IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

<Renuka Hebasur>
<17-01-2025>

# Outline

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

# Executive Summary

- Summary of methodologies
  - The goal of this project was to analyze SpaceX Falcon 9 launch data, using various methods and tools:
    - Data Collection via API and Web Scraping
    - Data Wrangling and Label Creation
    - Exploratory Data Analysis (EDA) with SQL and Visualizations
    - Interactive Maps (Folium) and Dashboards (Plotly Dash)
    - Predictive Modeling with Logistic Regression, SVM, Decision Trees, and KNN
- Summary of all results
  - The report aims to assist space agencies in predicting the success of SpaceX first-stage landings and understanding the impact of launch parameters.

# Introduction

- Project background
  - With the rise of private space travel, SpaceX has positioned itself as a cost-effective leader by reusing first-stage boosters. Each SpaceX launch costs ~$62M, significantly lower than competitors
- Problems you want to find answers
  - Predict if the first stage of Falcon 9 will land successfully.
  - Analyze parameters like launch site, payload mass, and orbit type on landing success.
  - Correlate launch data with success rates.

Section 1

# Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - SpaceX API: Leveraged the SpaceX API to gather real-time data on all launches, including details like launch dates, rocket types, and mission outcomes.
  - Web scrape Falcon 9 and Falcon Heavy launch records from Wikipedia: Extracted comprehensive historical data on Falcon 9 and Falcon Heavy launches using web scraping techniques to supplement the API data.
- Perform data wrangling
  - Cleaned and organized the collected data to ensure accuracy, handling missing values and standardizing formats for consistency.
  - Determined labels for training the supervised models by converting mission outcomes into training labels (0-unsuccessful, 1-successful): Transformed the mission outcomes into a binary classification format for effective model training.

# Methodology

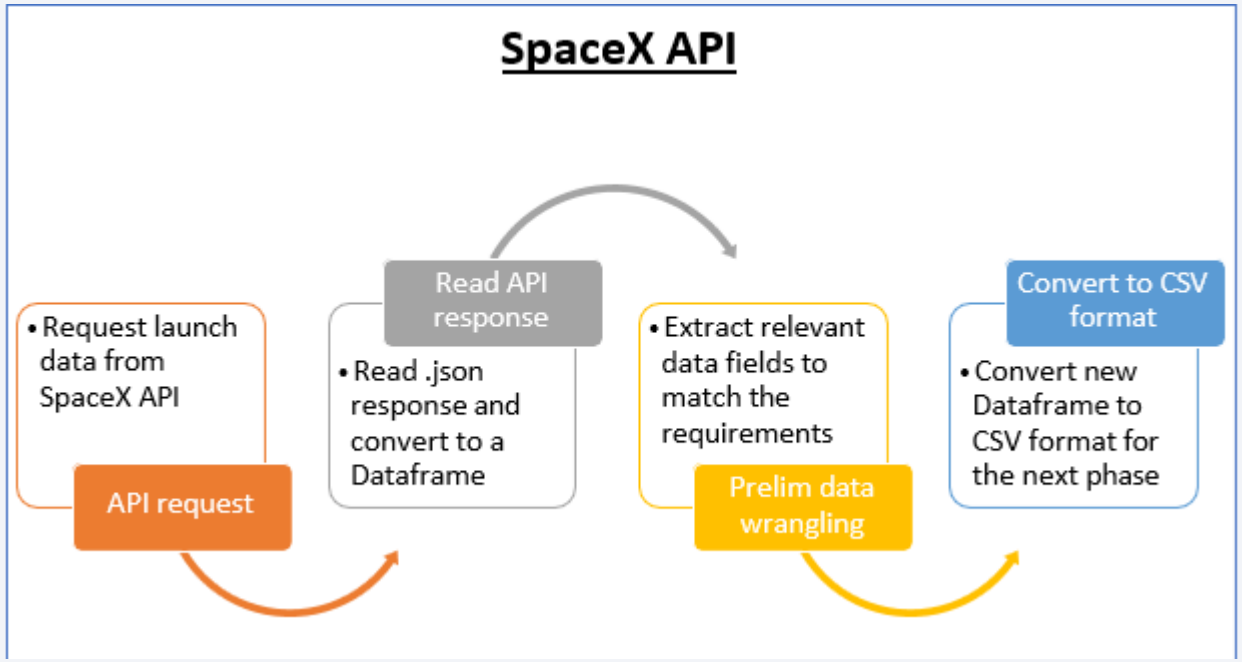
## Executive Summary

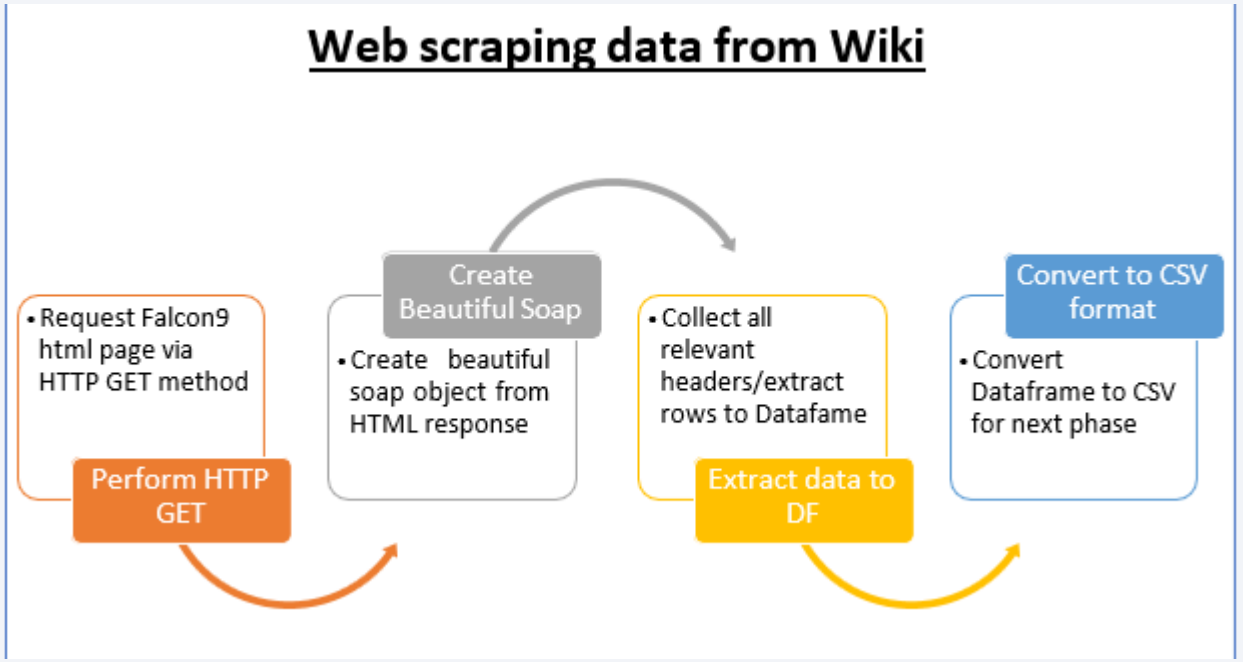
- Perform exploratory data analysis (EDA) using visualization and SQL:
  - Conducted EDA to identify trends, patterns, and correlations in the launch data, utilizing SQL queries and visualizations to present findings.
- Perform interactive visual analytics using Folium and Plotly Dash:
  - Created interactive dashboards to visualize launch trajectories and success rates, enhancing the understanding of SpaceX's performance over time.
- Perform predictive analysis using classification models:
  - Implemented various classification algorithms to predict the success of future launches based on historical data.

# Data Collection

Describe how data sets were collected.

- For this project, data was collected via SpaceX API and Web scrapping Wiki pages for relevant launch data.



# Data Collection – SpaceX API

1. API Request and read response into DF
2. Declare global variables
3. Call helper functions with API calls to populate global vars
4. Construct data using dictionary
5. Convert Dict to Dataframe, filter for Falcon9 launches, covert to CSV

1. Create API GET request, normalize data and read in to a Dataframe:

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

response = requests.get(spacex_url)

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

2. Declare global variable lists that will store data returned by helper functions with additional API calls to get relevant data

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

3. Call helper functions to get relevant data where columns have IDs (e.g., rocket column is an identification number)
   - getBoosterVersion(data)
   - getLaunchSite(data)
   - getPayloadData(data)
   - getCoreData(data)

4. Construct dataset from received data & combine 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}
```

4. Create Dataframe from dictionary and filter to keep only the Falcon9 launches:

```
# Create a data from launch_dict
df_launch = pd.DataFrame(launch_dict)
```

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df_launch[df_launch['BoosterVersion']!= 'Falcon 1']
```

```
data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

GitHub

# Data Collection - Scraping

1. Perform HTTP GET to request HTML page
2. Create Beautiful Soap object
3. Extract column names from HTML table header
4. Create Dictionary with keys from extracted column names
5. Call helper functions to fill up dict with launch records
6. Convert Dictionary to Dataframe

1. Create API GET method to request Falcon9 launch HTML page

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

html_data = requests.get(static_url).text
```

2. Create Beautiful Soap object

```
soup = BeautifulSoup(html_data,"html.parser")
```

3. Find all the tables on the Wiki page and extract relevant column names from the HTML table header

```
html_tables = soup.find_all ('table')
```

```
column_names = []

# Apply find_all() function with `th` element on firs
# Iterate each th element and apply the provided extr
# Append the Non-empty column name (`if name is not N
colnames = soup.find_all('th')
for x in range (len(colnames)):
    name2 = extract_column_from_header(colnames[x])
    if (name2 is not None and len(name2) > 3):
        column_names.append(name2)
```

4. Create an empty Dictionary with keys from extracted column names:

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each vc
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

5. Fill up the launch_dict with launch records extracted from table rows.
   - Utilize following helper functions to help parse HTML data

```
def date_time(table_cells):

def booster_version(table_cells):

def landing_status(table_cells):

def get_mass(table_cells):
```

