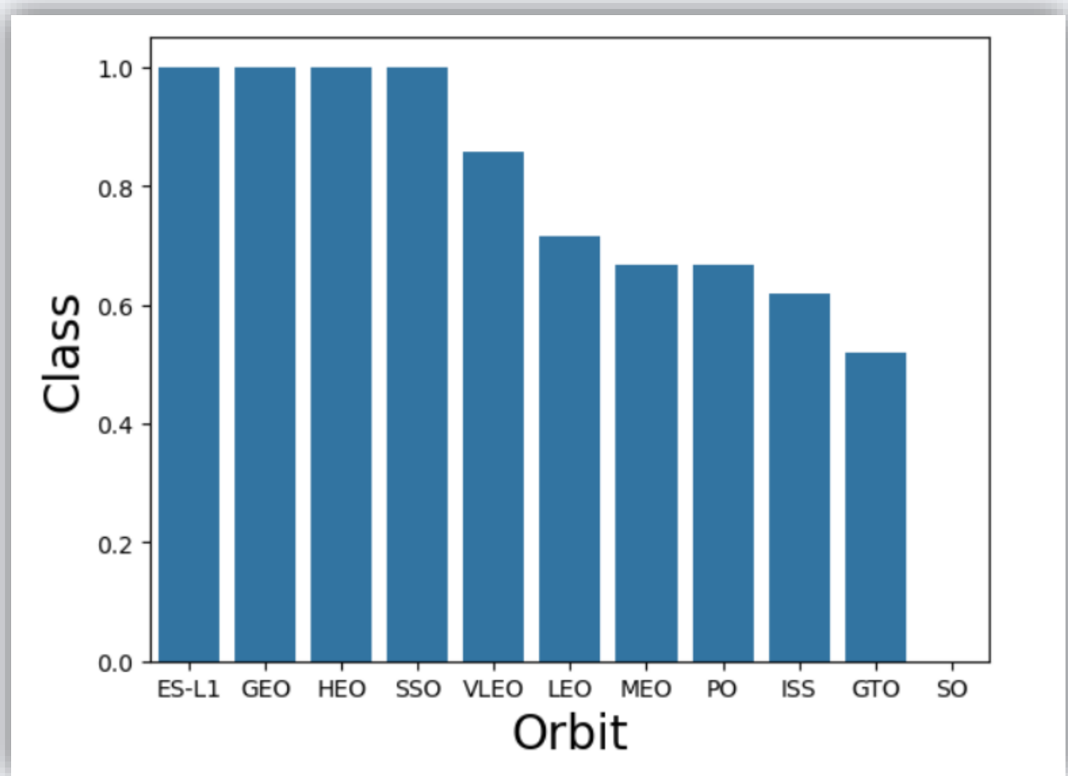
Payload vs. Launch Site

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



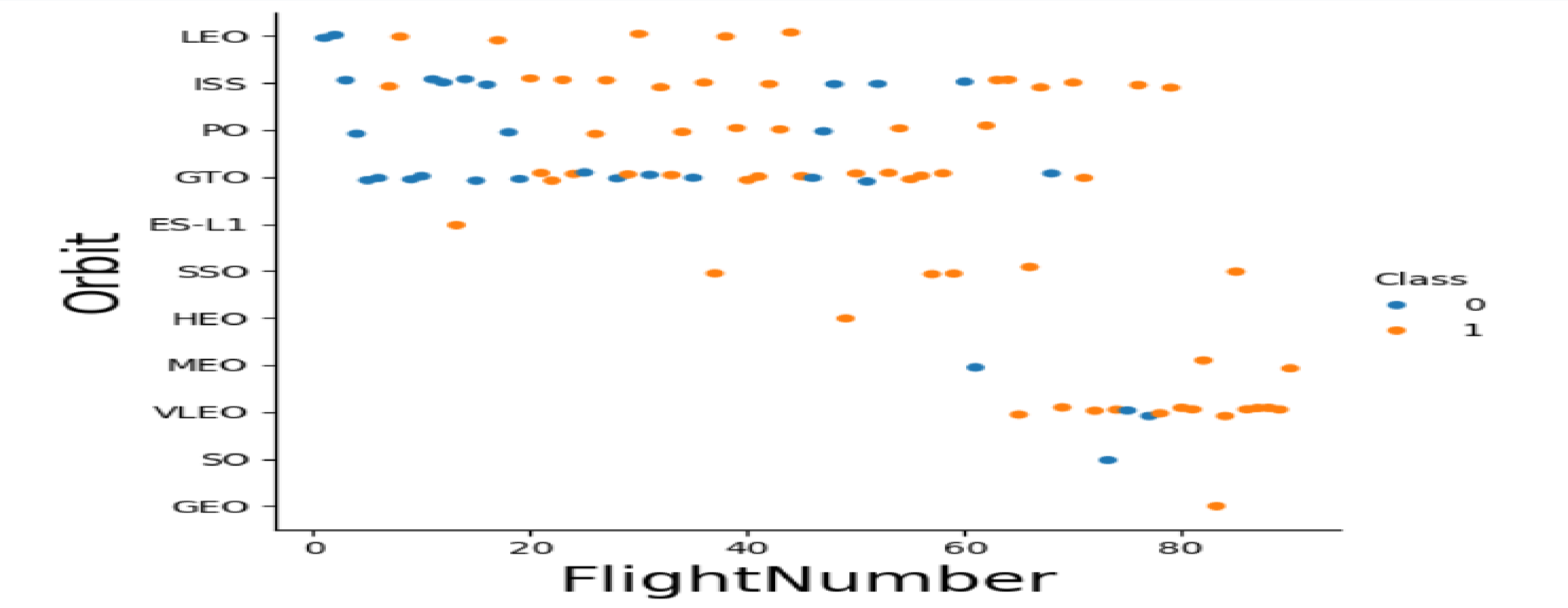
Success Rate vs. Orbit Type

- From the pilot, we can see the ES-L1, GEO, HEO,SSO,VLSO had the most success rate



