IBM DATA SCIENCE CAPSTONE PROJECT

SPACE X

OUTLINE

EXECUTIVE SUMMARY

SUMMARY OF METHODOLOGIES:

- Data collection
- Data wrangling
- EDA with data visualization and SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly dash
- Predictive Analysis using machine learning

SUMMARY OF ALL RESULTS:

- EDA results
- Dashboard analytics
- Predictive analysis

INTRODUCTION

Project background and context

We aimed to predict whether the Falcon 9's first stage would successfully land. SpaceX promotes Falcon 9 launches at a cost of $62 million, which is significantly lower compared to other providers that charge upwards of $165 million. Much of these savings come from SpaceX's ability to reuse the first stage of the rocket. By predicting whether the first stage will land, we can better estimate the overall cost of a launch. This insight could be valuable for other companies looking to compete with SpaceX in bidding for rocket launches.

Problems that needed solving:

- What determines whether the rocket will successfully land?
- Various rocket-related factors play a significant role in determining the likelihood of a successful landing. Each of these variables impacts the overall success rate of landing.
- Additionally, SpaceX needs to meet specific conditions to optimize performance and ensure a high probability of successful landings.

METHODOLOGY

METHODOLOGY

Data collection methodology:

- SpaceX Rest API and Web Scrapping

Performed Data Wrangling

- One Hot Encoding data fields for Machine Learning and dropping irrelevant columns

Performed EDA using visualization and SQL

- Plotting : Scatter, Bar, Pie Graphs to show relationships between variables to show patterns of data.

Performed interactive visual analytics using Folium and Plotly Dash

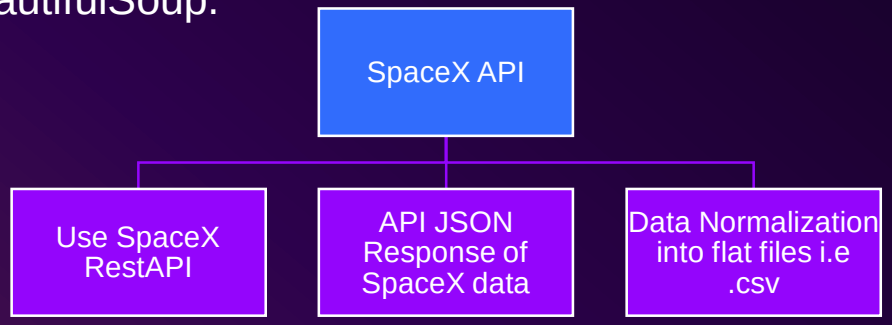
Performed predictive analysis using classification models

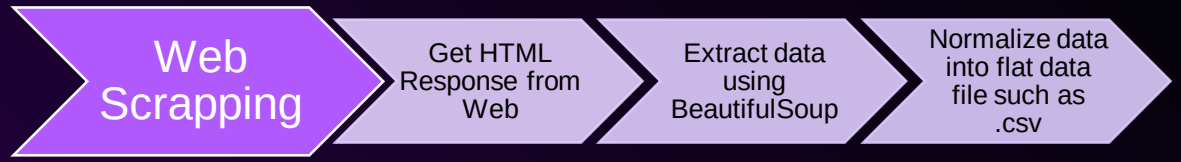
- building, tuning and evaluating classification model

METHODOLOGY

The following datasets were collected by:

- We analyzed SpaceX launch data retrieved from the SpaceX REST API.
- This API provides detailed information about launches, including data on the rocket used, the payload, and specifics related to the launch and landing, including the outcome of the landing.
- The objective of our analysis is to predict whether SpaceX will attempt to land the rocket during a given launch.
- The SpaceX REST API's base URL starts with api.spacexdata.com/v4/.
- Additionally, Wikipedia is another popular source for Falcon 9 launch data, accessible via web scraping using BeautifulSoup.





Data Collection – SpaceX API

1. Get Response from SpaceX API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2. Response updated to .json format

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

3. Data Cleaning

```
# Call getLaunchSite
getLaunchSite(data)
# Call getPayloadData
getPayloadData(data)
# Call getCoreData
getCoreData(data)
```

```
# Call getBoosterVersion
getBoosterVersion(data)
```

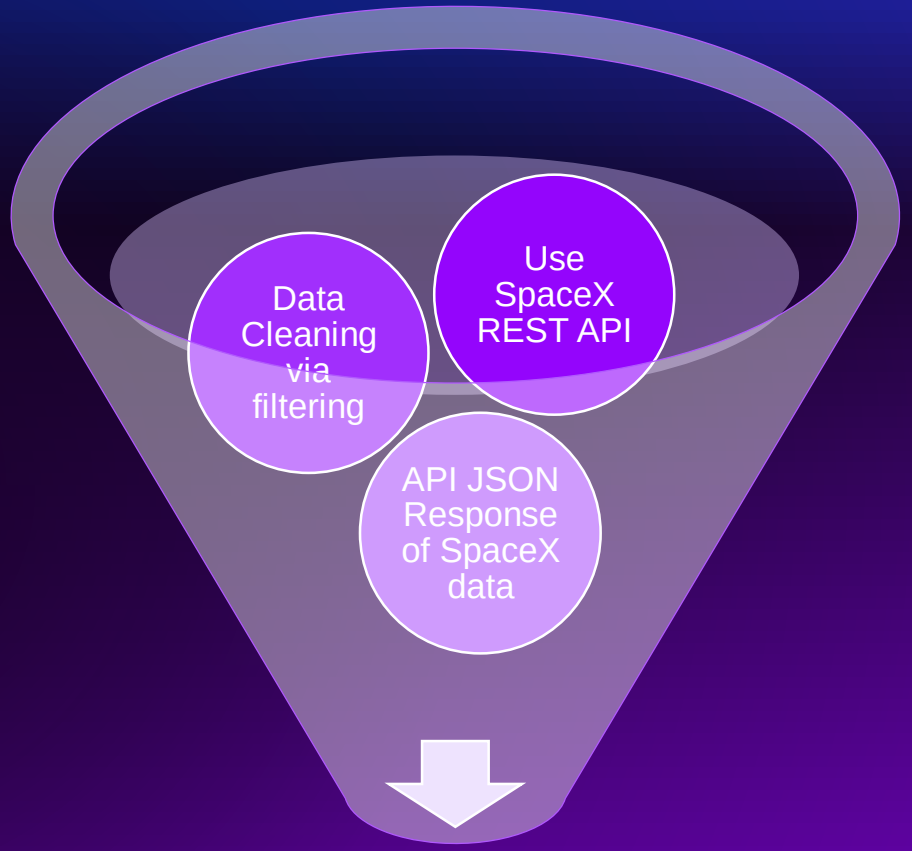
4. Combine the columns into a dictionary.

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

5. Filter Dataframe and export to .csv format

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

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



Normalized Data into .csv format

GitHub URL to Notebook

Data Collection – Web Scrapping

1. Get Response from HTML

```
# use requests.get() method with the provided static_url
page = requests.get(static_url)
```

2. Making BeautifulSoup object

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

3. Locating tables

```
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

4. Get column names

```
column_names = []
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
```

5. Dictionary creation

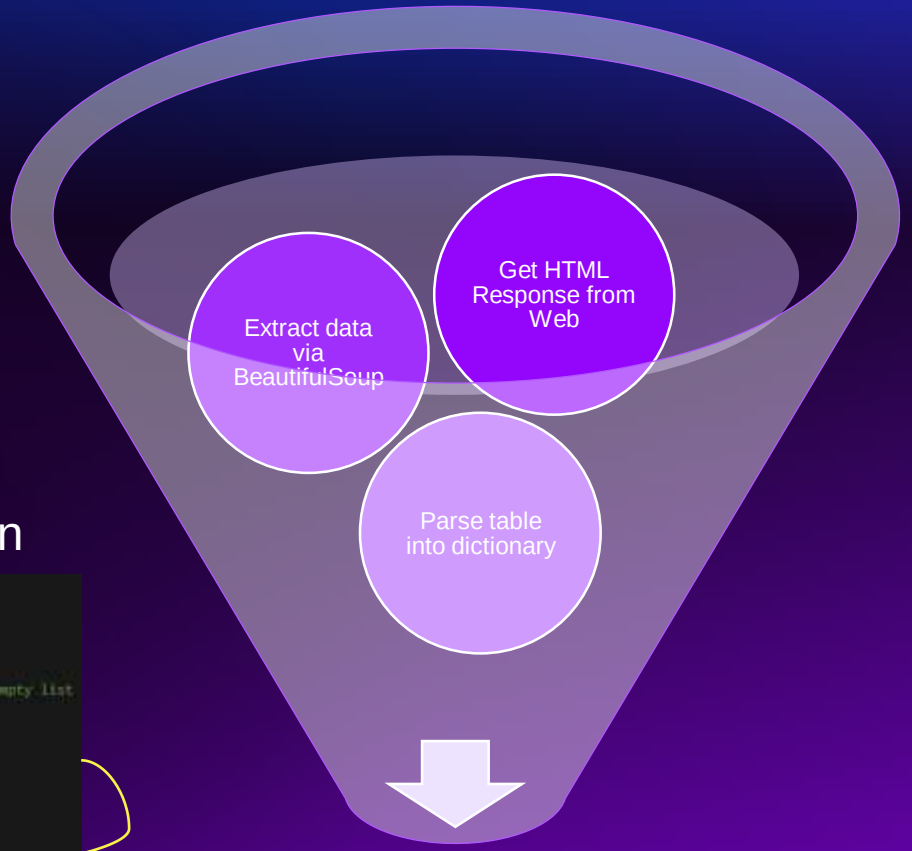
```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
_dict['Orbit'] = []
_dict['Customer'] = []
_dict['Launch outcome'] = []
d some new columns
_dict['Version Booster']=[]
_dict['Date']=[]
_dict['Time']=[]
```

6. Appending data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
    # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
```