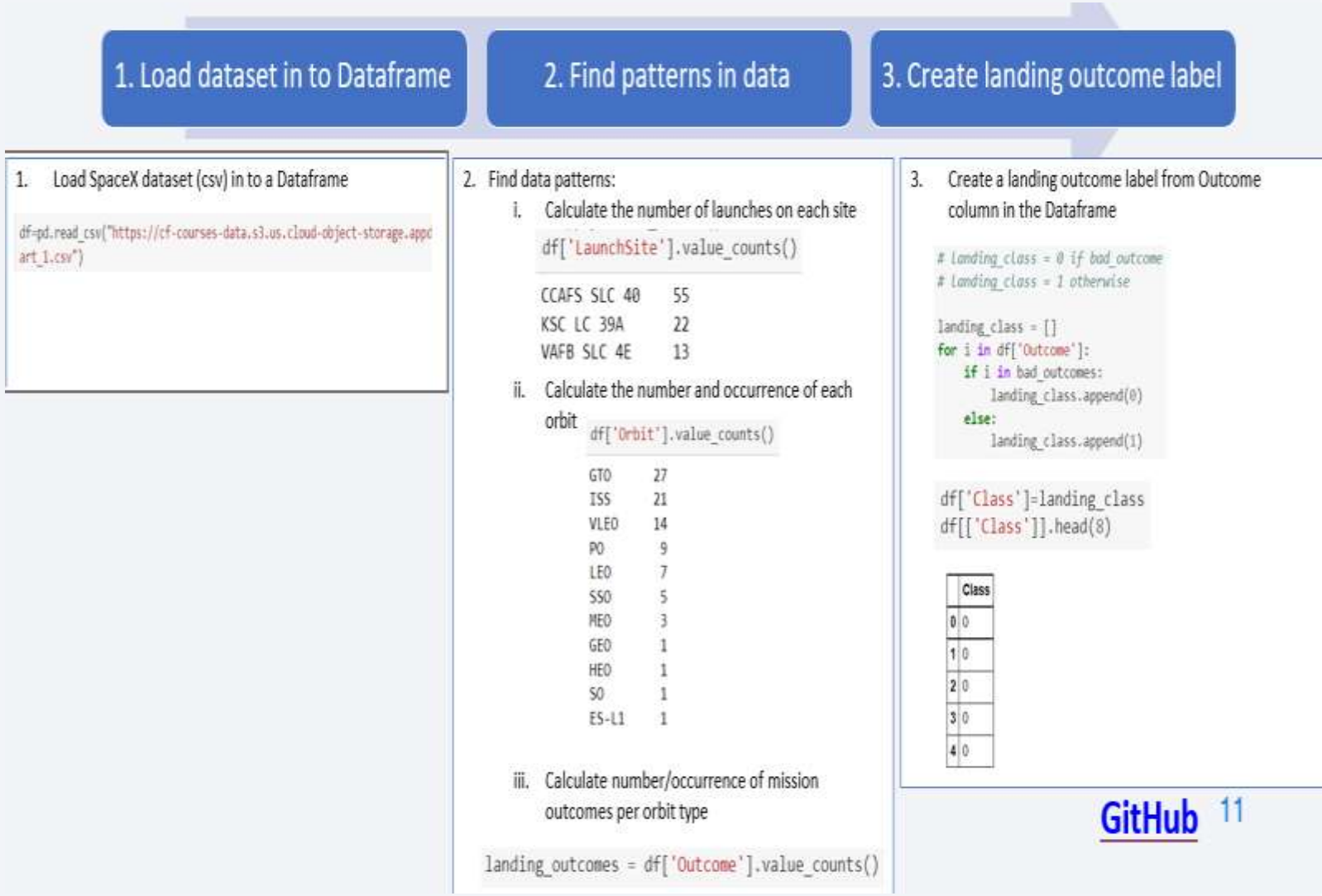
6. Convert launch_dict to Dataframe:

```
df=pd.DataFrame(launch_dict)
```

GitHub

# Data Wrangling

- Conducted Exploratory Data Analysis (EDA) to find patters in data and define
- labels for training supervised models
- The data set contained various mission outcomes that were converted into Training Labels with 1 meaning the booster successfully landed and 0 meaning booster was unsuccessful in landing.



# EDA with Data Visualization

- As part of the Exploratory Data Analysis (EDA), following charts were plotted to gain further insights into the dataset:

1. Scatter plot:
   - Shows relationship or correlation between two variables making patterns easy to observe
   - Plotted following charts to visualize:
     - Relationship between Flight Number and Launch Site
     - Relationship between Payload and Launch Site
     - Relationship between Flight Number and Orbit Type
     - Relationship between Payload and Orbit Type
2. Bar Chart:
   - Commonly used to compare the values of a variable at a given point in time. Bar charts makes it easy to see which groups are highest/common and how other groups compare against each other. Length of each bar is proportional to the value of the items that it represents
   - Plotted following Bar chart to visualize:
     - Relationship between success rate of each orbit type
3. Line Chart: Commonly used to track changes over a period of time

# EDA with SQL

- To better understand SpaceX data set, following SQL queries/operations were
- performed on an IBM DB2 cloud instance:

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string 'CCA'
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome in ground pad was achieved.
6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. List the total number of successful and failure mission outcomes
8. List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
9. List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
10. Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

# Build an Interactive Map with Folium

- Folium interactive map helps analyze geospatial data to perform more interactive visual analytics and better understand factors such location and proximity of launch sites that impact launch success rate.
- Following map object were created and added to the map:
  - Mark all launch sites on the map. This allowed to visually see the launch sites on the map.
    - Added 'folium.circle' and 'folium.marker' to highlight circle area with a text label over each launch site.
  - Added a 'MarkerCluster()' to show launch success (green) and failure (red) markers for each launch site.
  - Calculated distances between a launch site to its proximities (e.g., coastline, railroad, highway, city)
    - Added 'MousePosition() to get coordinate for a mouse position over a point on the map
    - Added 'folium.Marker()' to display distance (in KM) on the point on the map (e.g., coastline, railroad, highway, city)
    - Added 'folium.Polyline()' to draw a line between the point on the map and the launch site
    - Repeated steps above to add markers and draw lines between launch sites and proximities – coastline, railroad, highway, city)
- Building the Interactive Map with Folium helped answered following questions:
  - Are launch sites in close proximity to railways? YES
  - Are launch sites in close proximity to highways? YES
  - Are launch sites in close proximity to coastline? YES
  - Do launch sites keep certain distance away from cities? YES

# Build a Dashboard with Plotly Dash

- **Summary of Plots/Graphs and Interactions in the Dashboard**

**1.Dropdown Menu**:

1. **Purpose**: Filters data based on the selected launch site.
2. **Why Added**: Allows users to focus on specific sites and analyze their performance.

**2.Pie Chart (Mission Outcomes)**:

1. **Purpose**: Visualizes the success/failure ratio for the selected launch site.
2. **Why Added**: Quickly conveys the effectiveness of each launch site in terms of mission outcomes.

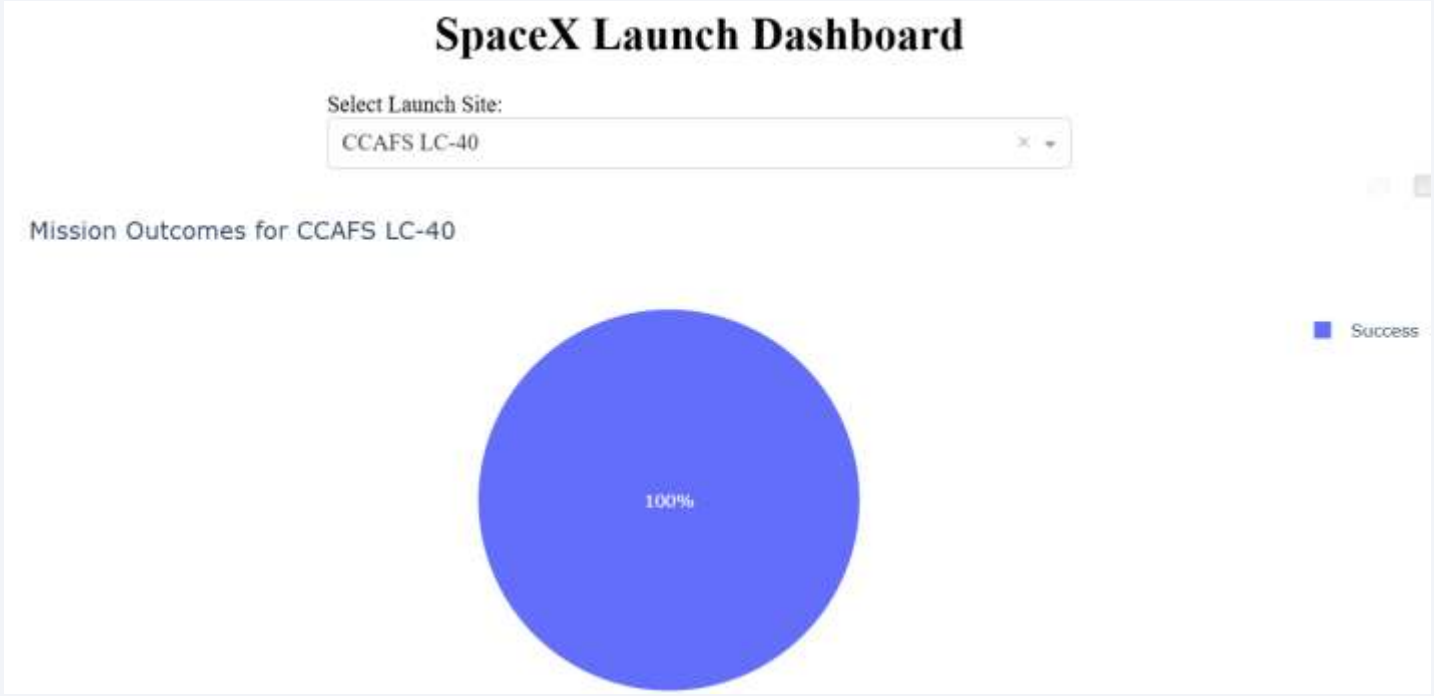
**3.Scatter Plot (Payload vs. Orbit)**:

1. **Purpose**: Shows the relationship between payload mass and orbit type, color-coded by mission outcome.
2. **Why Added**: Helps identify trends or patterns, such as which payload ranges are associated with successful outcomes for specific orbits.