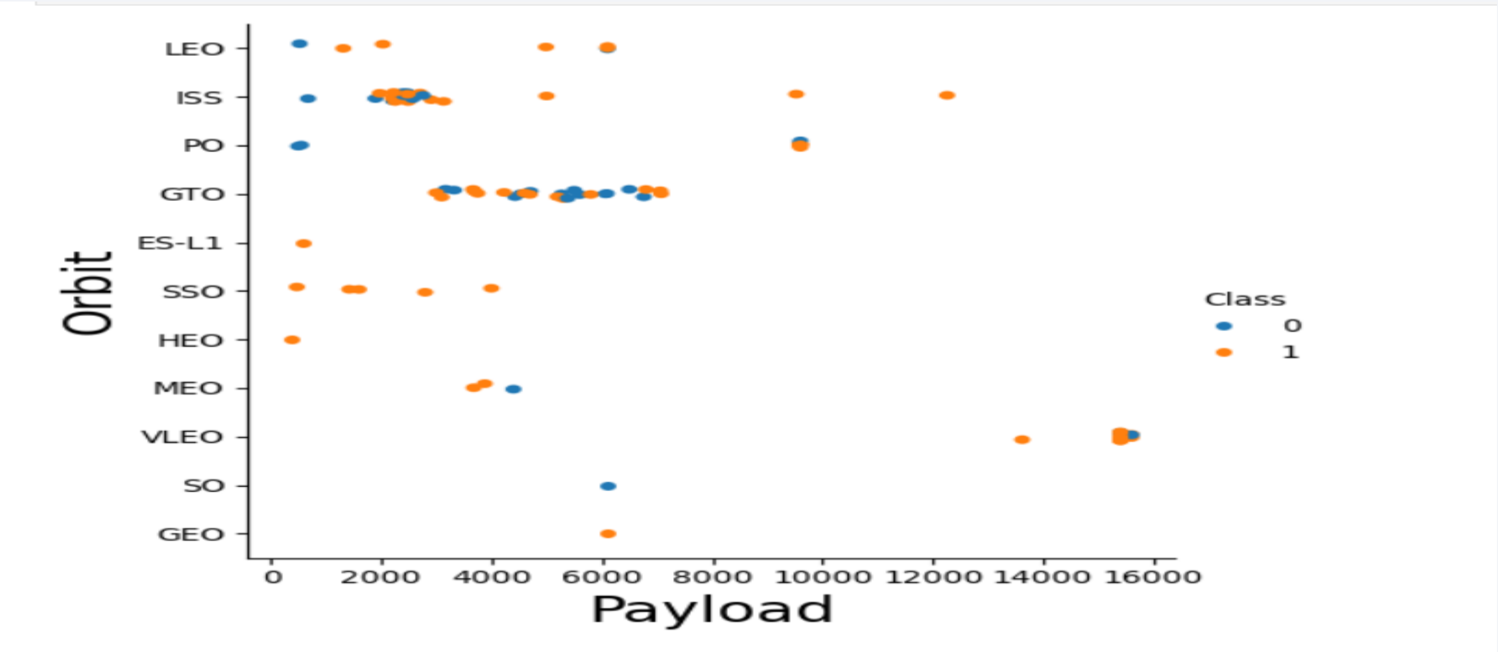
Flight Number vs. Orbit Type

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



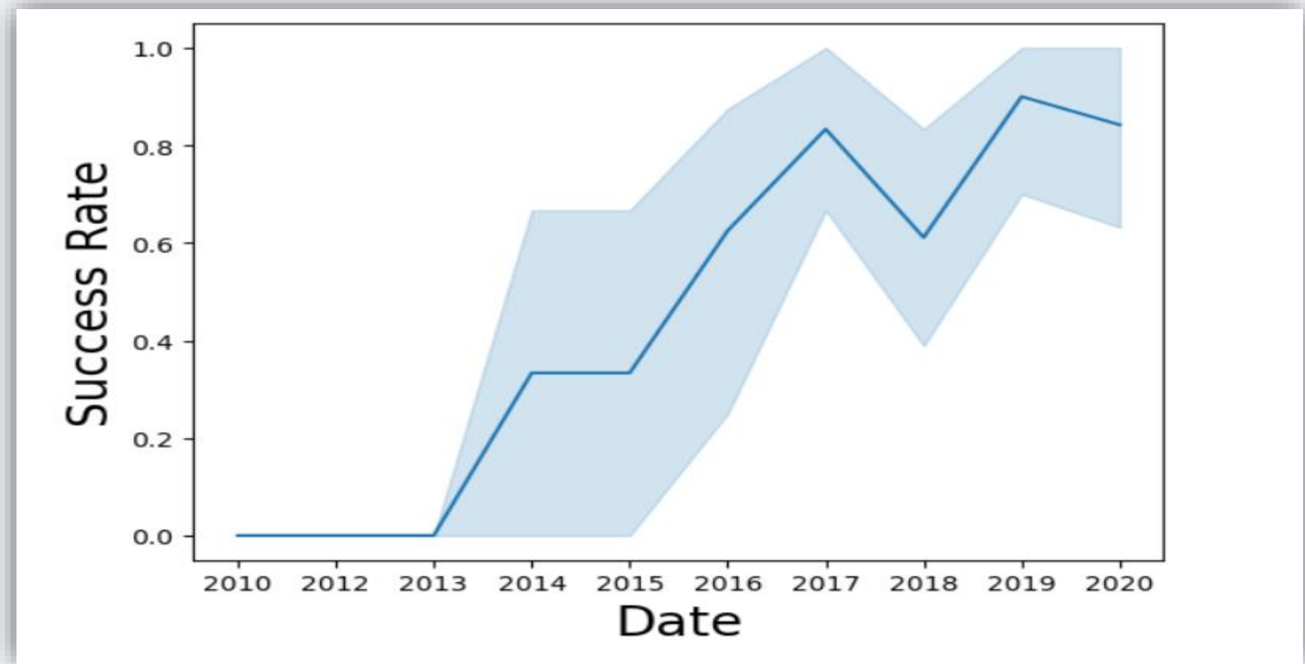
Payload vs. Orbit Type

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

[9]:

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

Launch Site Names Begin with 'CCA'

- Used the below query to display records where launch sites 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

Total Payload Mass

- The total payload mass carried by boosters launched by NASA was 45596

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

- The average payload mass carried by booster version F9 v1.1 was 2928.4

Task 4 ¶

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

[12]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

[12]:

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- The date of the first successful landing outcome on ground pad

Task 5

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

Hint:Use min function

```
14]: %sql SELECT MIN(DATE) \
FROM SPACEXTBL \
WHERE LANDING_OUTCOME = 'Success (ground pad)'
```

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

14]:

MIN(DATE)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The names of boosters which have successfully landed on drone ship and had 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