7. Convert to DataFrame and export to .csv format

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df.to_csv('spacex_web_scraped.csv', index=False)
```



Normalized Data into .csv format

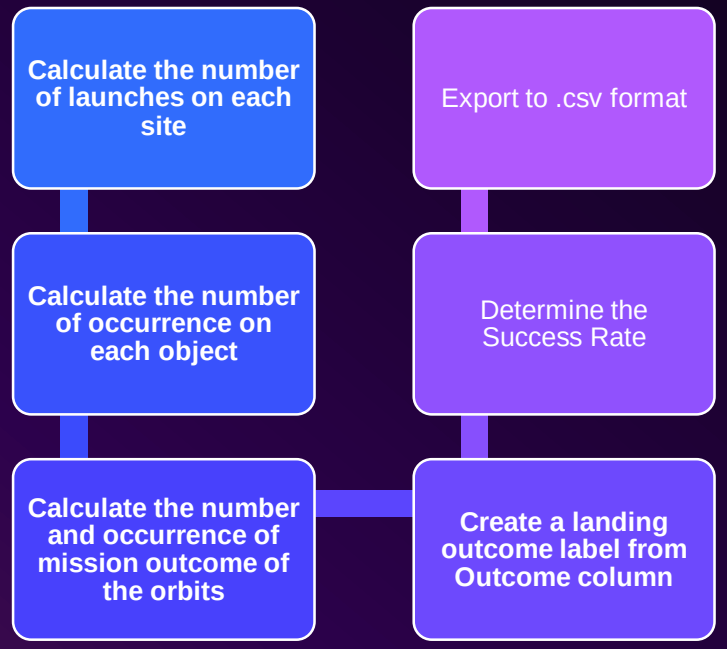
GitHub URL to Notebook

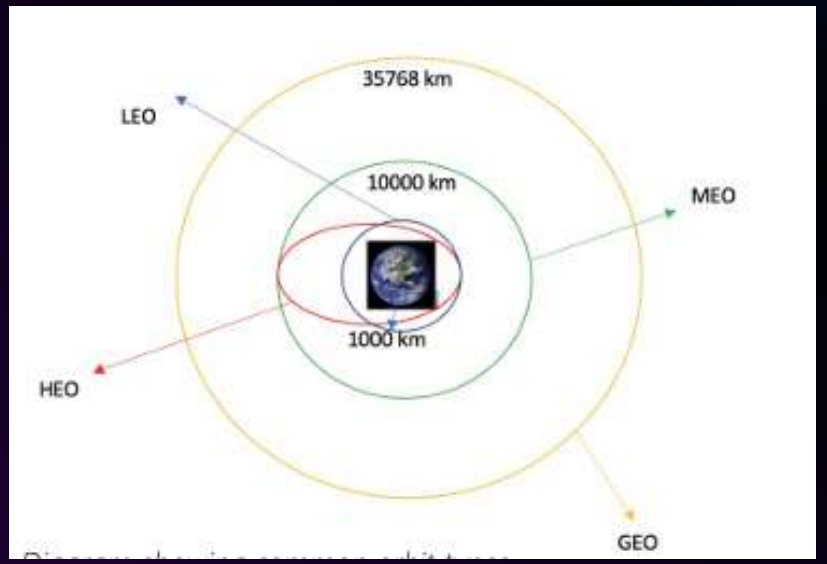
DATA WRANGLING

GitHub URL to Notebook

In the dataset, there are several instances where the booster did not successfully land. In some cases, a landing was attempted but failed due to an accident. For example, "True Ocean" indicates the booster successfully landed in a specific region of the ocean, whereas "False Ocean" means the landing attempt failed in that region. Similarly, "True RTLS" refers to a successful landing on a ground pad, while "False RTLS" indicates failure. "True ASDS" means the booster successfully landed on a drone ship, and "False ASDS" means the landing attempt on the drone ship was unsuccessful. These outcomes were converted into training labels: 1 for successful landings and 0 for unsuccessful ones.

Process:





SpaceX Orbit types

EDA WITH DATA VISUALIZATION

GitHub URL to Notebook

SCATTER GRAPH:

Scatter plots illustrate how one variable is influenced by another. The connection between the two variables is known as their **correlation**. Typically, scatter plots feature a large amount of data points, making it easier to visualize trends, patterns, or outliers in the relationship between the variables.

- Flight Number VS. Payload Mass
- Flight Number VS. Launch Site
- Payload VS. Launch Site
- Orbit VS. Flight Number
- Payload VS. Orbit Type
- Orbit VS. Payload Mass

BAR GRAPH:

A bar chart allows for quick and easy comparison of data across different categories. One axis represents the categories, while the other axis shows discrete values, illustrating the relationship between them. Bar charts are particularly useful for showing significant changes in data over time or highlighting differences between groups.

- Mean VS. Orbit

LINE GRAPH:

Line graphs are highly effective for clearly displaying data variables and trends, making it easier to understand changes over time. They are also helpful for making predictions about future data points based on trends in the recorded data.

- Success Rate VS. Year

EDA WITH SQL

GitHub URL to Notebook

Executed SQL queries to extract relevant information from the dataset for analysis and exploration.

- Displayed the names of the unique launch sites in the space mission
- Displayed 5 records where launch sites begin with the string 'KSC'
- Displayed the total payload mass carried by boosters launched by NASA (CRS)
- Displayed average payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes.
- Listing the names of the booster_versions which have carried the maximum payload mass.
- Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Ranking the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order

BUILT AN INTERACTIVE MAP WITH FOLIUM

To visualize the launch data on an interactive map, we plotted the latitude and longitude coordinates of each launch site and marked them with circle markers labeled with the launch site names. We categorized launch outcomes (successes and failures) into classes 0 and 1, using green markers for successful launches and red for failures, and displayed them using a MarkerCluster() for better organization.

We also calculated the distance from each launch site to various landmarks using Haversine's formula to explore spatial patterns. Lines were drawn on the map to visually represent the distance to these landmarks, helping identify trends related to the launch site's surroundings.

After ploting distance lines to the proximities, the findings were:

* 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**

GitHub URL to Notebook

BUILT A DASHBOARD WITH PLOTLY DASH

The dashboard is built using the Flask and Dash web frameworks, offering interactive visualizations of the launch data.

Graphs include:

- **Pie Chart:** Displays the total number of launches by specific sites or across all sites, illustrating the relative proportions of different categories of data. The size of the circle can be adjusted to reflect the quantity it represents.
- **Scatter Graph**: Shows the relationship between the launch outcome and the payload mass (kg) for various booster versions. This visualization highlights non-linear patterns and the data range, with clear observation points for determining maximum and minimum values. This type of graph is ideal for identifying trends in two variables.

GitHub URL to Notebook

PREDICTIVE ANALYSIS (CLASSIFICATION)

MODEL CONSTRUCTION

