IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Tim Hoogenboom
May 6, 2025

# Table of Contents

1. Executive Summary
2. Introduction
3. Methodology
4. Results
5. Conclusion
6. Appendix

# Executive Summary

## Summary of methodologies

- *Data Collection* via SpaceX API and web scraping of Wiki
- *Data Wrangling* to add label launch success or launch failure
- *Data Analysis* to query dataset and better understand content
- *Data Exploration* to graphically visualize relation between variables launch site, payload mass, orbit type, years, and success rate, and spatially visualize relation between launch site, success, and proximity to objects
- *Data dashboarding* to visualize relation between launch site, payload mass, and success rate
- *Predictive modelling* via LR, SVM, decision tree, and KNN to understand scoring accuracy

## Summary of results

- *Data Exploration* Success rate increases by each year launching independent of launching site,
- *Predictive Modelling* Decision Tree Model has the highest accuracy 87% accuracy (compared to 83% of the other models) of successful landing of the first stage rocket booster.
- Herewith competitor Space Y can make more informed bids.

# Introduction

## Background

SpaceX 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 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.

Therefor we want to understands better if we can predict if a SpaceX rocket will land successfully after the first stage and what factors affect it.

## Problem Statement

The following questions will be further investigated

1. Understand what variables affect successful launch outcome
2. Identify ML model scores highest accuracy on predicting new launch outcome (landing/not landing first stage)

Section 1

# Methodology

# Methodology

## Executive Summary

- *Data Collection* via SpaceX API (data1, Github) and web scraping of Wiki (data2, Github)
- *Data Wrangling* to reformat launch sites, orbit types and add label launch success or launch failure via IBM dataset (data3, Github)
- *Data Analysis* to query dataset and better understand content via IBM dataset (data4, Github)
- *Data Exploration* to graphically visualize relation between variables launch site, payload mass, orbit type, years, and success rate via IBM dataset (data5, Github), and spatially visualize relation between launch site, success, and proximity to objects via IBM dataset (data6, Github)
- *Data dashboarding* to visualize relation between launch site, payload mass, and success rate via IBM dataset (data7, python, Github)
- *Predictive modelling* via LR, SVM, decision tree, and KNN to understand scoring accuracy via IBM dataset (data5, Github)

# Data Collection

## What

- *Data Collection* via SpaceX API (data1, Github)

## How

- Request data via API via GET request
- Turn json response into dataframe
- Extract subset of columns into new dataframe
- Remove ‘Falcon 1’ launches from dataframe
- Wrangle Payload data by replacing missing values with mean values

### Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.`

[52]:
```
# Calculate the mean value of PayloadMass column
PayloadMassMean = data_launch_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_launch_falcon9['PayloadMass'].replace(np.nan, PayloadMassMean, inplace=True)

data_launch_falcon9.isnull().sum()
```

# Web Scraping

## What

- *Web scraping* of Wiki (data2, Github)

## How

- Request Falcon9 Launch HTML page as an HTTP response via HTTP get
- Create Beautiful soup object
- Extract HTML headers as column names
- Create dataframe based on column names and HTML table

To simplify the parsing process, we have provided an incomplete code snippet below to help you to fill up the `launch_dict`. Ple
write your own logic to parse all launch tables:

[28]:
```
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()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
```

# Web Scraping

## What

- *Web scraping* of Wiki (data2, Github)

## How

- Request Falcon9 Launch HTML page as an HTTP response via HTTP get
- Create Beautiful soup object
- Extract HTML headers as column names
- Create dataframe based on column names and HTML table

To simplify the parsing process, we have provided an incomplete code snippet below to help you to fill up the `launch_dict`. Ple
write your own logic to parse all launch tables:

[28]:
```
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()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
```

# Data Wrangling

## What

- *Data Wrangling* to reformat launch sites, orbit types and add label launch success or launch failure via IBM dataset (data3, Github)

## How

- Fetch external data set in dataframe
- Count number of launches per Launch_Site
- Count number of Orbits
- Count number of Outcomes and label into Successful [1] or Bad [0] in dataframe
- Combine dataframes