[16]:
```
%sql SELECT PAYLOAD \
FROM SPACEXTBL \
WHERE LANDING_OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

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

[16]:

Payload
JCSAT-14
JCSAT-16
SES-10
SES-11 / EchoStar 105

Total Number of Successful and Failure Mission Outcomes

- The total number of successful and failure mission outcomes

Task 7

List the total number of successful and failure mission outcomes

[17]:
```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number \
FROM SPACEXTBL \
GROUP BY MISSION_OUTCOME;
```

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

[17]:

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

Boosters Carried Maximum Payload

- The names of the booster which have carried the maximum payload mass

Task 8

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

```
[18]: %sql SELECT BOOSTER_VERSION \
FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

[18]:

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

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in 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, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

[24]:
```
%sql SELECT substr(Date,6,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, LANDING_OUTCOME \
FROM SPACEXTBL \
where [LANDING_OUTCOME] = 'Failure (drone ship)' and substr(Date,0,5)='2015';
```

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

[24]:

month	Date	Booster_Version	Launch_Site	Landing_Outcome
01	2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- 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

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.

```
[27]: %sql SELECT LANDING_OUTCOME, count(*) as count_outcomes \
FROM SPACEXTBL \
WHERE DATE between '2010-06-04' and '2017-03-20' group by LANDING_OUTCOME order by count_outcomes DESC;
```

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

[27]:

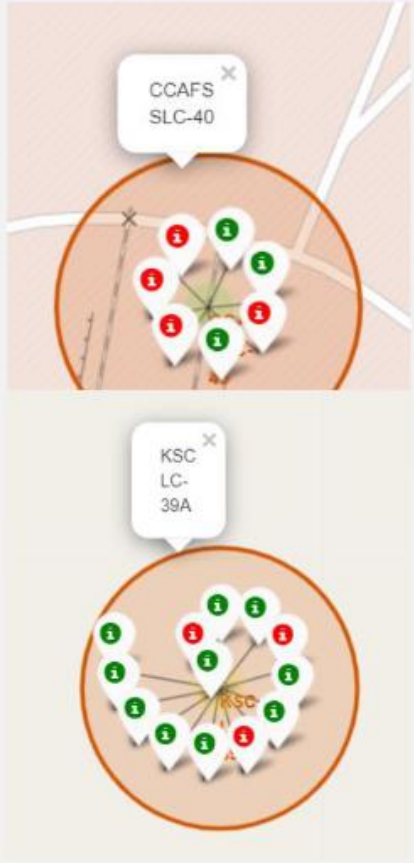
Landing_Outcome	count_outcomes
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