# Build a Dashboard with Plotly Dash

**Reason for Adding These Features**

- **Interactive Dropdown**: Empowers users to dynamically explore data for individual launch sites.
- **Visual Insights**: The pie chart summarizes outcomes at a glance, while the scatter plot provides detailed relationships between variables.
- **Ease of Use**: Together, these plots offer both high-level summaries and granular insights, making the dashboard both informative and engaging.



# Predictive Analysis (Classification)

- **Summary of Model Development Process**
- **1. Data Preparation**
- **Key Steps**:
  - Selected features: Launch site, payload mass, orbit type, and booster version.
  - Target variable: Binary classification (1 = Success, 0 = Failure).
  - Preprocessed data: Handled missing values, standardized numerical features, and encoded categorical variables.
- **2. Model Building**
- **Models Used**:
  - Logistic Regression
  - Decision Tree
  - Support Vector Machine (SVM)
  - K-Nearest Neighbors (KNN)
- **Process**:
  - Split data into training (80%) and testing (20%) sets.
  - Trained each model using the training dataset.
- **3. Model Evaluation**
- **Metrics**:
  - Accuracy
  - Precision, Recall, F1-Score
  - Confusion Matrix
- **Outcome**:
  - Decision Tree achieved the highest training accuracy (87.5%).

**4. Model Improvement**

•**Techniques Used**:

- Hyperparameter tuning using GridSearchCV.
- Tested parameters like tree depth (Decision Tree) and kernel types (SVM).
- Optimized performance by cross-validation to ensure generalizability.

**5. Best Performing Model**

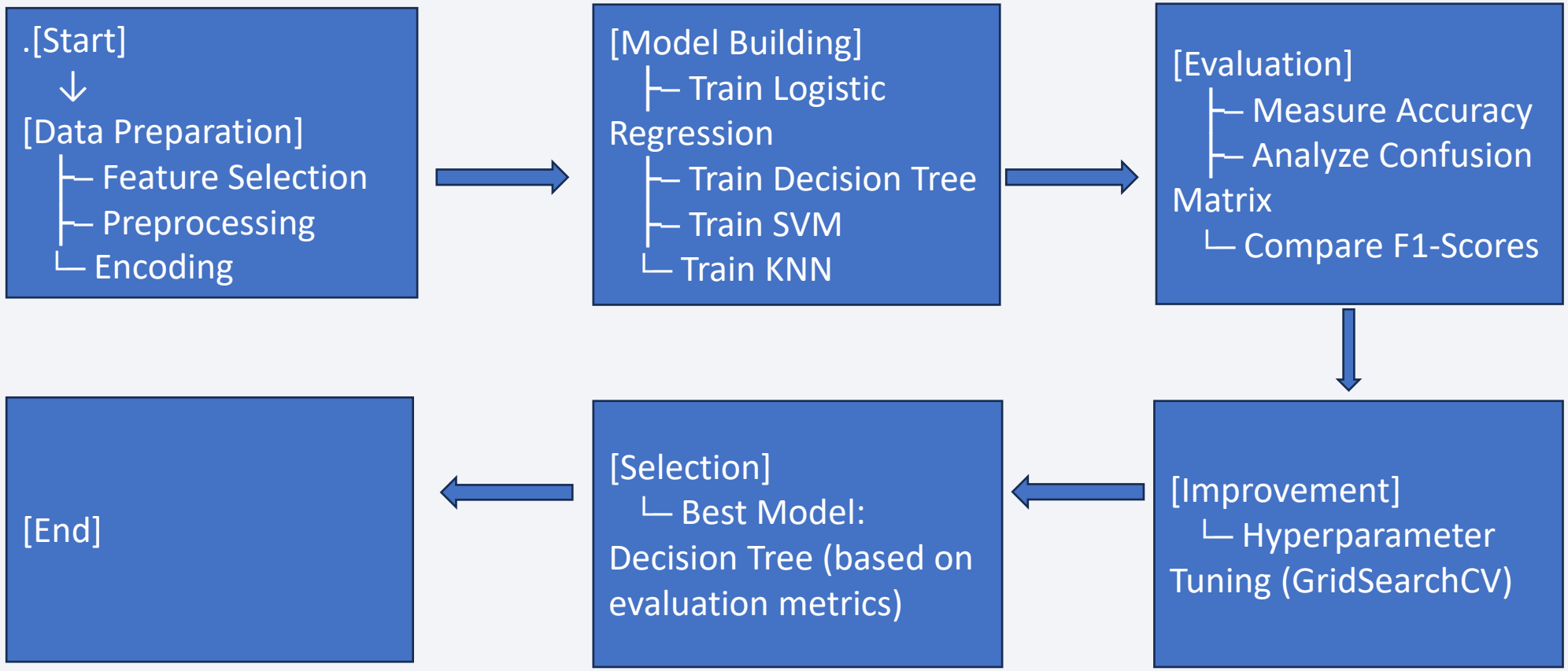
•**Final Choice**: Decision Tree

- Test accuracy: 83%
- Balanced precision and recall

•**Why Chosen**:

- High interpretability and ability to identify feature importance.

# Predictive Analysis (Classification) - Flowchart

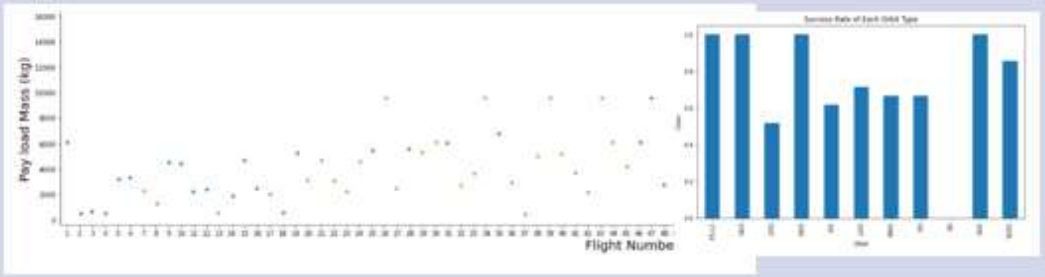


# Results

Following sections and slides explain results for:

**Exploratory data analysis results**

- Samples:




**Interactive analytics demo in screenshots**

- Samples


**Predictive analysis results**

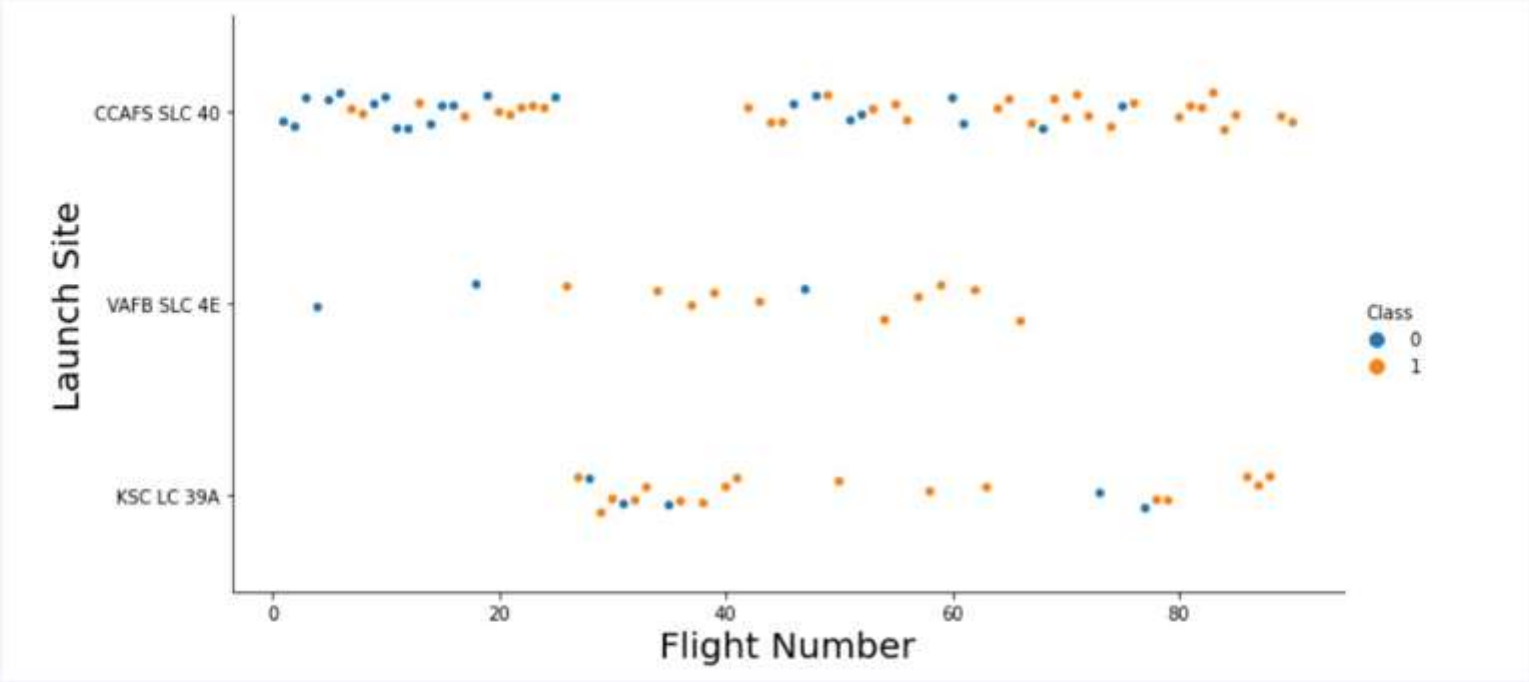
- Samples

| | Algo Type | Accuracy Score |
|---|---|---|
| 2 | Decision Tree | 0.903571 |
| 3 | KNN | 0.848214 |
| 1 | SVM | 0.848214 |
| 0 | Logistic Regression | 0.846429 |