**TASK 4: Create a landing outcome label from Outcome column**

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the s

[12]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

# Data Visualization

## What

- *Data Exploration* to graphically visualize relation between variables launch site, payload mass, orbit type, years, and success rate via IBM dataset (data5, Github)

## How

- Fetch external data set in dataframe
- Count number of launches per Launch_Site
- Count number of Orbits
- Count number of Outcomes and label into Successful [1] or Bad [0] in dataframe
- Combine dataframes

Section 2

# Insights drawn from EDA

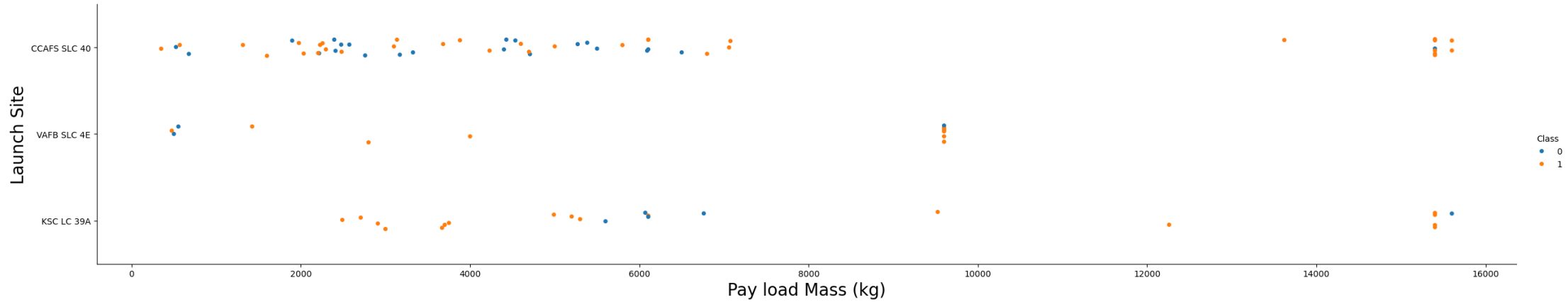
# Data Visualization

## Payload Mass vs. Launch Site

### What

- There appears to be no correlation between Payload Mass and Launch Site.

### Illustration



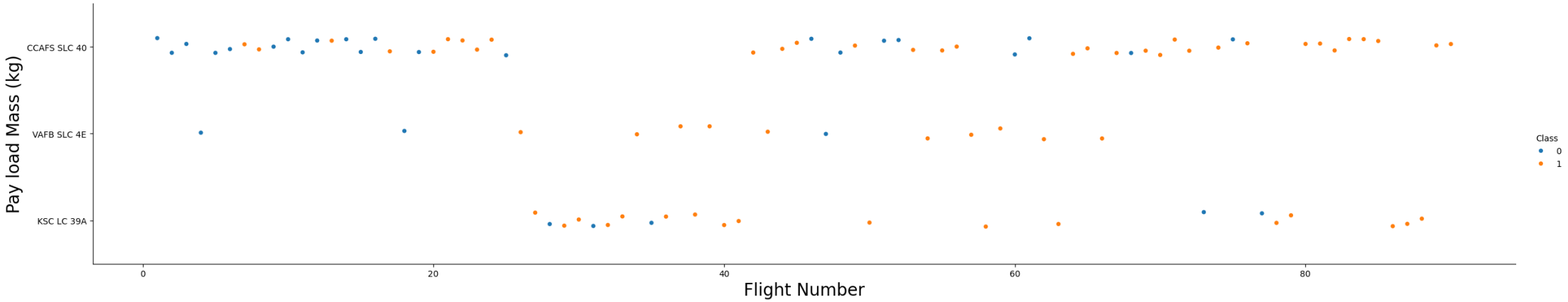
# Data Visualization

## Flight Number vs. Launch Site

### What

- As the flight number increases, the first stage is more likely to land successfully independent of the Launch_site.