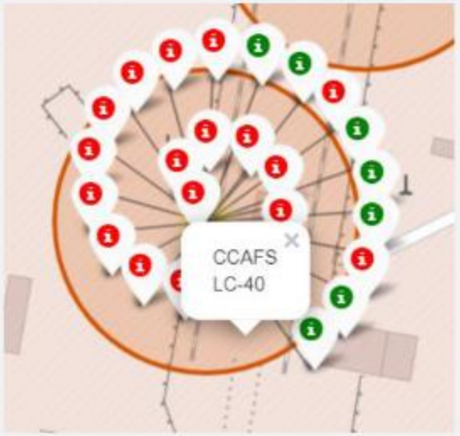
Section 3

Launch Sites Proximities Analysis

All launch site global map markers

Markers showing launch sites with color labels





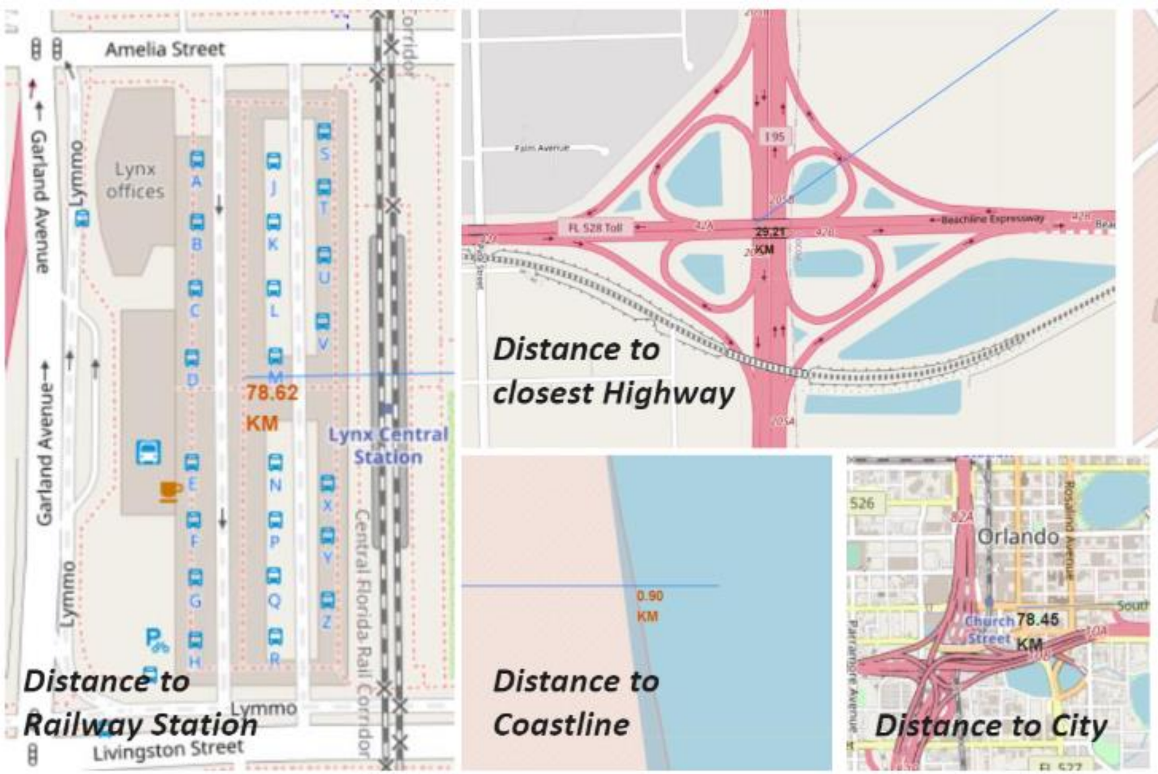
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