- Load dataset: Begin by loading the dataset into NumPy and Pandas for data manipulation and analysis.
- Data Transformation: Clean and preprocess the data to make it suitable for model training.
- Data Splitting: Divide the dataset into training and test sets to evaluate model performance.
- Sample Check: Verify the number of test samples to ensure an appropriate split.
- Algorithm Selection: Choose machine learning algorithms that fit the nature of the problem.
- Hyperparameter Tuning: Use GridSearchCV to set parameters and find the best combination of hyperparameters for the selected algorithms.
- Model Training: Train the models using the training set by fitting the data into the GridSearchCV object.

MODEL EVALUATION

- Accuracy Check: Assess the performance of each model by calculating accuracy.
- Hyperparameter Results: Retrieve the tuned hyperparameters that work best for each algorithm.
- Confusion Matrix: Plot a confusion matrix to visually evaluate the performance of the classification model.

MODEL IMPROVEMENT

- Feature Engineering: Refine the features to improve model performance.
- Algorithm Tuning: Further fine-tune the algorithms to optimize results.

FINDING THE BEST PERFORMING CLASSIFICATION MODEL

- Model Selection: The model with the highest accuracy score is considered the best performing model.
- Algorithm Dictionary: The notebook contains a dictionary of algorithms, along with their corresponding scores, to easily compare performance across different models.

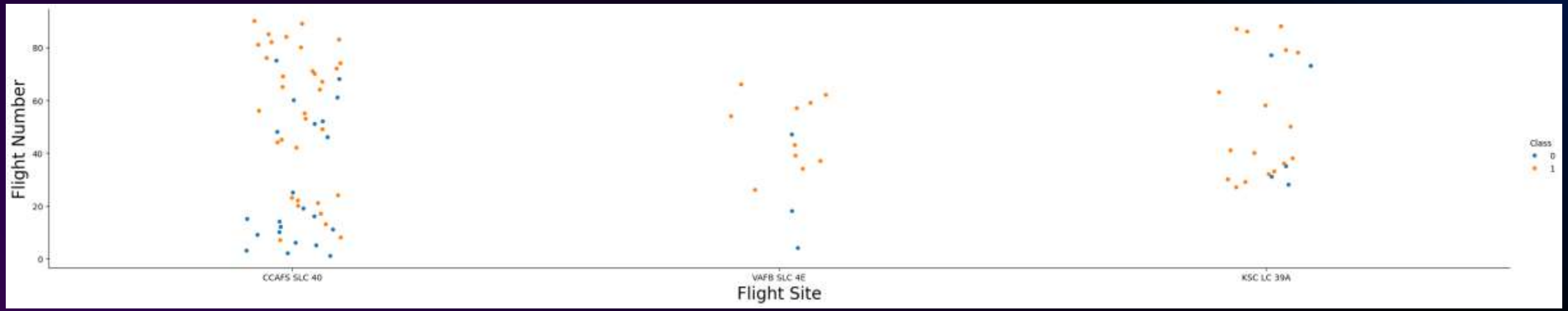
GitHub URL to Notebook

RESULTS

- EDA RESULTS
- DASHBOARD ANALYTICS
- PREDICTIVE ANALYSIS

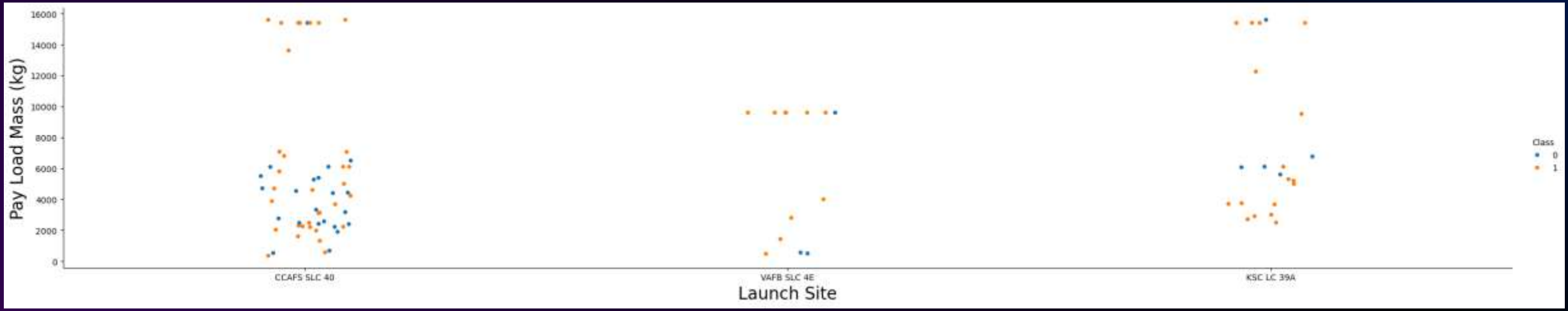
INSIGHTS DRAWN FROM EDA

FLIGHT NUMBER VS. FLIGHT SITE



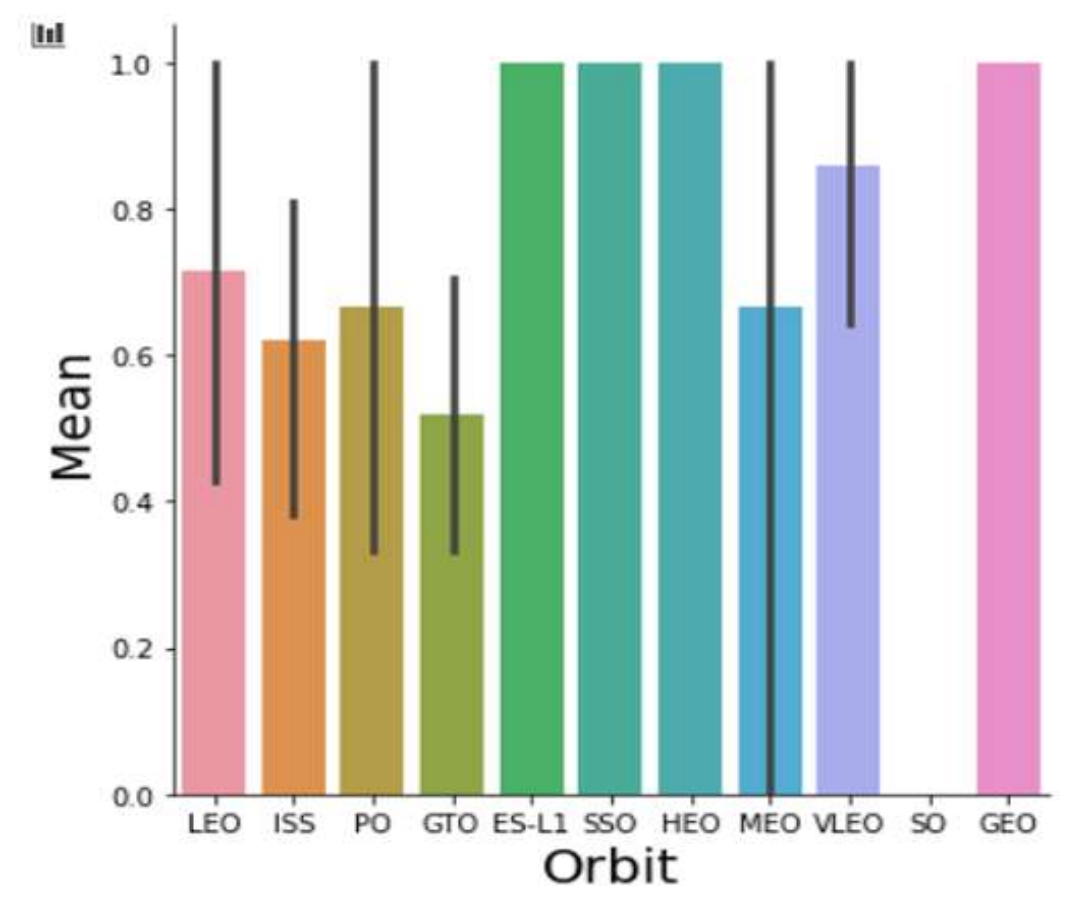
Greater the amount of flights at a launch site the higher the success rate

PAYLOAD MASS VS. LAUNCH SITE



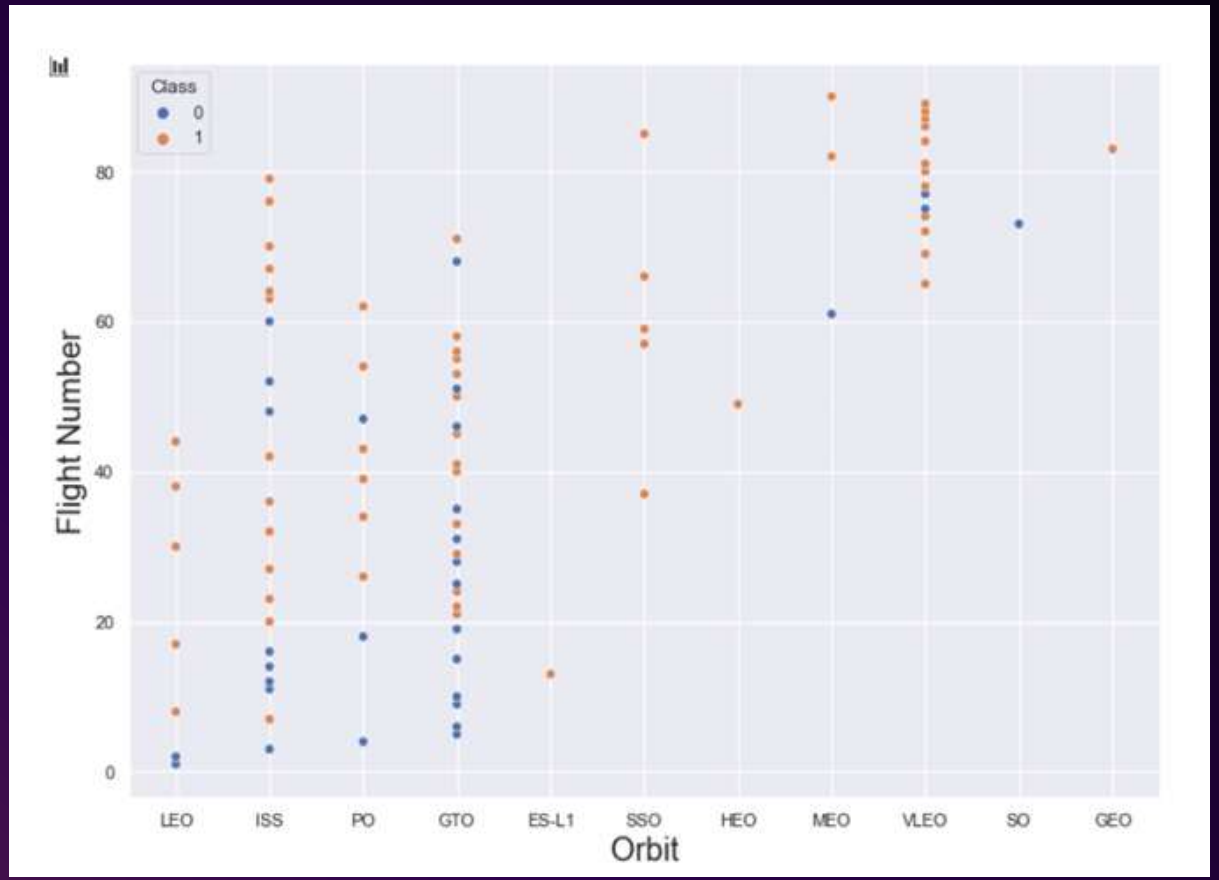
A higher payload mass at the CCAFS SLC 40 launch site correlates with a greater success rate for rocket landings. However, this visualization does not provide a definitive pattern to determine whether the success of the launch is solely dependent on payload mass at this particular launch site. The relationship between payload mass and launch success remains inconclusive based on the current data.

SUCCESS RATE VS. ORBIT TYPE



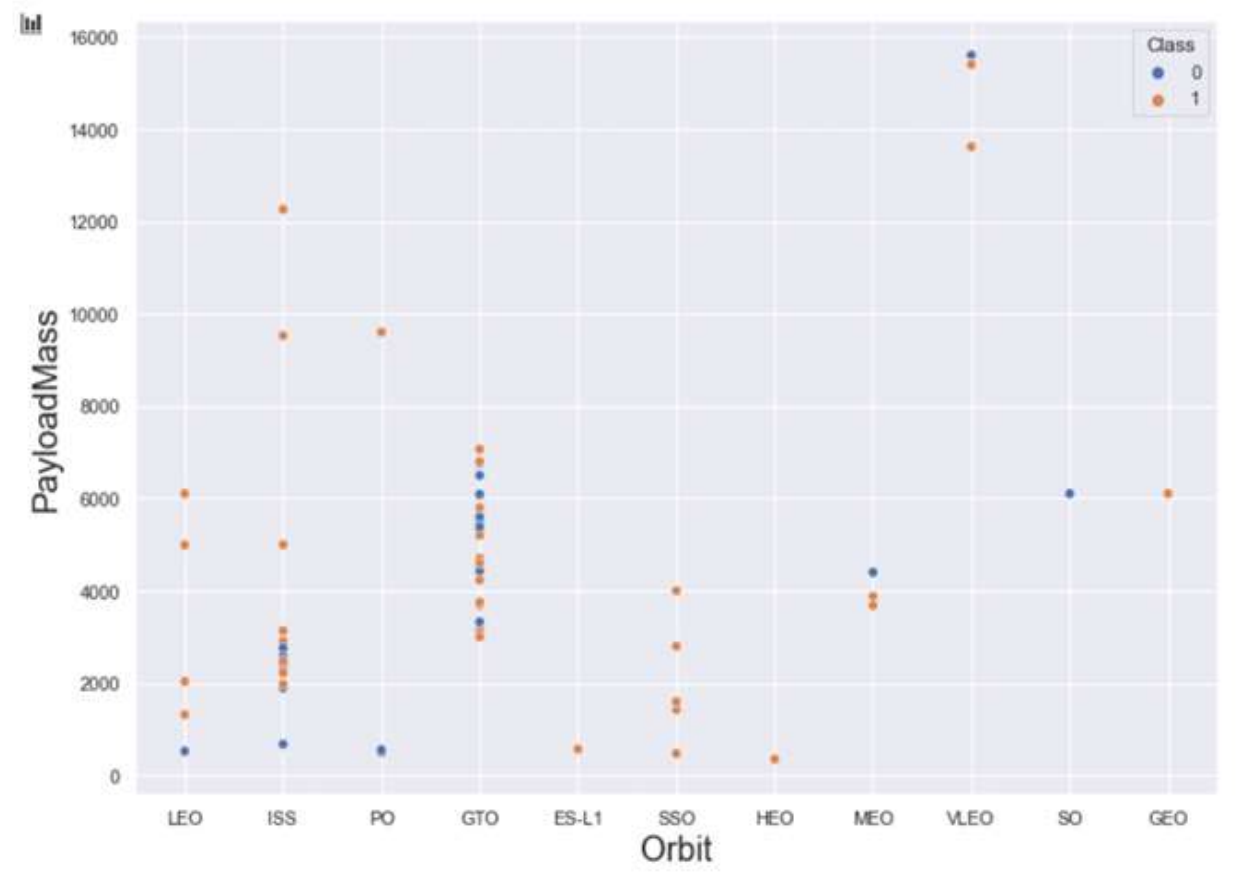
Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

FLIGHT NUMBER VS. ORBIT TYPE



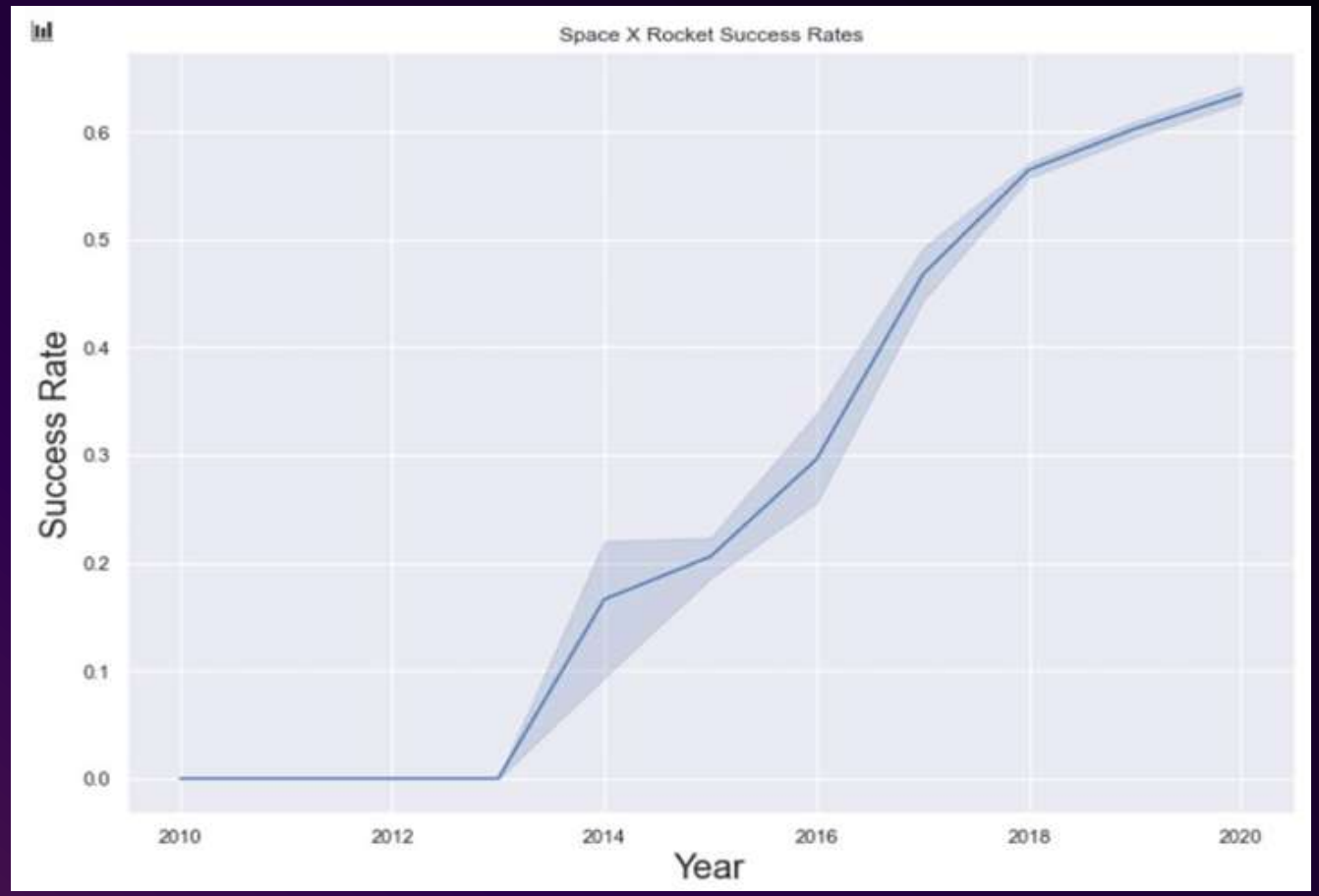
In the LEO (Low Earth Orbit), the data shows a positive correlation between the number of flights and successful landings. As the number of flights increases, the success rate of rocket landings tends to improve. On the other hand, when analyzing GTO (Geostationary Transfer Orbit), there seems to be no clear relationship between the number of flights and landing success, indicating that other factors may play a more significant role in determining the outcome for GTO launches.

PAYLOAD VS. ORBIT



*It can be observed that **heavy payloads** negatively impact rocket success rates in **GTO (Geostationary Transfer Orbit)** missions. In contrast, heavier payloads seem to have a positive effect on success rates for **GTO** and **Polar LEO (Low Earth Orbit)** missions. This suggests that while increased mass may pose challenges for certain orbits, it can enhance performance in others. Understanding these dynamics is crucial for optimizing launch strategies based on payload weight and orbit type.*

LAUNCH SUCCESS VS. YEARLY TREND



Success rate has been on a rise since 2013 and is increasing till 2020.

EDA WITH

SQL

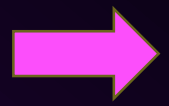
UNIQUE LAUNCH SITE NAMES

SQL QUERY

SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL

Query Explanation

The use of the keyword **DISTINCT** in the SQL query ensures that only unique values from the Launch_Site column in the SPACEXTBL table are returned.

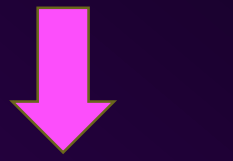


Unique Launch Sites
CCAFS LC-40
CCAFS SLC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

LAUNCH SITE NAMES BEGIN WITH ‘CCA’

SQL QUERY

SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;



Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

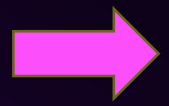
Query Explanation

The use of **LIMIT 5** in the query limits the results to only five records from the **SPACEXTBL** table. Additionally, the LIKE keyword, combined with the wildcard **‘CCA%’**, specifies that the Launch_Site name must start with CCA, effectively filtering for launch sites that begin with those characters.

TOTAL PAYLOAD MASS BY CUSTOMER NASA (CRS)

SQL QUERY