### Illustration



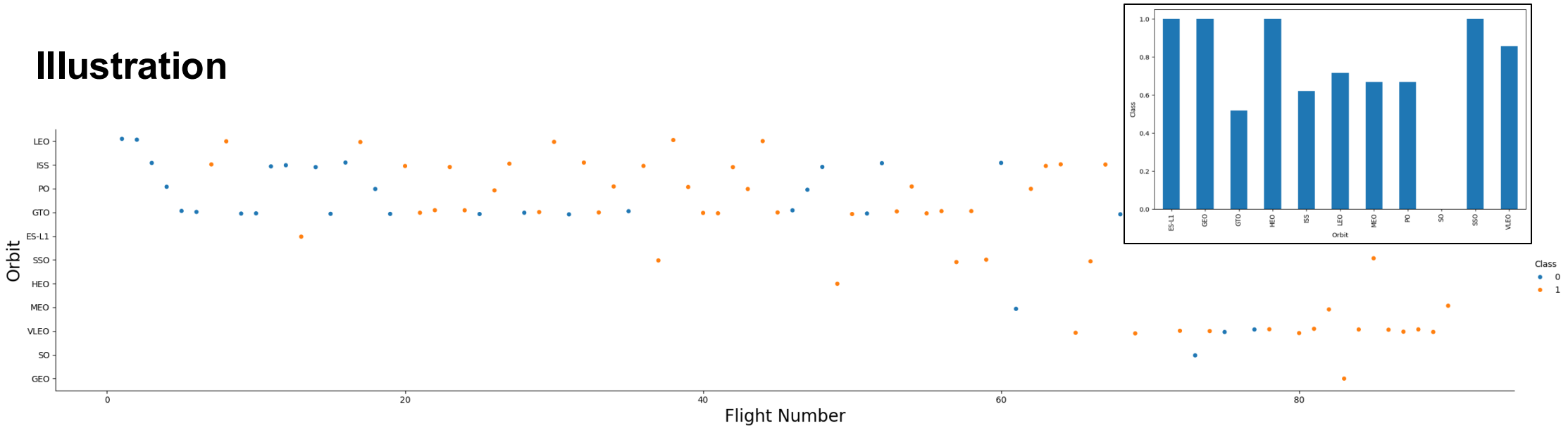
# Data Visualization

## Success Rate vs. Orbit Type (bar chart) and Flight Number vs. Orbit Type

### What

- Orbits ES-L1, GEO, HEO, SSO, VLEO have the highest average success in Launch_Outcome based on a the Bar Chart, but for orbit GEO, HEO, ES-L1 only <u>one</u> flight was executed so less statically relevant.