Launch Site distance to landmarks

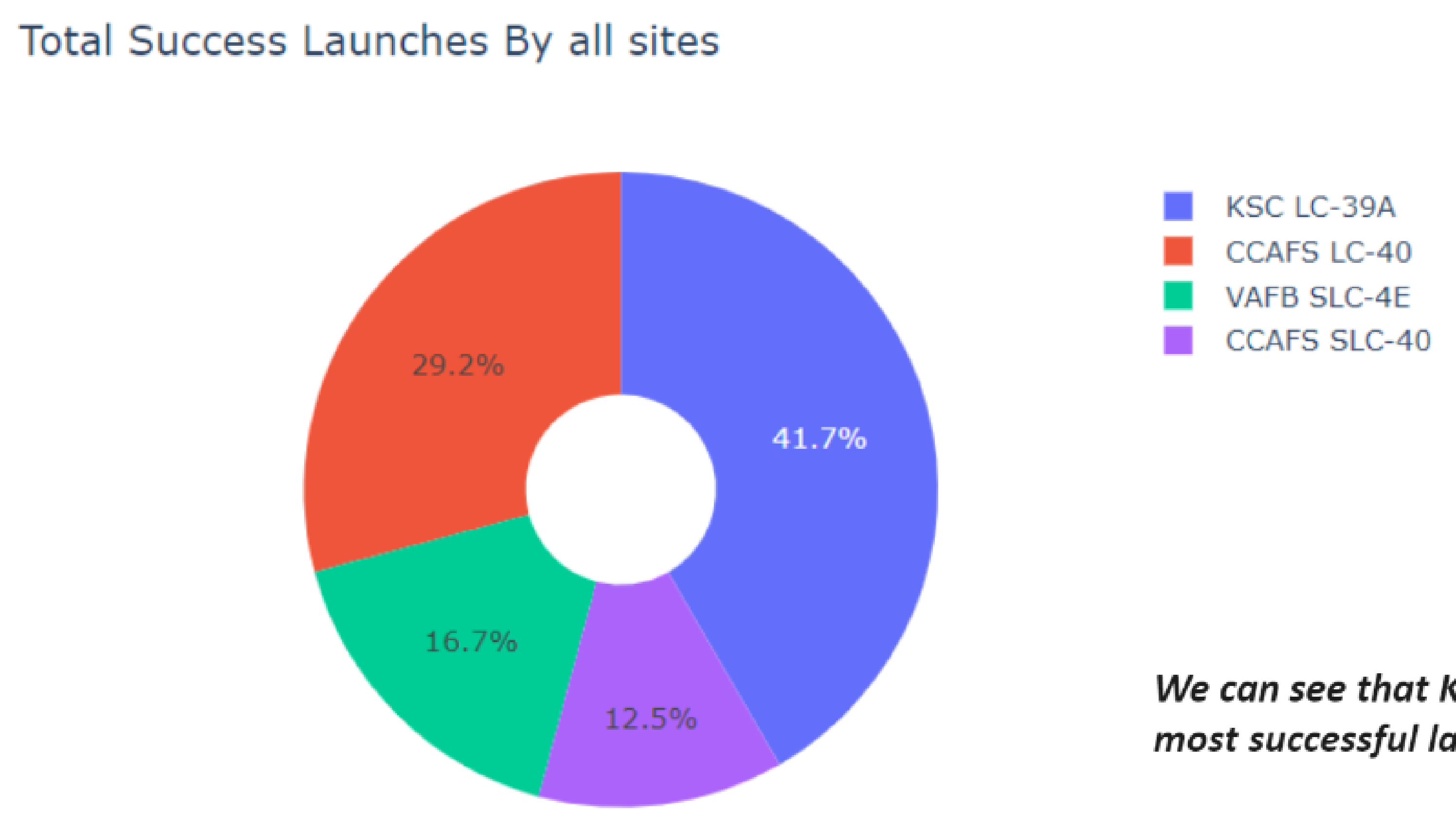


•Are launch sites in close proximity to railways? No
•Are launch sites in close proximity to highways? No