```
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL WHERE Customer =
'NASA (CRS)';
```



	Total Payload Mass
0	45596

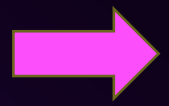
Query Explanation

The **SUM** function calculates the total value in the **PAYLOAD_MASS_KG_** column. The **WHERE** clause restricts the dataset to include only records associated with Customer NASA (CRS), ensuring that the summation reflects the payload mass specifically for that customer.

AVERAGE PAYLOAD MASS CARRIED BY BOOSTER VERSION F9 V1.1

SQL QUERY

```
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL WHERE
Booster_Version LIKE 'F9 v1.1%';
```



Average Payload Mass
0 2928

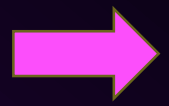
Query Explanation

The **AVG** function computes the average value in the **PAYLOAD_MASS_KG_** column. The **WHERE** clause filters the dataset to ensure that calculations are only performed on records corresponding to the **Booster_version F9 v1.1**. This allows for an accurate average that reflects only the specified booster version's payload mass.

FIRST SUCCESSFUL GROUND LANDING DATE

SQL QUERY

```
SELECT MIN(Date) FROM SPACEXTBL
WHERE Landing_Outcome = 'Success
(ground pad)'
```



MIN(Date)
2015-12-22

Query Explanation

Using the function **MIN** works out the minimum date in the column Date. The **WHERE** clause filters the dataset to only perform calculations on **Landing_Outcome = 'Success (ground pad)'**

SUCCESSFUL DRONE SHIP LANDING WITH PAYLOAD BETWEEN 4000 AND 6000

SQL QUERY