Section 2

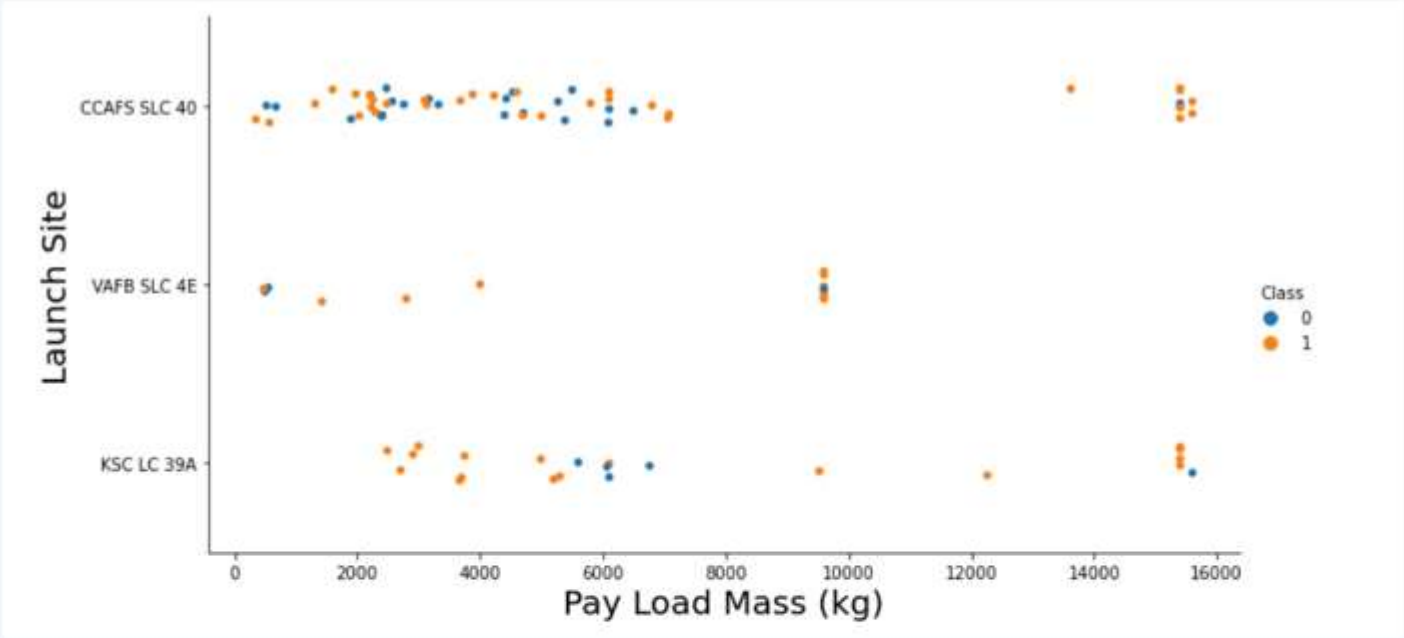
# Insights drawn from EDA

# Flight Number vs. Launch Site



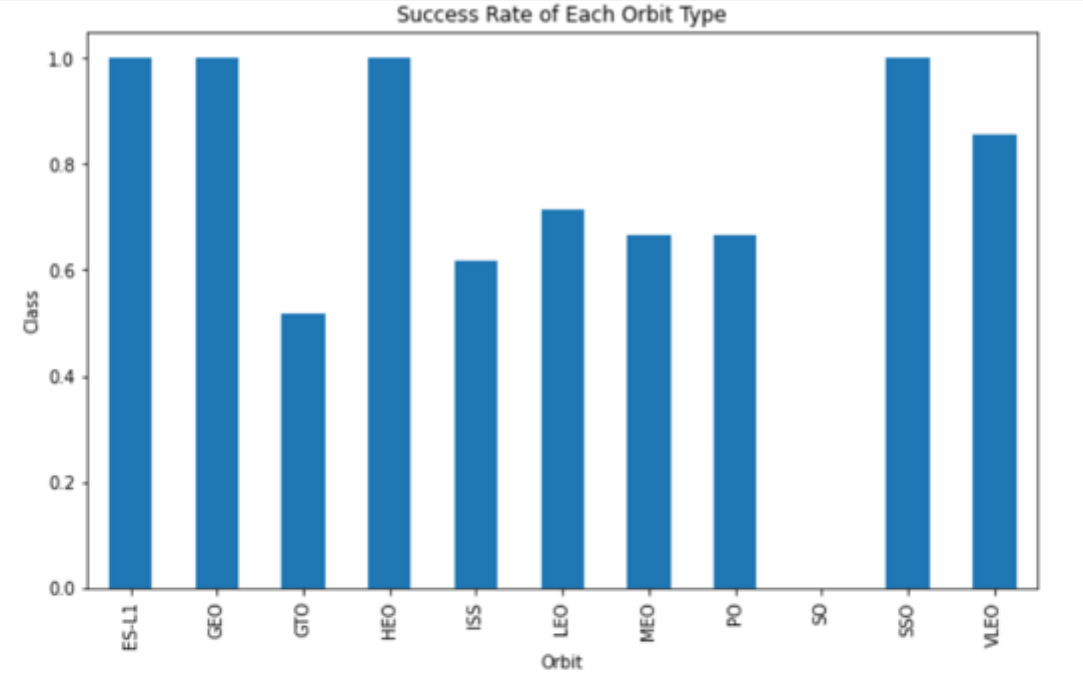

- Success rates (Class=1) increases as the number of flights increase
- For launch site 'KSC LC 39A', it takes at least around 25 launches before a first successful launch

# Payload vs. Launch Site



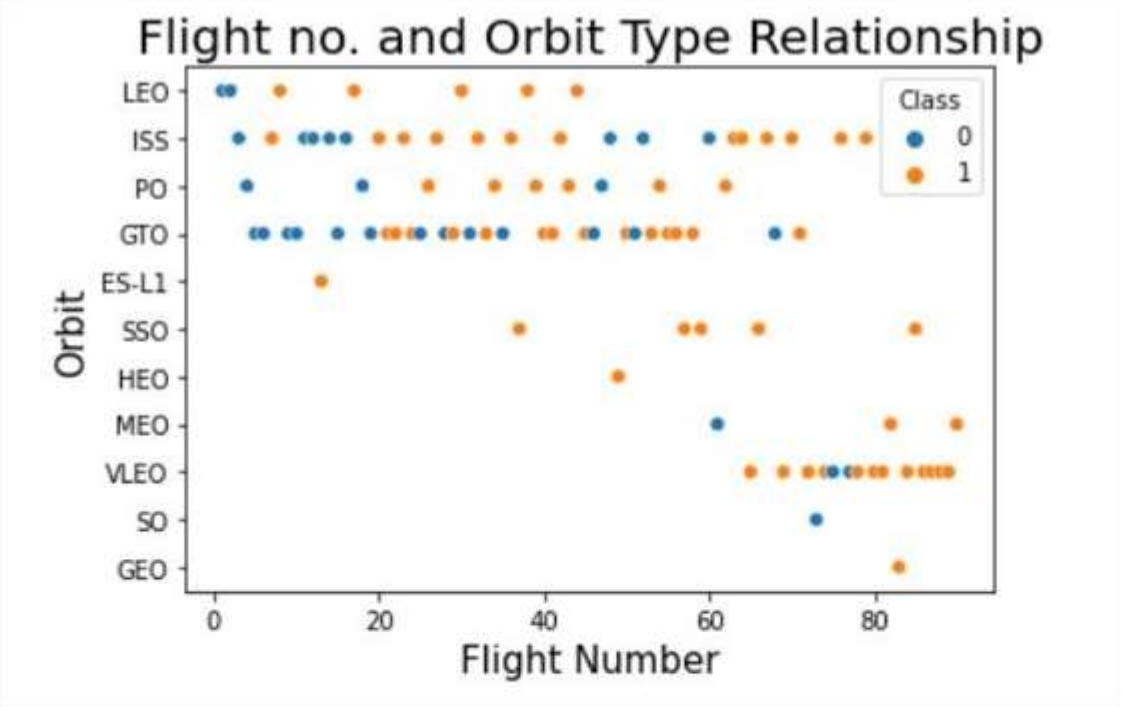

- For launch site 'VAFB SLC 4E', there are no rockets launched for payload greater than 10,000 kg
- Percentage of successful launch (Class=1) increases for launch site 'VAFB SLC 4E' as the payload mass increases
- There is no clear correlation or pattern between launch site and payload mass