•Are launch sites in close proximity to coastline? Yes
•Do launch sites keep certain distance away from cities? Yes

Section 4

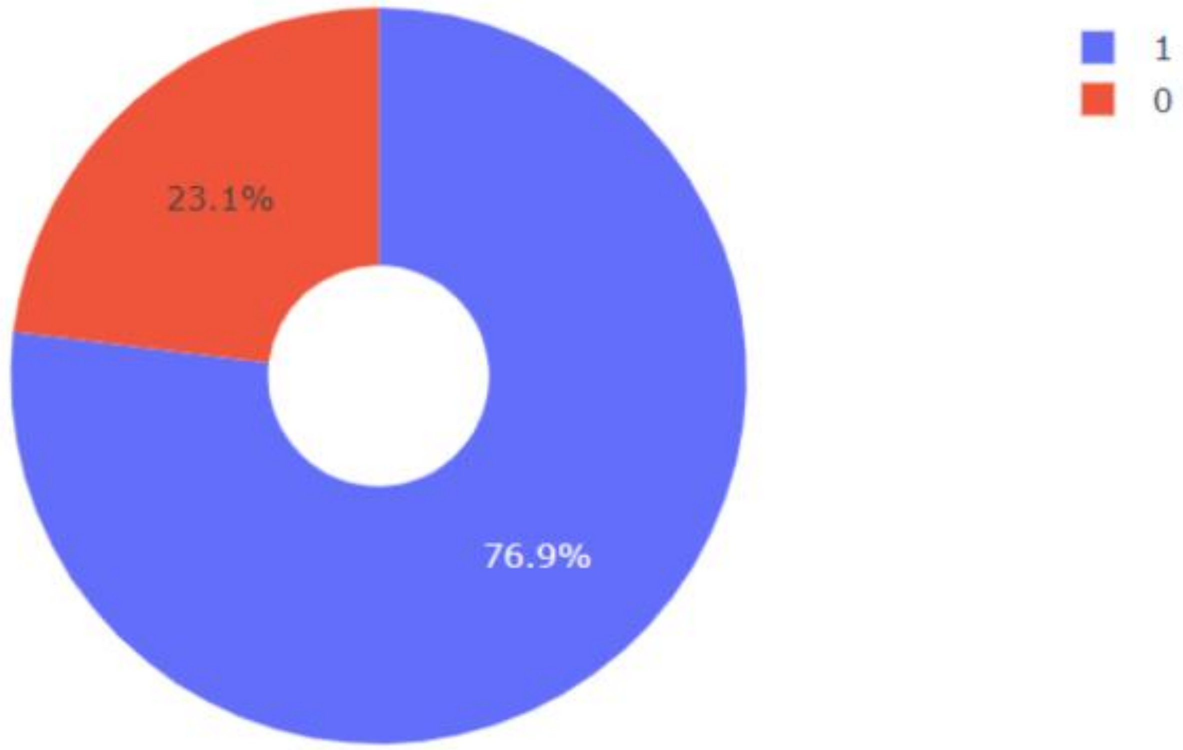
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