```
SELECT BOOSTER_VERSION FROM
SPACEXTBL WHERE LANDING_OUTCOME
= 'Success (drone ship)' AND 4000 <
PAYLOAD_MASS__KG_ < 6000;
```

Booster_Version
F9 FT B1021.1
F9 FT B1022
F9 FT B1023.1
F9 FT B1026
F9 FT B1029.1
F9 FT B1021.2
F9 FT B1029.2
F9 FT B1036.1
F9 FT B1038.1
F9 B4 B1041.1
F9 FT B1031.2
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

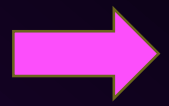
Query Explanation

Selecting only **Booster_Version**. The **WHERE** clause filters the dataset to **Landing_Outcome = 'Success (drone ship)'**, the **AND** clause specifies additional filter conditions **Payload_MASS_KG_** > 4000 AND **Payload_MASS_KG_** < 6000

TOTAL NUMBER OF SUCCESSFUL AND FAILURE MISSION OUTCOMES

SQL QUERY

```
SELECT MISSION_OUTCOME,
COUNT(MISSION_OUTCOME) AS
TOTAL_NUMBER FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```



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

Query Explanation

Using the **COUNT** function to see all the outcomes of every **Mission_Outcome.**

BOOSTERS CARRIED MAXIMUM PAYLOAD

SQL QUERY

```
SELECT DISTINCT BOOSTER_VERSION FROM
SPACEXTBL WHERE PAYLOAD_MASS__KG_ =
(SELECT MAX(PAYLOAD_MASS__KG_) FROM
SPACEXTBL);
```

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

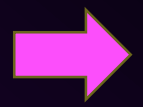
Query Explanation

Using the word **DISTINCT** in the query means that it will only show Unique values in the **Booster_Version** column from **SPACEXTBL.** Followed by a subquery to filter the max payload mass from the table.

2015 LAUNCH RECORDS

SQL QUERY

```
SELECT substr(Date, 6,2), LANDING_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE Landing_Outcome = 'Failure (drone ship)' AND substr(Date,0,5)='2015'
```



substr(Date, 6,2)	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

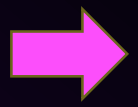
Query Explanation

This query extracts specific columns (**LANDING_OUTCOME, BOOSTER_VERSION, and LAUNCH_SITE**) from the **SPACEXTBL** table, filtering for records where the **LANDING_OUTCOME** is **'Failure (drone ship)'**. It also uses the **substr** function to extract parts of the Date column: the month (substr(Date, 6, 2)) and checks if the year (substr(Date, 0, 5)) is **'2015'**.

RANK SUCCESS COUNT BETWEEN 2010-06-04 AND 2017-03-20

SQL QUERY

SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING_OUTCOME ORDER BY TOTAL_NUMBER DESC



Landing_Outcome	TOTAL_NUMBER
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

Query Explanation

Function COUNT counts records in column. **WHERE filters data, LIKE (wildcard), AND (conditions), GROUPING BY** Landing Outcome in descending order.

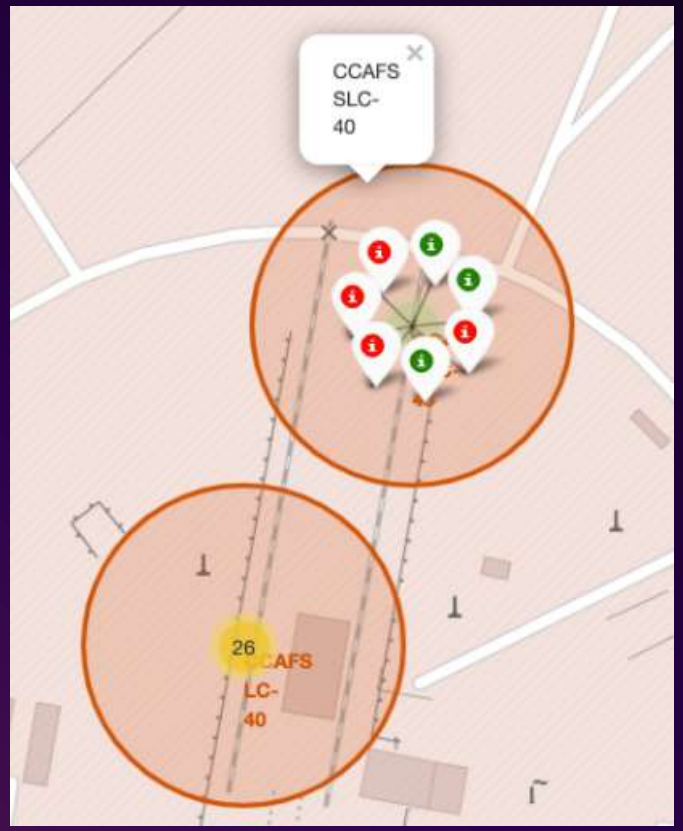
INTERACTIVE MAP WITH FOLIUM

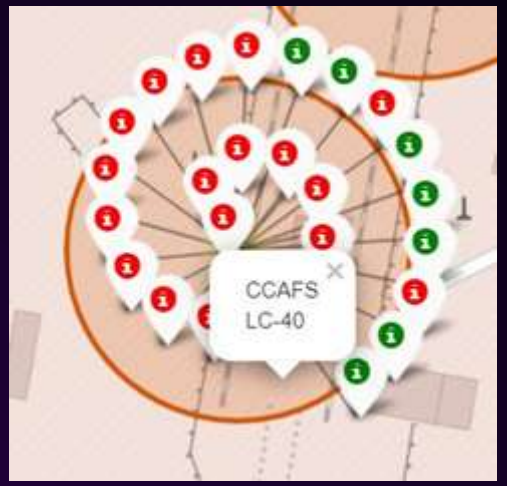
LAUNCH SITE PROXIMITIES ANALYSIS

ALL LAUNCH SITES GLOBAL MAP MARKERS

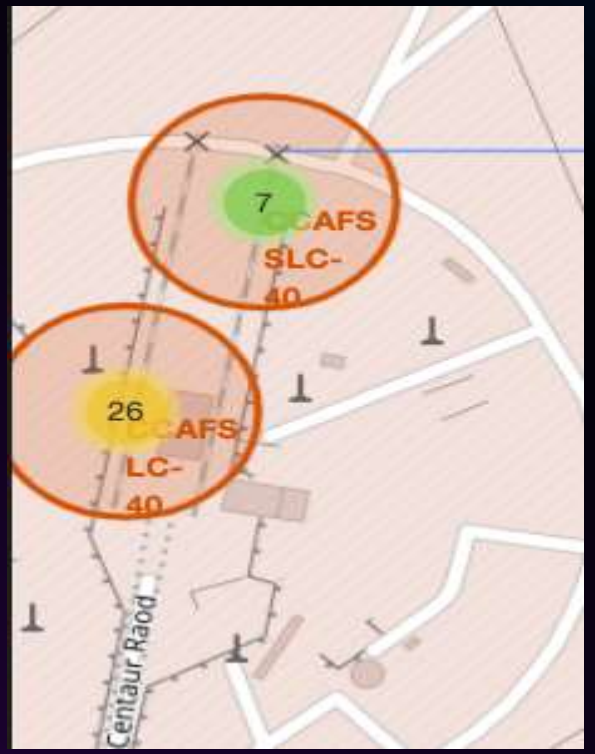
We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