# Success Rate vs. Orbit Type



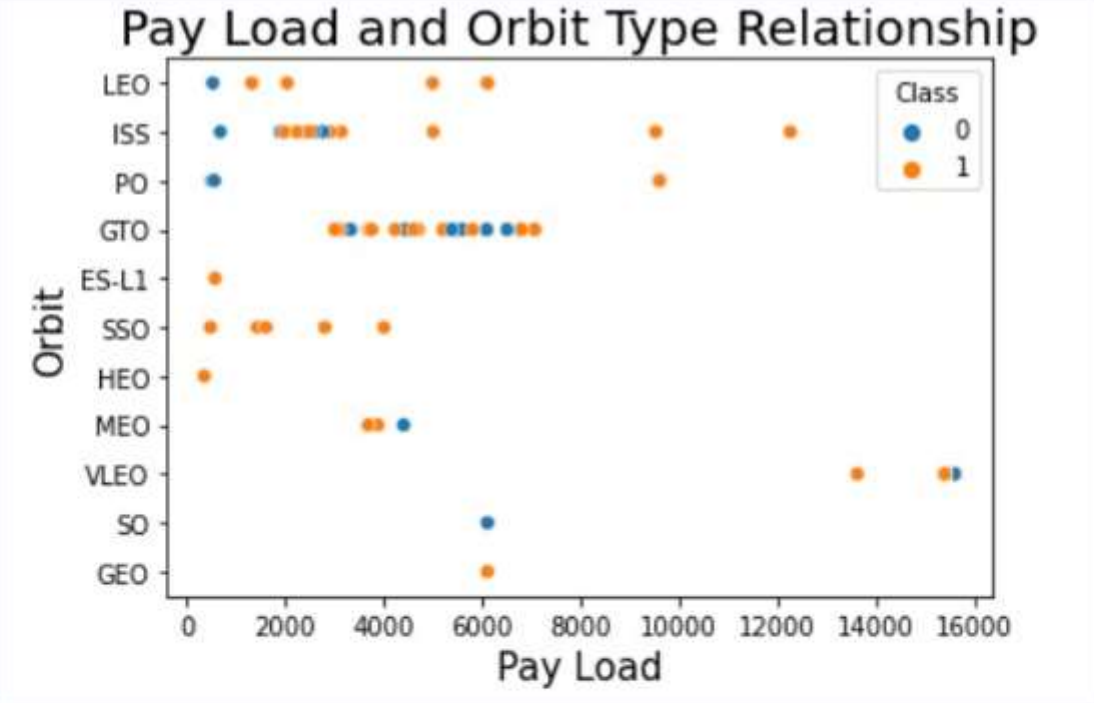

- Orbits ES-LI, GEO, HEO, and SSO have the highest success rates
- GTO orbit has the lowest success rate

# Flight Number vs. Orbit Type



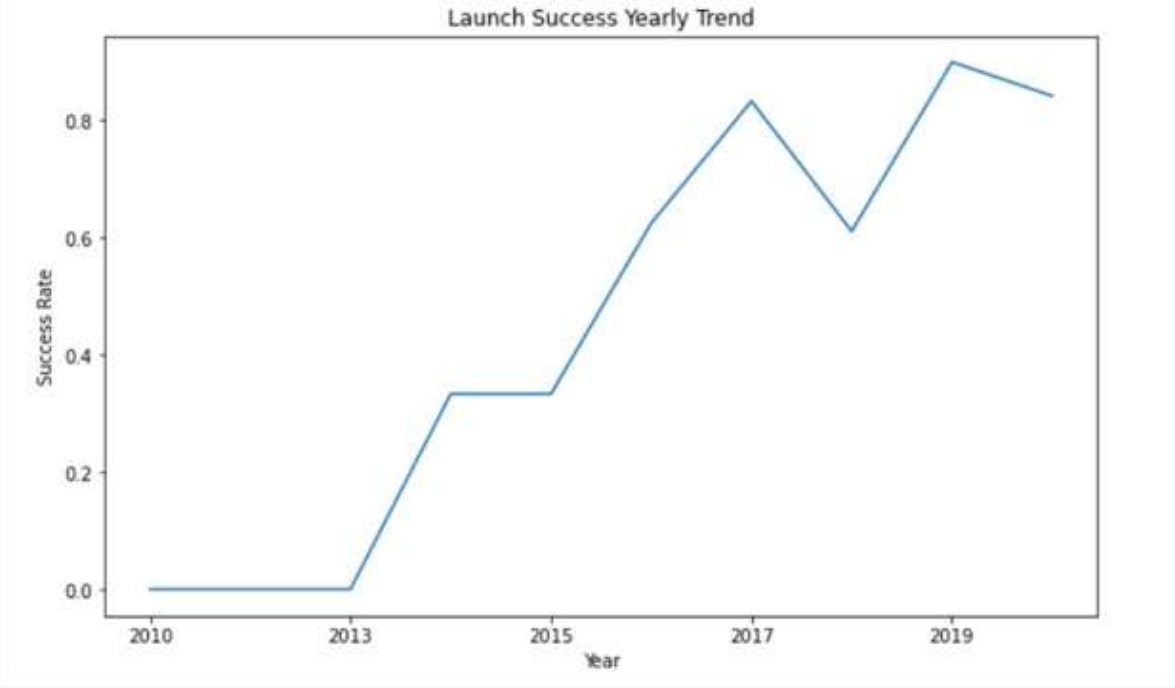

- For orbit VLEO, first successful landing (class=1) doesn't occur until 60+ number of flights
- For most orbits (LEO, ISS, PO, SSO, MEO, VLEO) successful landing rates appear to increase with flight numbers
- There is no relationship between flight number and orbit for GTO

# Payload vs. Orbit Type




- Successful landing rates (Class=1) appear to increase with pay load for orbits LEO, ISS, PO, and SSO
- For GEO orbit, there is not clear pattern between payload and orbit for successful or unsuccessful landing

# Launch Success Yearly Trend




- Success rate (Class=1) increased by about 80% between 2013 and 2020
- Success rates remained the same between 2010 and 2013 and between 2014 and 2015
- Success rates decreased between 2017 and 2018 and between 2019 and 2020

# All Launch Site Names

Here is an SQL query to find the unique launch sites for Falcon 9 and Falcon Heavy missions:

**SQL Query:**

```
SELECT DISTINCT launch_site
FROM launch_sites;
```

**Explanation:**

- This query will retrieve a list of all different launch sites without any duplicates.
- The DISTNICT KEYWORD ensures that only unique launch site names are returned from the launch_sites table.

**Example Result:**

The result will return unique launch site names like:

1. **Cape Canaveral Space Force Station (CCSFS)**
2. **Kennedy Space Center**
3. **Vandenberg Space Force Base**
4. **SpaceX Boca Chica**
5. **Wallops Flight Facility**

# Launch Site Names Begin with 'CCA'

SQL query to find 5 records where the launch site names begin with "CCA":
**SQL QUERY**

```
SELECT launch_site, location, description, result
FROM launch_sites
WHERE launch_site LIKE 'CCA%'
LIMIT 5;
```

**RESULT:**

| 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 | 7:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-08-10 | 0:35:00 | 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

SQL query to calculate the total payload carried by boosters for NASA missions:

**SQL QUERY**

```
SELECT SUM(payload_weight) AS total_payload
FROM launch_missions
WHERE organization = 'NASA';
```

**Explanation:**

- The SUM(payload_weight) function adds up the total payload carried by all missions in the launch_missions table where the organization is 'NASA'.
- The WHERE organization = 'NASA' condition filters the records to only include NASA-related missions.
- The result will be the total payload weight carried by Falcon 9 or Falcon Heavy boosters in NASA missions.

**Result:**

```
total_payload
--------------
49956 kg
```

# Average Payload Mass by F9 v1.1

**SQL Query:**

```
SELECT AVG(payload_weight) AS average_payload_mass
FROM launch_missions
WHERE booster_version = 'F9 v1.1';
```

**Explanation:**

- The AVG(payload_weight) function calculates the average payload weight for all missions using the booster version "F9 v1.1."
- The WHERE booster_version = 'F9 v1.1' condition filters the records to include only missions where the booster version is "F9 v1.1."
- The result will be the average payload mass carried by the F9 v1.1 booster.

**Result:**

If the calculated average payload mass for F9 v1.1 was found, the result might look like this:

```
average_payload_mass
--------------------
2928 kg
```

# First Successful Ground Landing Date

- **SQL Query:**

```
SELECT MIN(landing_date) AS first_successful_landing_date
FROM launch_missions
WHERE landing_outcome = 'Success’
AND landing_site = 'Ground Pad';
```

- **Explanation:**
  - MIN(landing_date) retrieves the earliest (first) successful landing date.
  - The WHERE landing_outcome = 'Success' condition filters for successful landings.
  - The AND landing_site = 'Ground Pad' condition ensures only landings on a ground pad are considered.
  - This query will return the date of the first successful landing outcome on a ground pad.
- **Example Result:**

```
first_successful_landing_date
---------------------------
2015-12-22
```

# Successful Drone Ship Landing with Payload between 4000 and 6000

**SQL Query:**

```
SELECT booster_version
FROM launch_missions
WHERE landing_outcome = 'Success'
AND landing_site = 'Drone Ship'
AND payload_weight > 4000
AND payload_weight < 6000;
```

**Explanation:**

- landing_outcome = 'Success' filters missions where the landing was successful.
- landing_site = 'Drone Ship' ensures the landing occurred on a drone ship.
- payload_weight > 4000 AND payload_weight < 6000 restricts the query to missions with payload masses between 4000 kg and 6000 kg.
- The SELECT booster_name retrieves the names of the boosters that meet these criteria.

**Result:**

```
booster_version
------------
F9 FT B1022
F9 FT B1026
```

# Total Number of Successful and Failure Mission Outcomes

- **SQL Query:**

```
SELECT landing_outcome, COUNT(*) AS total_missions
FROM launch_missions
GROUP BY landing_outcome;
```

- **Explanation:**
- landing_outcome groups the missions by their landing outcomes, such as "Success" or "Failure."
- COUNT(*) counts the number of missions for each landing outcome.
- GROUP BY landing_outcome ensures the results are grouped by the outcome type.
- **Result:**

```
landing_outcome | total_missions
----------------|----------------
Success         | 99
Failure         | 1
```

# Boosters Carried Maximum Payload

**SQL Query:**

```
SELECT booster_version, MAX(payload_weight) AS max_payload_mass
FROM launch_missions
GROUP BY booster_name
ORDER BY max_payload_mass DESC
LIMIT 1;
```

**Explanation:**

- MAX(payload_weight) calculates the maximum payload weight carried by each booster.
- GROUP BY booster_name groups the records by each booster name.
- ORDER BY max_payload_mass DESC sorts the boosters in descending order based on the maximum payload mass.
- LIMIT 1 ensures that only the booster with the maximum payload is returned.