### Illustration



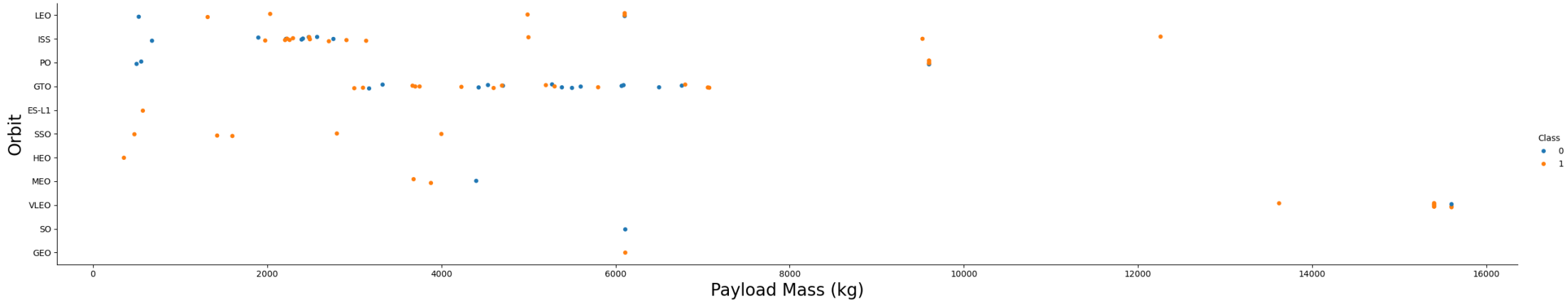
# Data Visualization

## Payload vs. Orbit Type

### What

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

### Illustration



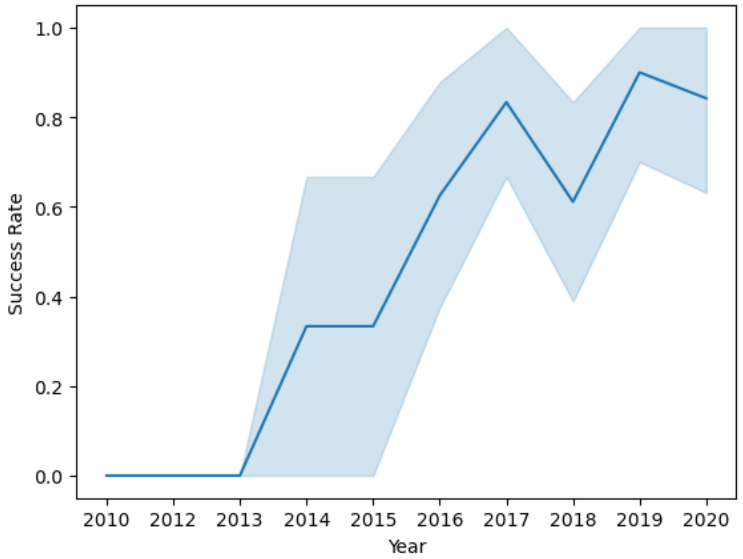
# Data Visualization

## Launch Success Yearly Trend

### What

- General conclusion is that the success rate of Launch_Outcome appears to increase over the years that flights are executed.

### Illustration



Section 3

# Launch Sites Proximities Analysis

# Data Exploration

## What

- *Data Exploration* to spatially visualize relation between launch site, success, and proximity to objects via IBM dataset (data6, Github)

## How

- Fetch external data set in dataframe
- Extract Launch_Site coordinates and create Map Objected (marker)
- Mark Launch_Site on Folium geospacial map
- Add Success [1] and Failed [1] missions per Launch_Site
- Calculate distance and visualize PolyLine between Launch_Site and selected markers in the vicinity
- Hereafter one can coclude that Launch Sites are close to railways, highways and coastline and away from cities

# Data Exploration

## What

- Explore the generated folium map and make a proper screenshot to include all launch sites' location markers on a global map

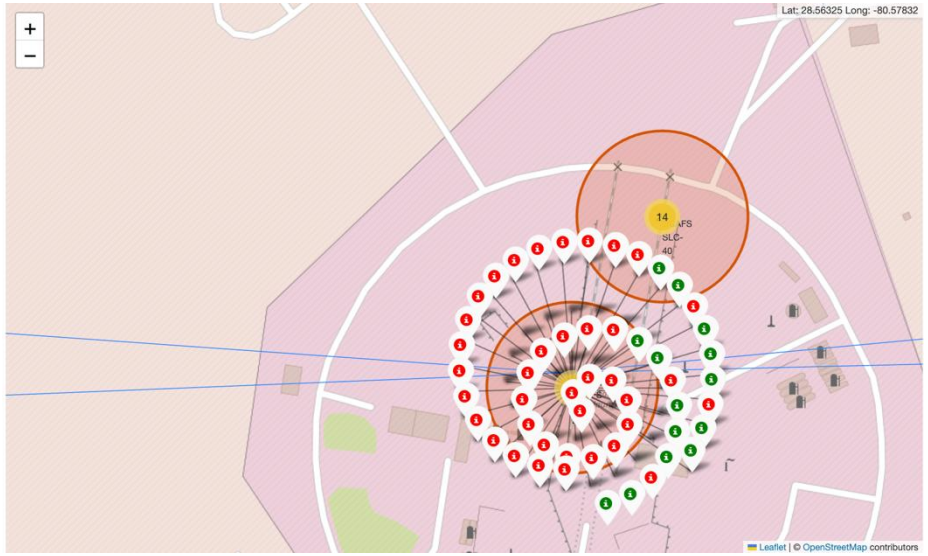
## Illustration

# Data Exploration

## What

- Visualization per Launch_Site based on all launch records if a launch was successful [1, green marker] or failure [0, red marker]