COLOR LABELLED MARKERS



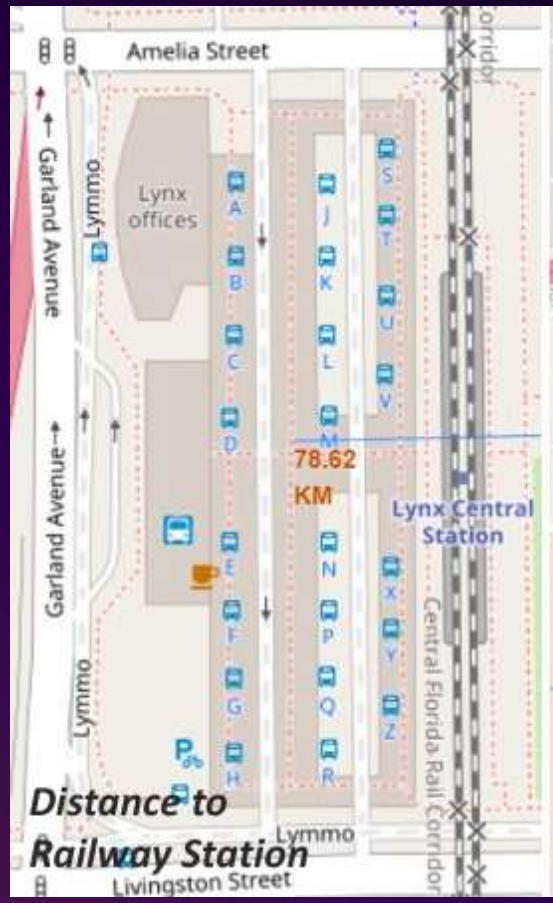


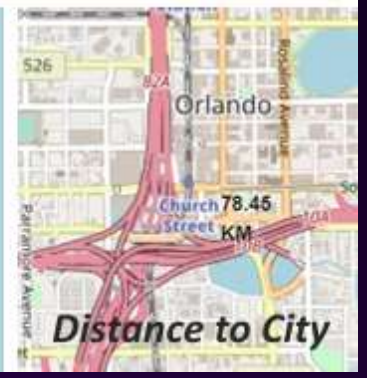
Florida launch sites

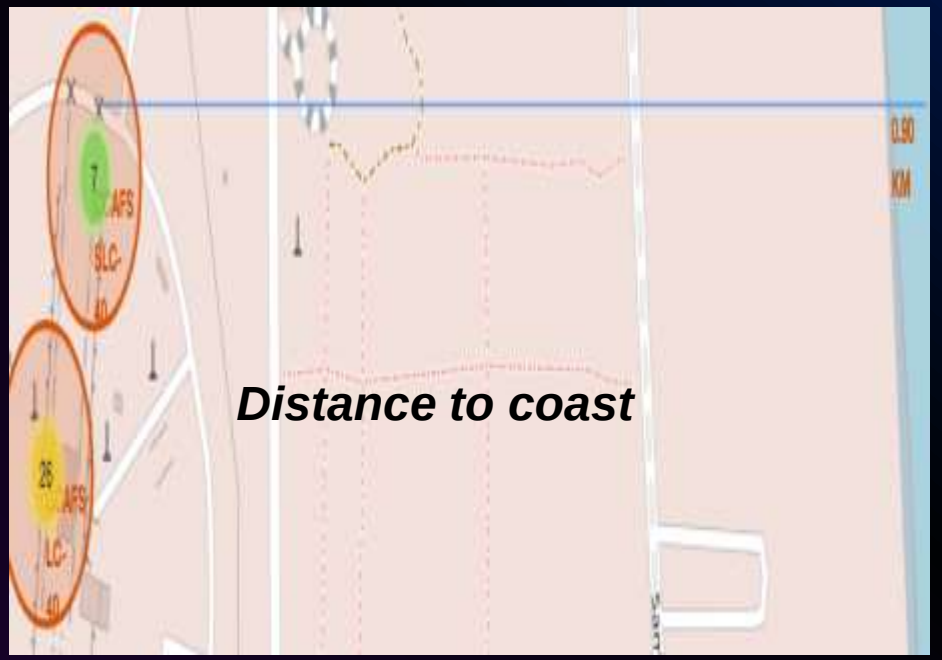


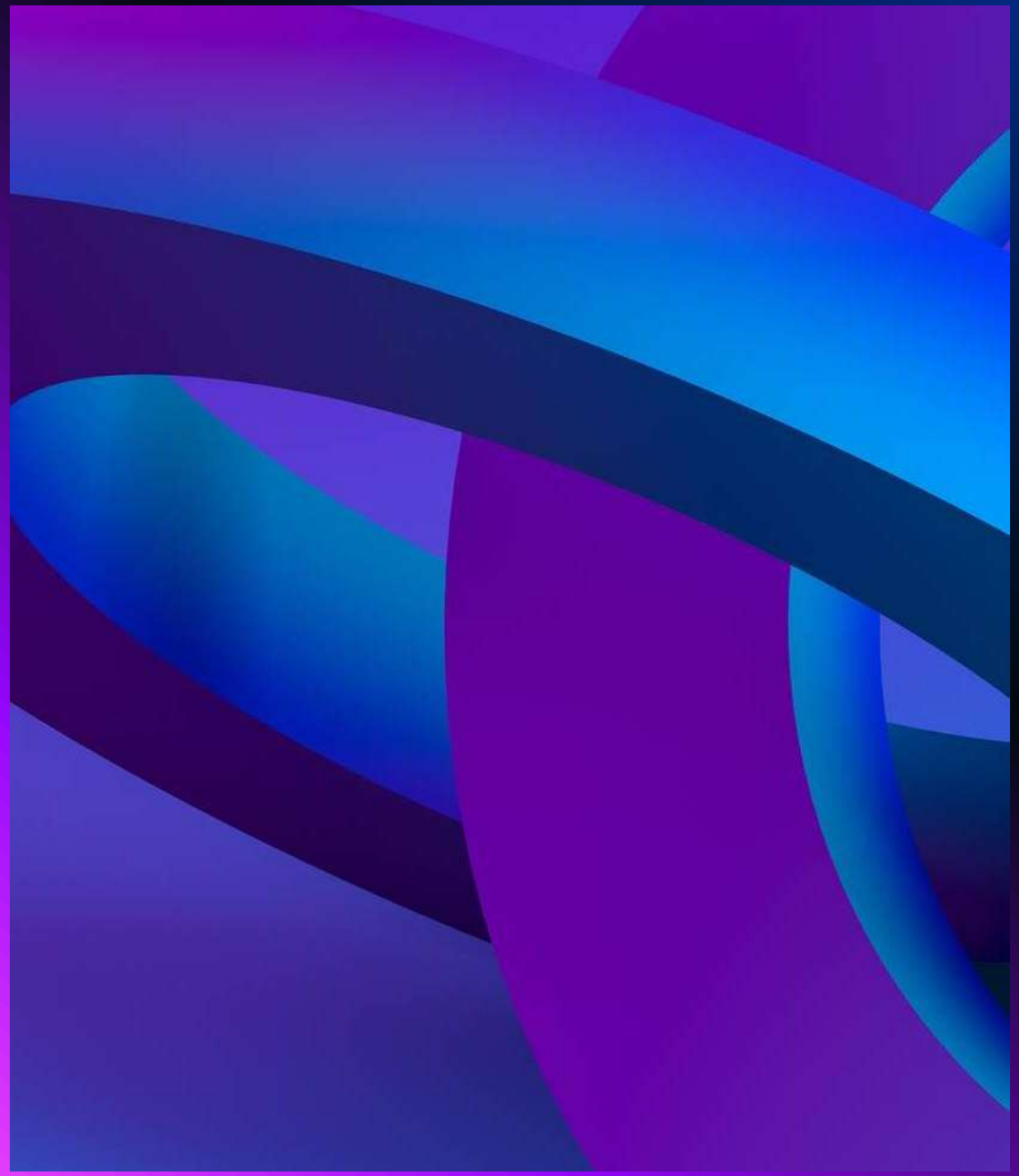
Cali launch sites

LAUNCH SITE TO ITS PROXIMITIES SUCH AS RAILWAY, HIGHWAY, COASTLINE, WITH DISTANCE CALCULATED



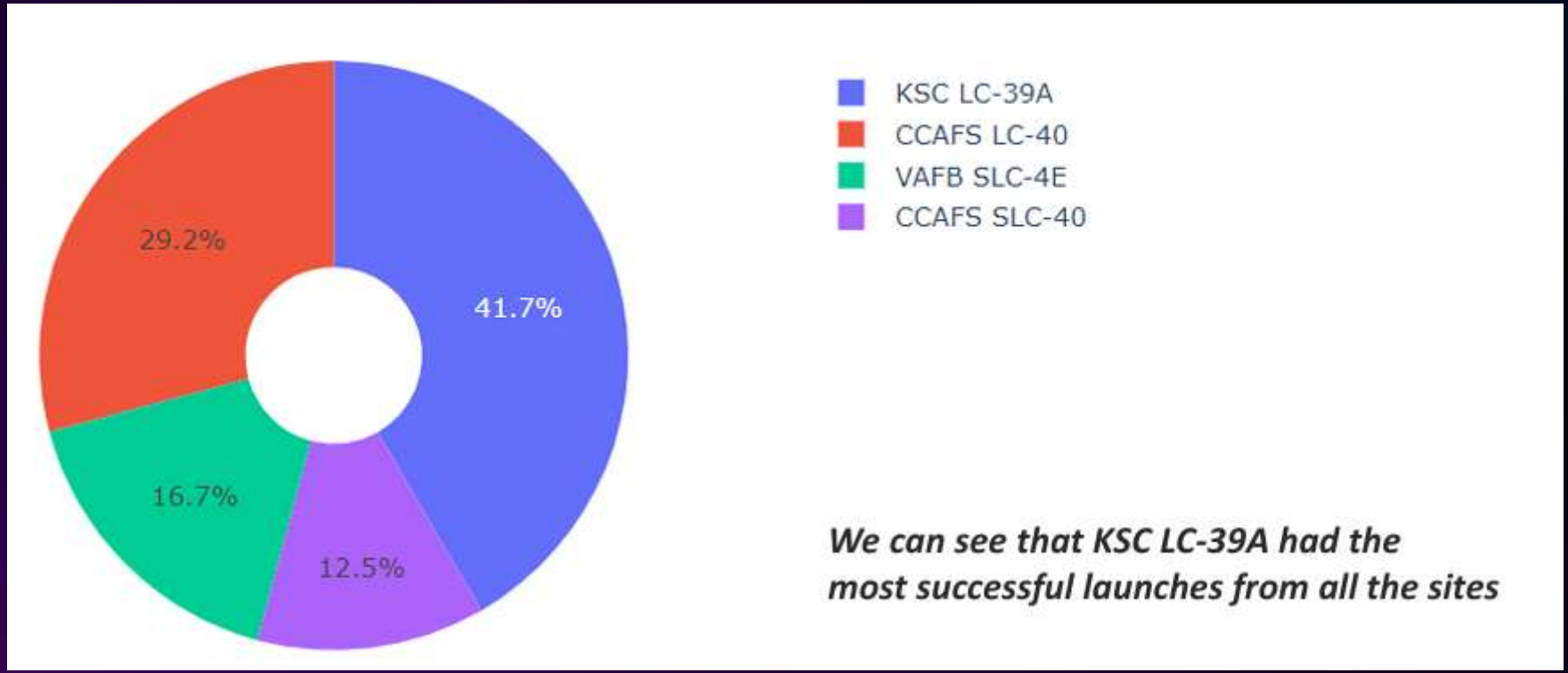




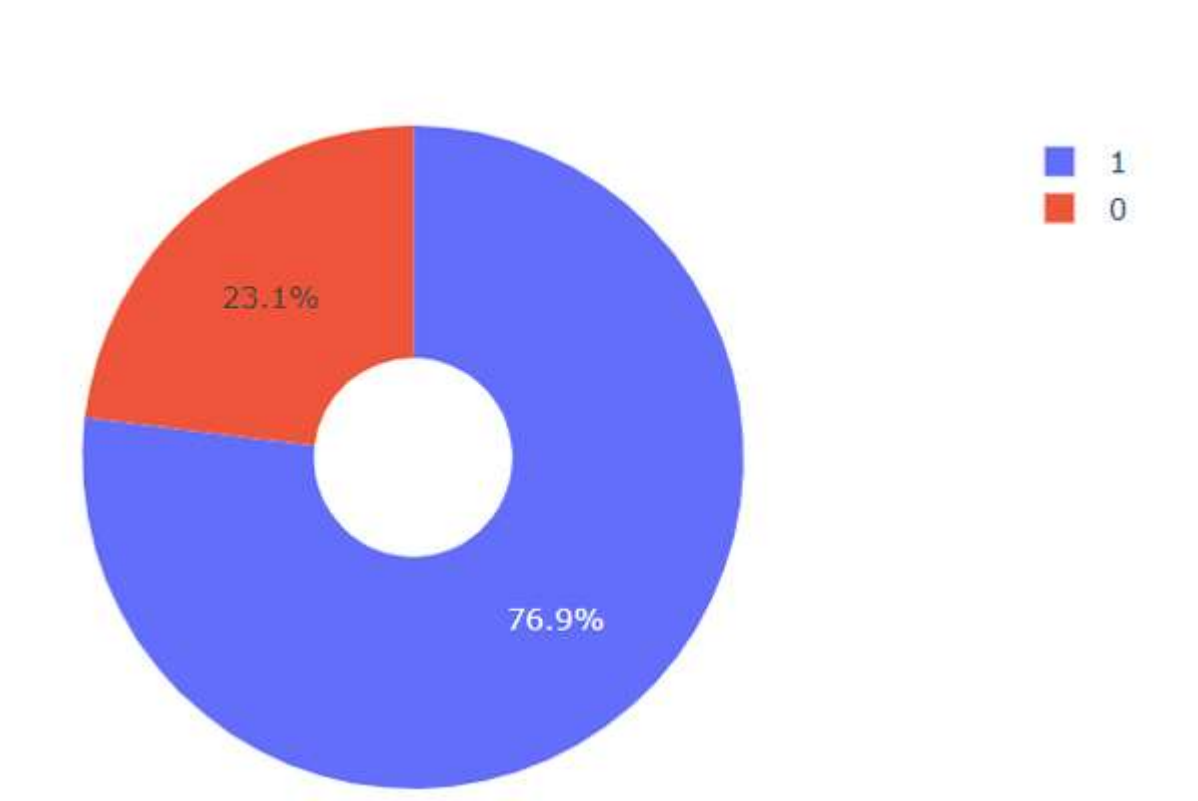


DASHBOARD WITH PLOTLY DASH

DASHBOARD–PIE CHART SHOWING THE SUCCESS PERCENTAGE ACHIEVED BY EACH LAUNCH SITE