**Result:**

```
booster_version      | max_payload_mass
-----------------|------------------
F9 B5 B1048.4      | 15600 kg
F9 B5 B1094.4      | 15000 kg
```

# 2015 Launch Records

- **SQL Query:**

```
SELECT landing_outcome, booster_version, launch_site
FROM launch_missions
WHERE landing_outcome = 'Failure'
AND landing_site = 'Drone Ship'
AND YEAR(launch_date) = 2015;
```

- **Explanation:**
  - landing_outcome = 'Failure' filters for failed landing outcomes.
  - landing_site = 'Drone Ship' ensures the landing occurred on a drone ship.
  - YEAR(launch_date) = 2015 restricts the records to missions that took place in 2015.
  - The SELECT clause retrieves the landing_outcome, booster_version, and launch_site for each matching record.
- **Result:**

```
landing_outcome | booster_version    | launch_site
----------------|--------------------|-------------
Failure         | Falcon 9 v1.1      | CCAFS LC-40
Failure         | Falcon 9 v1.2 Block 3 |CCAFS LC-40
```

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

- **SQL Query:**

```
SELECT landing_outcome, COUNT(*) AS outcome_count
FROM launch_missions
WHERE launch_date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY landing_outcome
ORDER BY outcome_count DESC;
```

- **Explanation:**
  - launch_date BETWEEN '2010-06-04' AND '2017-03-20' filters the records to include only those within the specified date range.
  - COUNT(*) AS outcome_count counts the number of missions for each landing outcome.
  - GROUP BY landing_outcome groups the results by landing outcome (e.g., "Failure" or "Success").
  - ORDER BY outcome_count DESC sorts the outcomes in descending order based on the count.
- **Result:**

```
landing_outcome | outcome_count
----------------|--------------
Success         | 5
Failure         | 5
```

Section 3

# Launch Sites Proximities Analysis

# <Folium Map Screenshot 1>

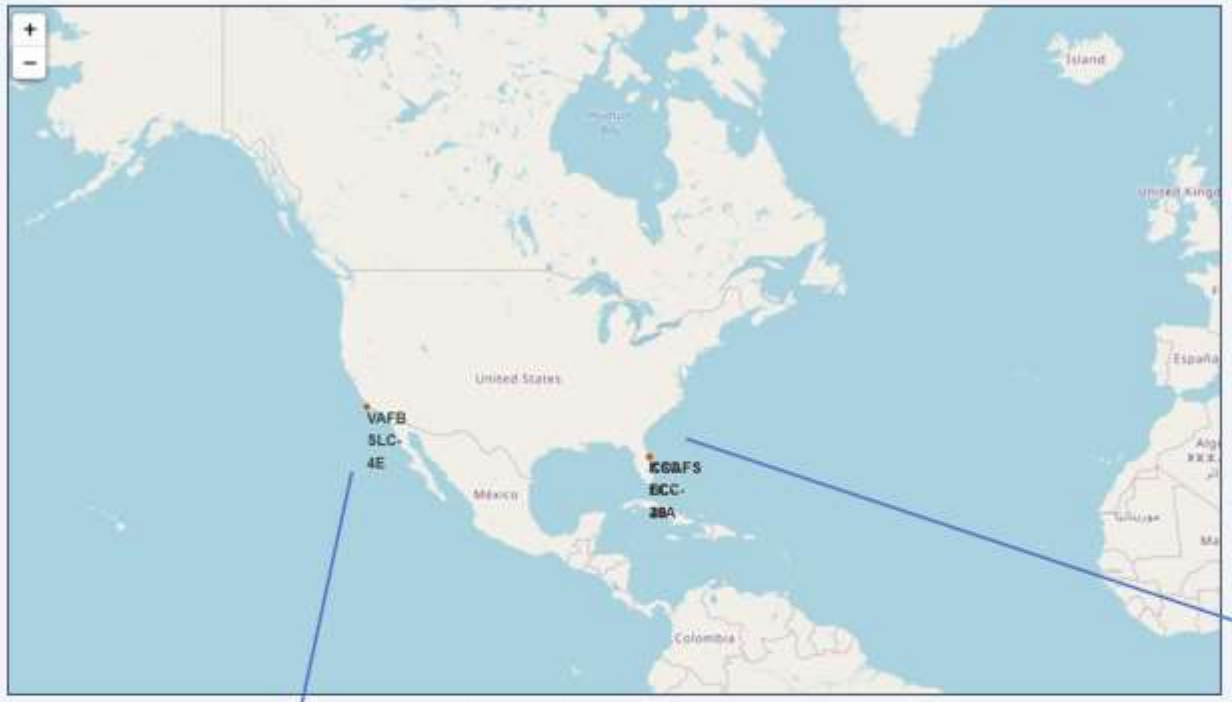



Fig 1 – Global Map


Fig 2 – Zoom 1


Fig 3 – Zoom 2

Figure 1 on left displays the Global map with Falcon 9 launch sites that are located in the United States (in California and Florida). Each launch site contains a circle, label, and a popup to highlight the location and the name of the launch site. It is also evident that all launch sites are near the coast.

Figure 2 and Figure 3 zoom in to the launch sites to display 4 launch sites:

- VAFB SLC-4E (CA)
- CCAFS LC-40 (FL)
- KSC LC-39A (FL)
- CCAFS SLC-40 (FL)

# <Folium Map Screenshot 2>


Fig 1 – US map with all Launch Sites

- Figure 1 is the US map with all the Launch Sites. The numbers on each site depict the total number of successful and failed launches
- Figure 2, 3, 4, and 5 zoom in to each site and displays the success/fail markers with green as success and red as failed
- By looking at each site map, KSC LC-39A Launch Site has the greatest number of successful launches