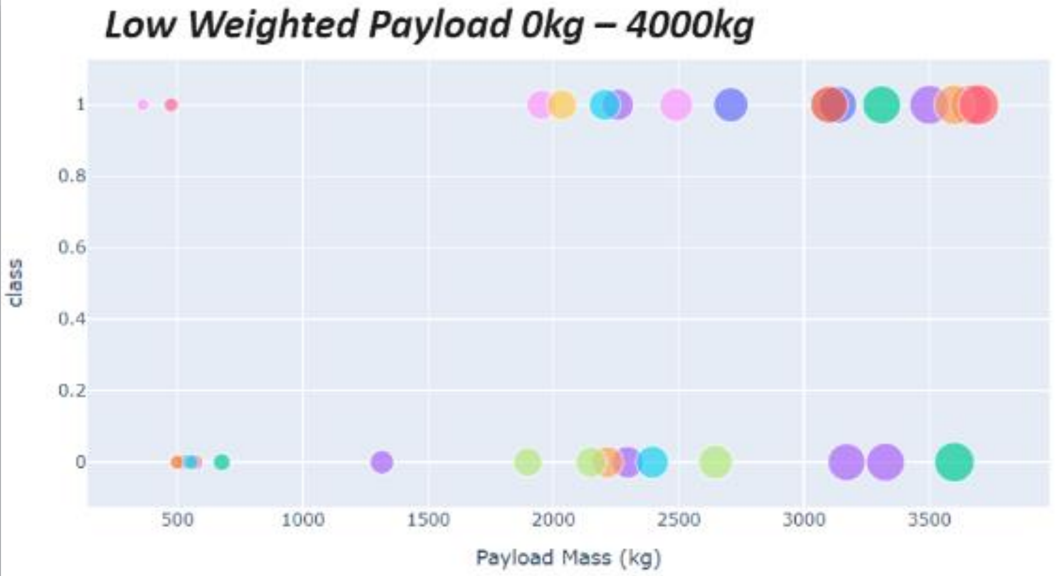
We can see that KSC LC-39A had the most successful launches from all the sites

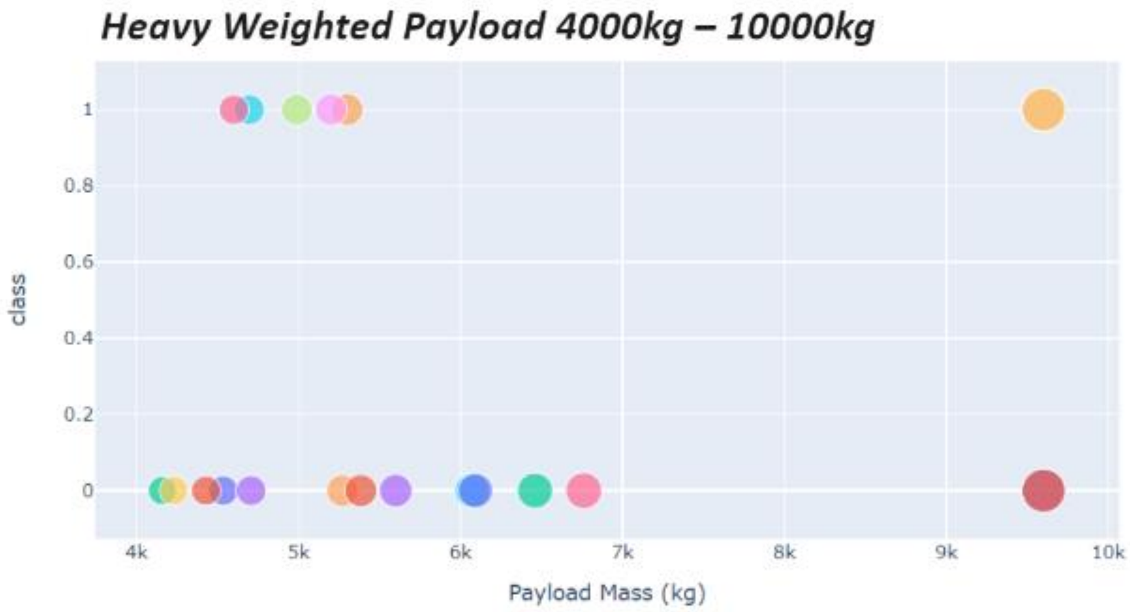
Pie chart showing the Launch site with the highest launch success ratio