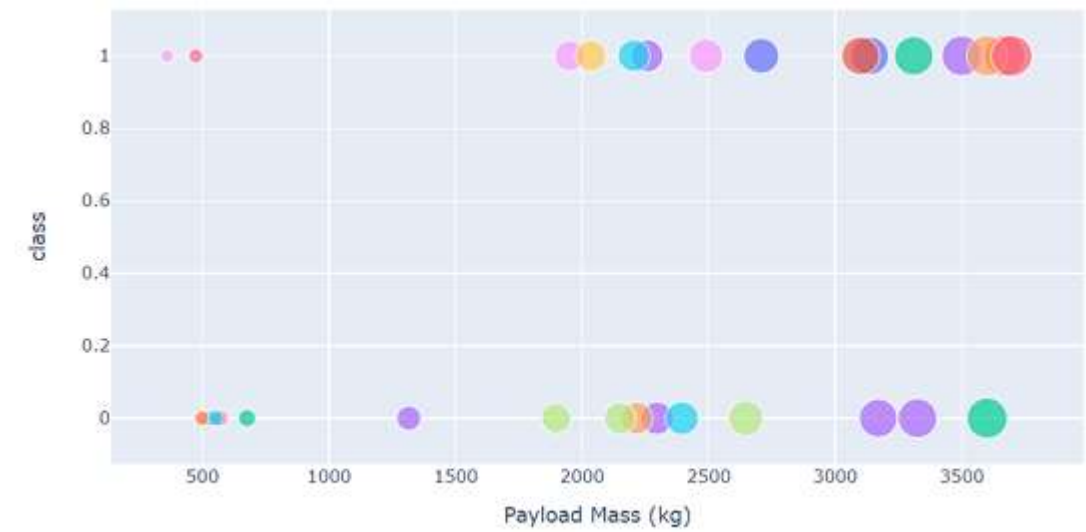


DASHBOARD–PIE CHART FOR THE LAUNCH SITE WITH HIGHEST LAUNCH SUCCESS RATIO

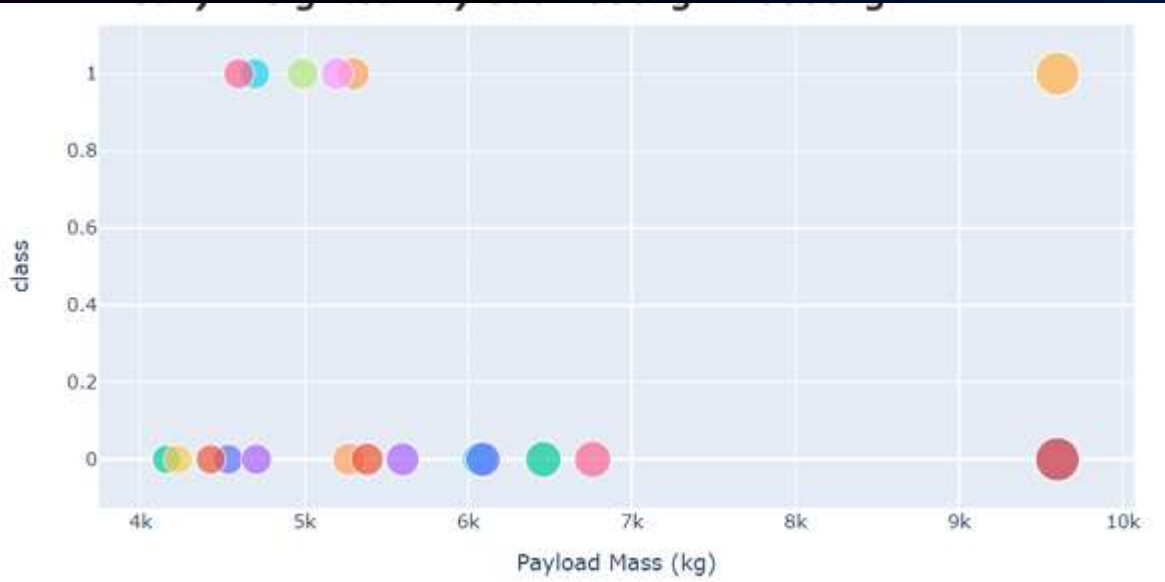


KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

PAYLOAD VS. LAUNCH OUTCOME SCATTER PLOT FOR ALL SITES, WITH DIFFERENT PAYLOAD SELECTED IN THE RANGE SLIDER



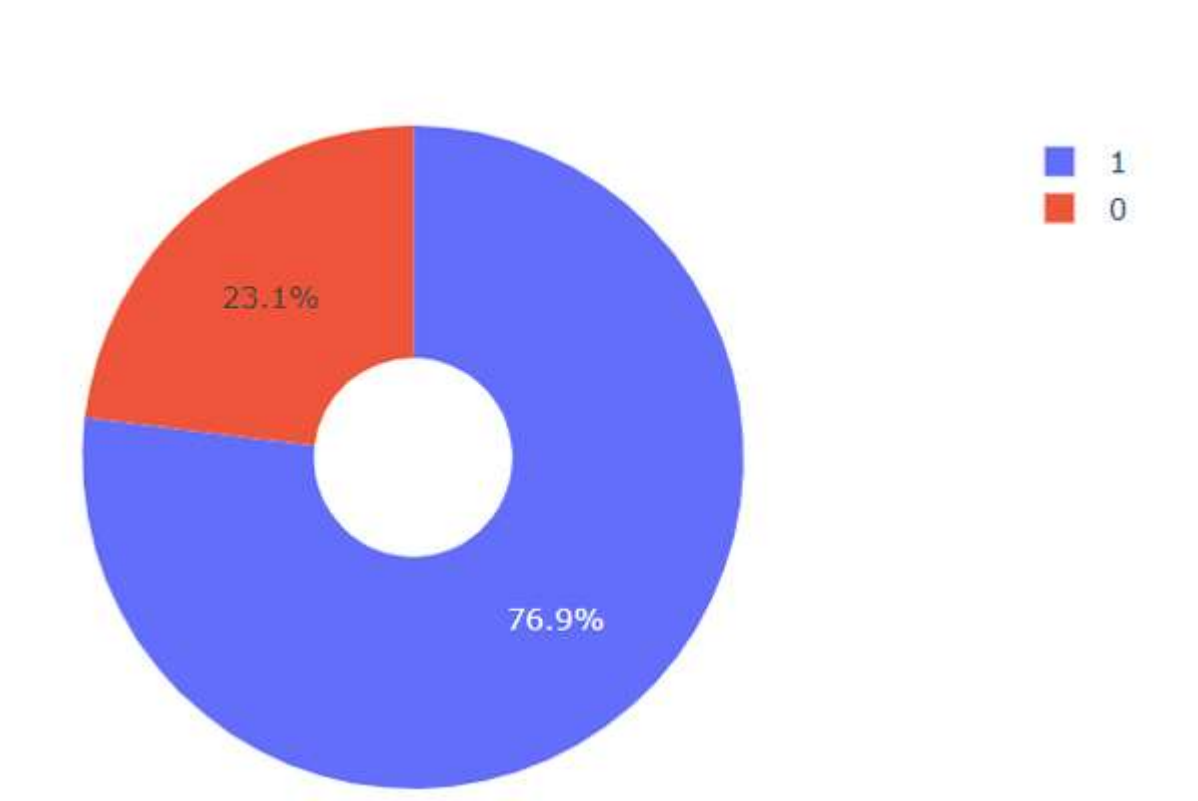
Low Weighted payload scatter plot



High Weighted payload scatter plot

The success rates for low weighted payloads is higher than the heavy weighted payload