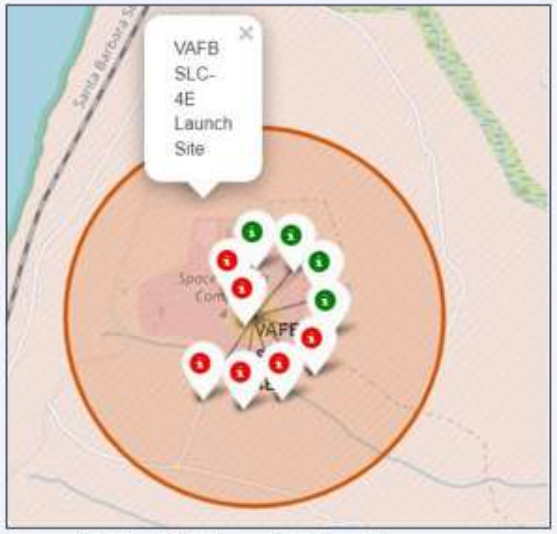



Fig 2 – VAFB Launch Site with success/failed markers

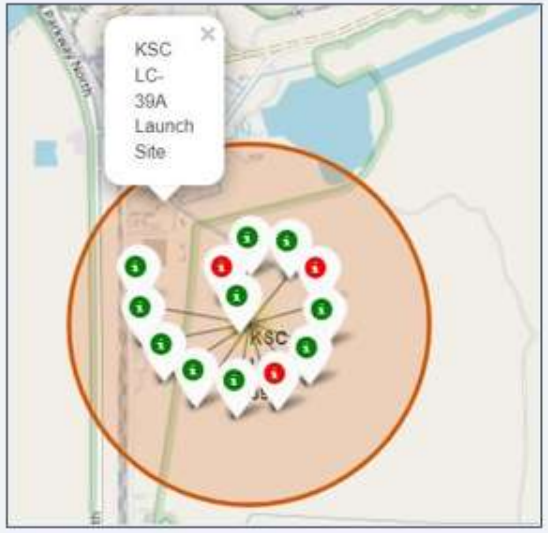



Fig 3 – KSC LC-39A success/failed markers

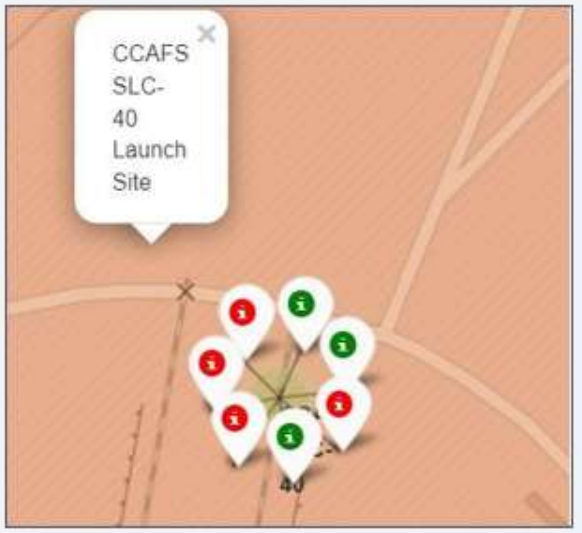



Fig 4 – CCAFS SLC-40 success/failed markers

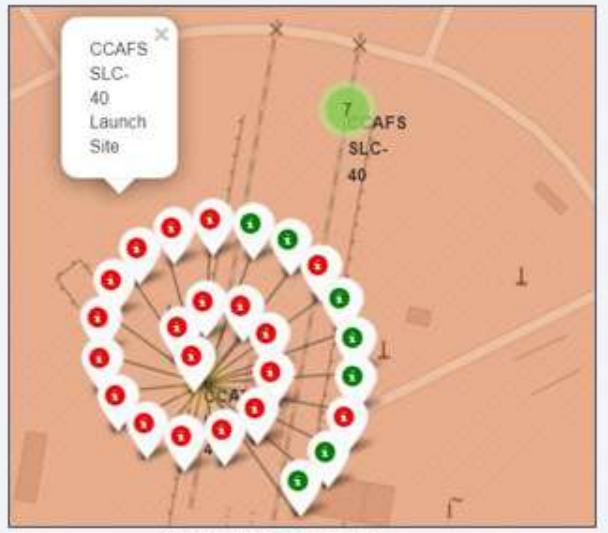



Fig 5 – CCAFS SLC-40 success/failed markers

# <Folium Map Screenshot 3>


Fig 1 – Proximity site map for VAFB SLC-4E

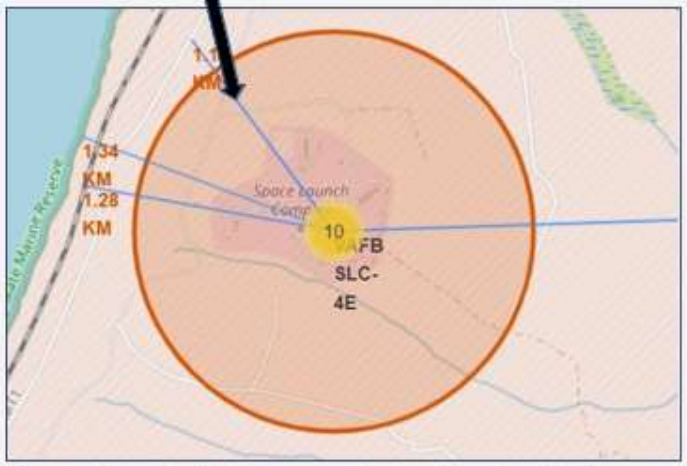
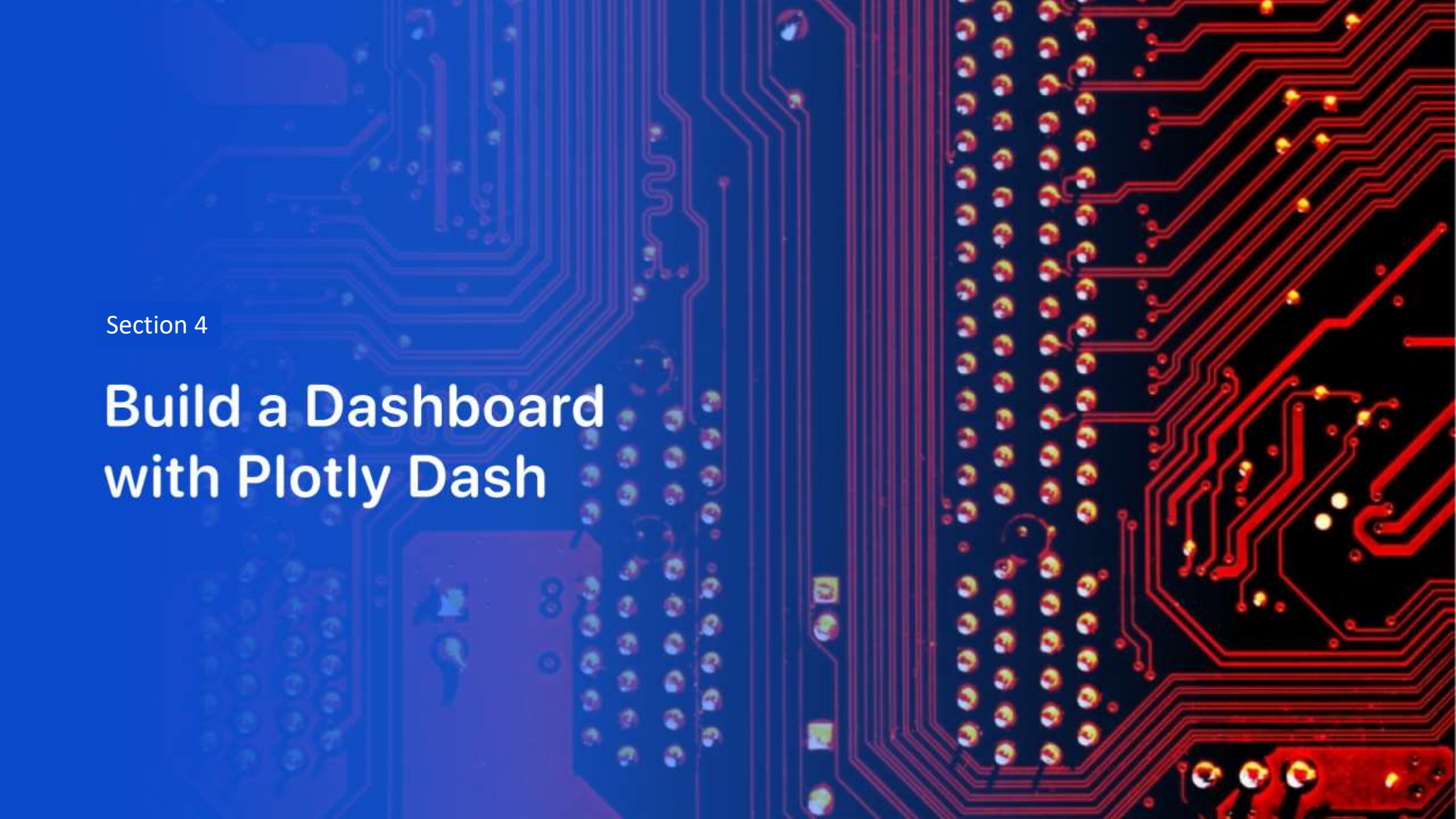



Fig 2 – Zoom in for sites – coastline, railroad, and highway

Figure 1 displays all the proximity sites marked on the map for Launch Site VAFB SLC-4E. City Lompoc is located further away from Launch Site compared to other proximities such as coastline, railroad, highway, etc. The map also displays a marker with city distance from the Launch Site (14.09 km)

Figure 2 provides a zoom in view into other proximities such as coastline, railroad, and highway with respective distances from the Launch Site

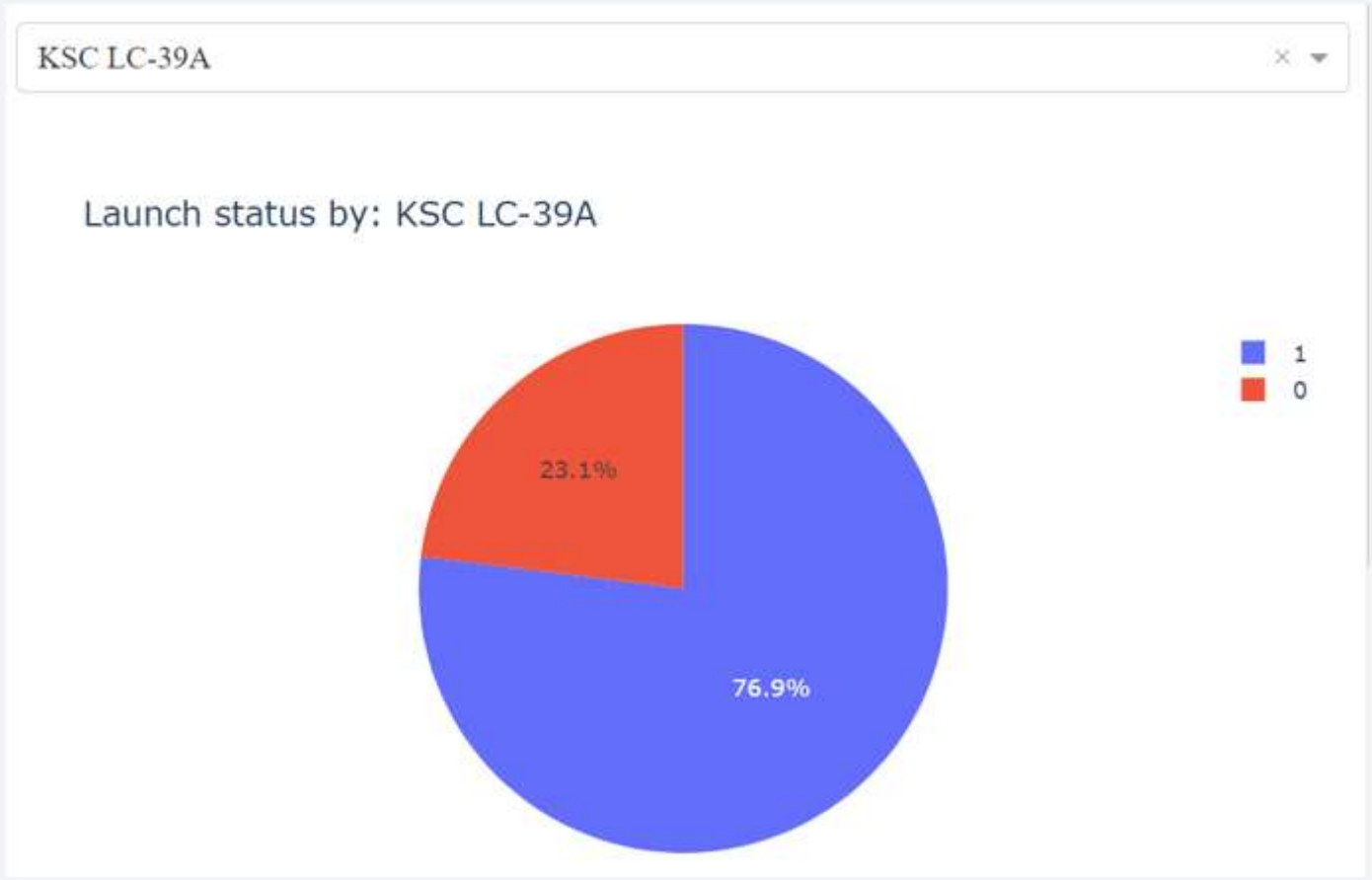
In general, cities are located away from the Launch Sites to minimize impacts of any accidental impacts to the general public and infrastructure. Launch Sites are strategically located near the coastline, railroad, and highways to provide easy access to resources.