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





We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

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

Find the method performs best:

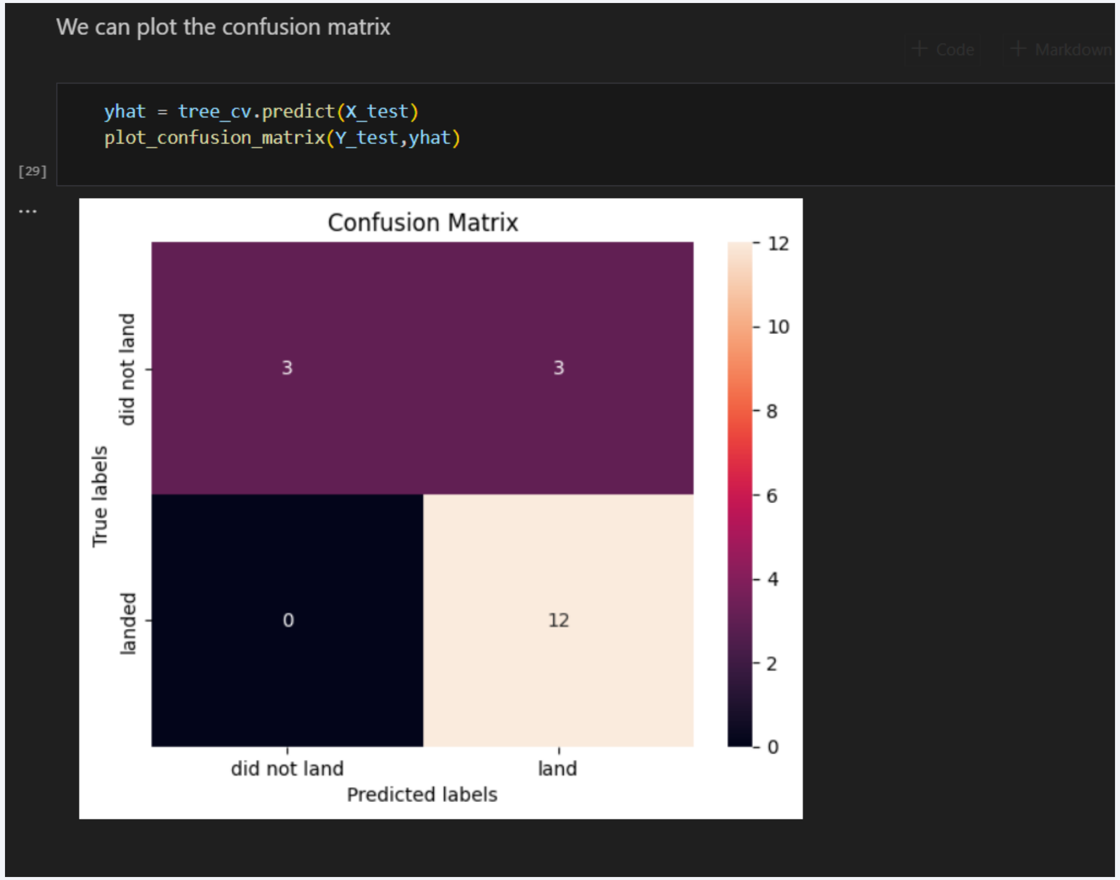
```python
accuracy = [svm_cv_score, logreg_score, knn_cv_score, tree_cv_score]
accuracy = [i * 100 for i in accuracy]

method = ['Support Vector Machine', 'Logistic Regression', 'K Nearest Neighbour', 'Decision Tree']
models = {'ML Method':method, 'Accuracy Score (%)':accuracy}

ML_df = pd.DataFrame(models)
ML_df

models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

[37]

```
Best model is DecisionTree with a score of 0.8767857142857143
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'random'}
```

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 a 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!