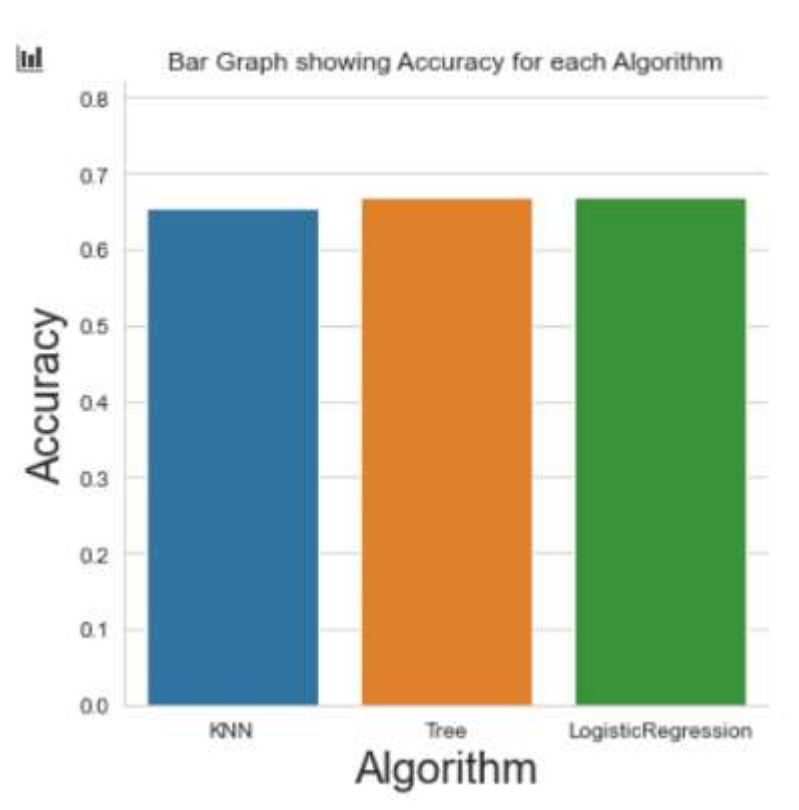
DASHBOARD–PIE CHART FOR THE LAUNCH SITE WITH HIGHEST LAUNCH SUCCESS RATIO



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

PREDICTIVE ANALYSIS (CLASSIFICATION)

CLASSIFICATION ACCURACY



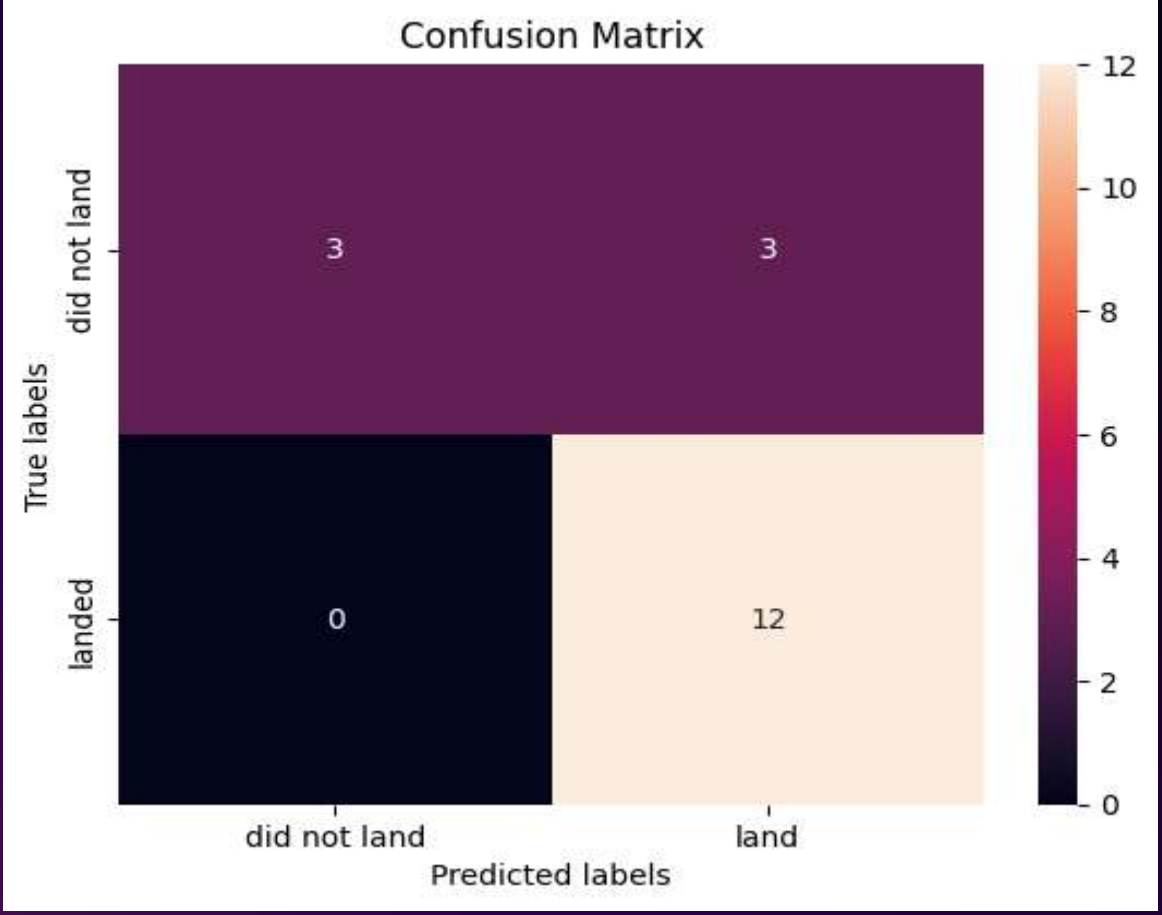
As you can see, the accuracy is remarkably close, but we do have a winner—it all comes down to the decimal points!

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8767857142857143
Best Params is : {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_sp
lit': 5, 'splitter': 'best'}
```

The tree algorithm is the winner! With a score of 87.76% and accuracy of 83.33% on test data.

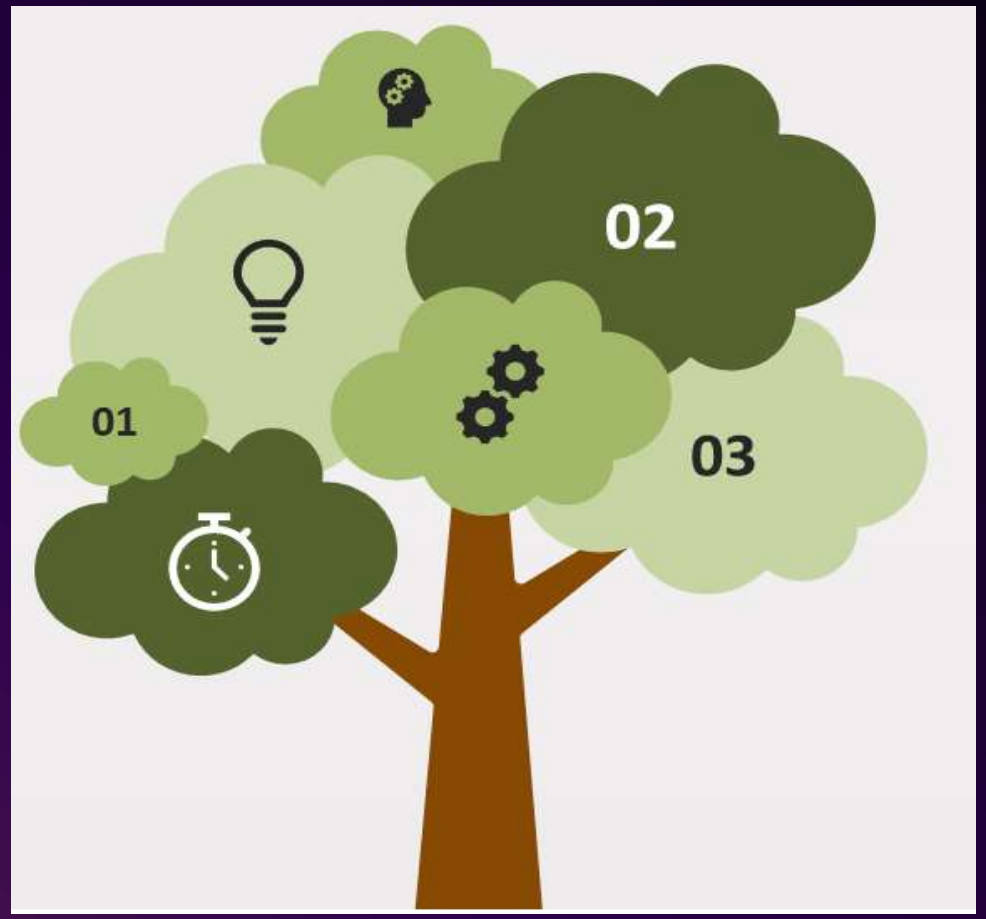
CONFUSION MATRIX – DECISION TREE



In summary:

- The model predicted 15 out of 18 cases correctly.
- The model made 3 incorrect predictions, all in the form of false positives (where it thought the rocket would land but it didn’t).
- The absence of false negatives suggests the model didn’t miss any actual landings.

CONCLUSION



- The Tree Classifier Algorithm stands out as the most effective machine learning technique for this dataset.
- Lighter payloads tend to show better performance compared to heavier payloads.
- The success rates of SpaceX launches appear to increase over the years, indicating continuous improvement and eventual launch perfection.
- Among all the launch sites, KSC LC-39A has recorded the highest number of successful launches.
- The GEO, HEO, SSO, and ES-L1 orbits demonstrate the best success rates for launches.

APPENDIX

```python
def plot_confusion_matrix(y,y_predict):
    "this function plots the confusion matrix"
    from sklearn.metrics import confusion_matrix

    cm = confusion_matrix(y, y_predict)
    ax= plt.subplot()
    sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells
    ax.set_xlabel('Predicted lab
    ax.set_ylabel('True labels')
    ax.set_title('Confusion Matr
    ax.xaxis.set_ticklabels(['di
    plt.show()
```

```python
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

```python
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
gscv = GridSearchCV(lr,parameters,scoring='accuracy',cv=10)
logreg_cv = gscv.fit(X_train,Y_train)
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```python
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

THANK YOU

Eesha Qureshi

eesha_qureshi@outlook.com