Section 4
Build a Dashboard with Plotly Dash

# <Dashboard Screenshot 1>

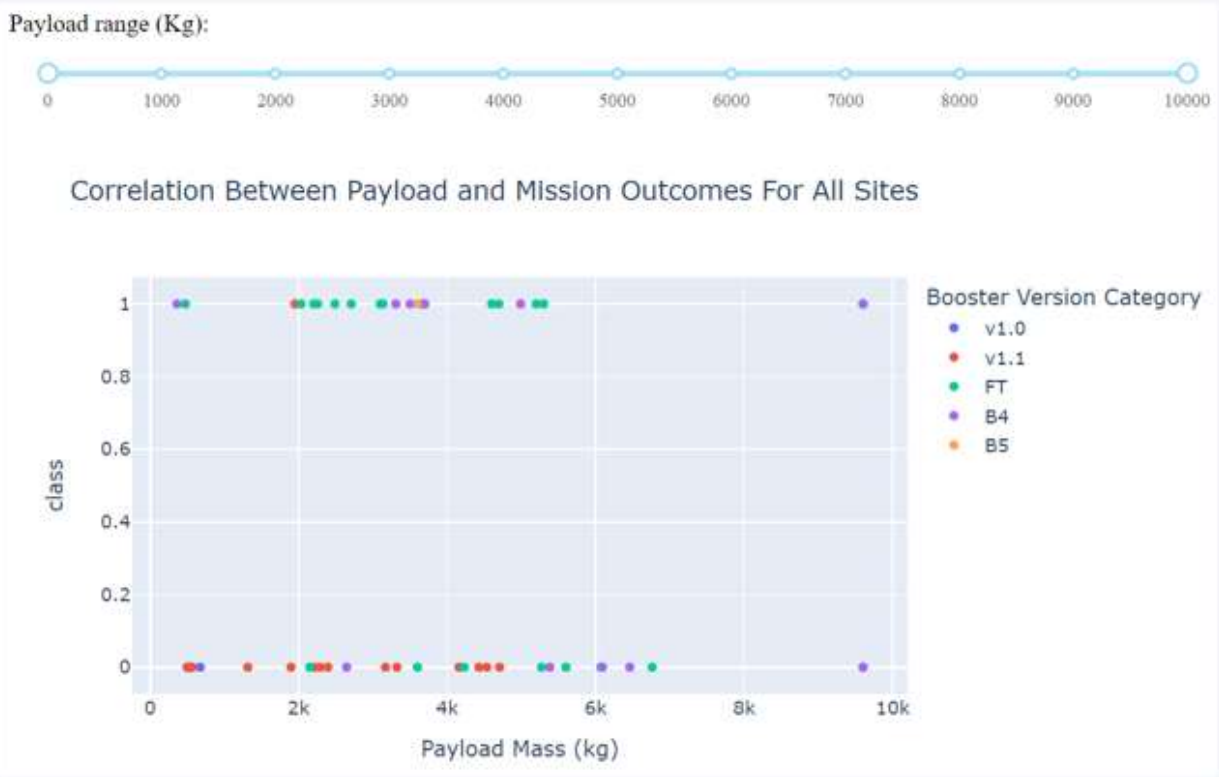

- Launch Site 'KSC LC-39A' has the highest launch success rate
- Launch Site 'CCAFS SLC-40' has the lowest launch success rate

# <Dashboard Screenshot 2>




- KSC LC-39A Launch Site has the highest launch success rate and count
- Launch success rate is 76.9%
- Launch success failure rate is 23.1%

# <Dashboard Screenshot 3>



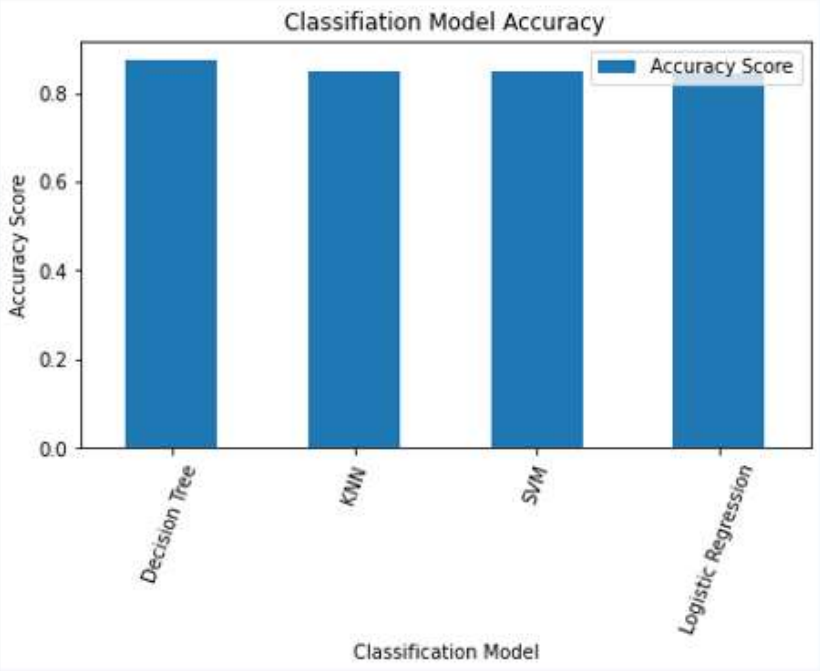


- Most successful launches are in the payload range from 2000 to about 5500
- Booster version category ‘FT’ has the most successful launches

Section 5

# Predictive Analysis (Classification)

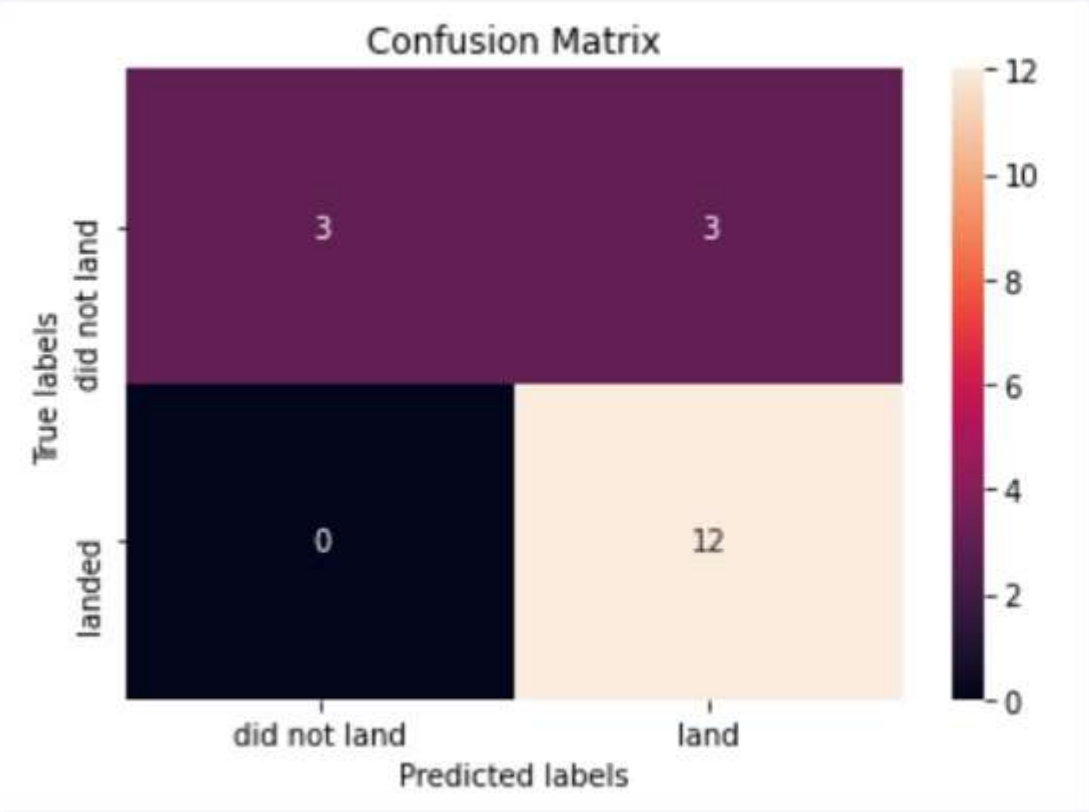
# Classification Accuracy




| | Algo Type | Accuracy Score | Test Data Accuracy Score |
|---|---|---|---|
| 2 | Decision Tree | 0.875000 | 0.833333 |
| 3 | KNN | 0.848214 | 0.833333 |
| 1 | SVM | 0.848214 | 0.833333 |
| 0 | Logistic Regression | 0.846429 | 0.833333 |

- Based on the Accuracy scores and as also evident from the bar chart, Decision Tree algorithm has the highest classification score with a value of .8750
- Accuracy Score on the test data is the same for all the classification algorithms based on the data set with a value of .8333
- Given that the Accuracy scores for Classfication algorithms are very close and the test scores are the same, we may need a broader data set to further tune the models

# Confusion Matrix




- The confusion matrix is same for all the models (LR, SVM, Decision Tree, KNN)
- Per the confusion matrix, the classifier made 18 predictions
- 12 scenarios were predicted Yes for landing, and they did land successfully (True positive)
- 3 scenarios (top left) were predicted No for landing, and they did not land (True negative)
- 3 scenarios (top right) were predicted Yes for landing, but they did not land successfully (False positive)
- Overall, the classifier is correct about 83% of the time ((TP + TN) / Total) with a misclassification or error rate ((FP + FN) / Total) of about 16.5%

# Conclusions

- **Conclusions:**

1. **Landing Outcome Successes Outnumber Failures**
   In the specified period, successful landings (whether on drone ships or ground pads) greatly outnumbered failed landings, indicating a trend of improving reliability over time in SpaceX's landing operations.

2. **Drone Ship Landings Are Commonly Associated with Success**
   The majority of successful landings occurred on drone ships, demonstrating SpaceX’s success with remote, ocean-based landing sites.

3. **Ground Pad Landings Also Show High Success Rate**
   Ground pad landings, while less frequent compared to drone ship landings, have also been successful, adding to the consistency and reliability of SpaceX’s landing technology.

4. **Continuous Improvement in Landing Technologies**
   The overall improvement in landing success rates, especially as the number of missions increases, suggests SpaceX has significantly refined its booster recovery technologies between 2010 and 2017.

# 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!