## Illustration

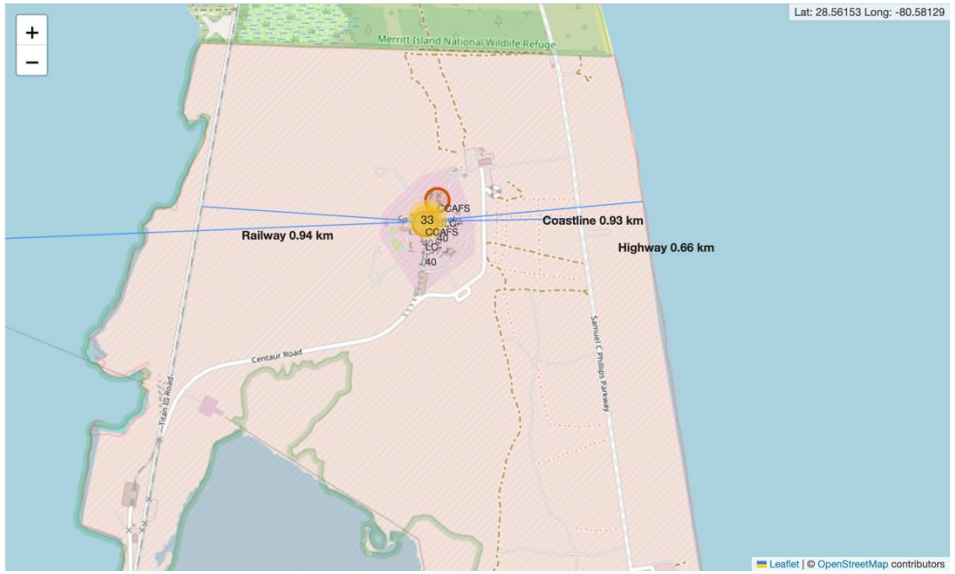



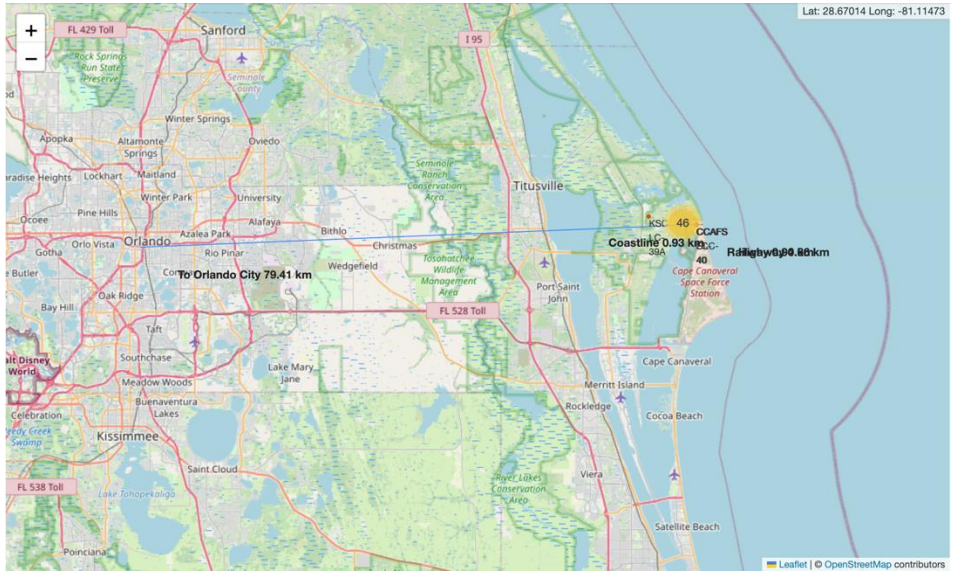
# Data Exploration

## What

- General conclusion is Launch_Sites are always in close proximity to any railway, highway, coastline, etc.
- General conclusion is Launch_Sites are always in remote areas away from cities and close to equator for favorable most weather conditions sun/clear skies.

## Illustration



# Data Analysis with SQL

## What

- *Data Analysis* to query and undertstand dataset and better understand content via IBM dataset (data4, Github)

## How

- Load external data set into SQL DB and query via MagicSQL.
- Not working for me so credits fly out to Raegan Tysh for sharing a workaround!!!
- Execute SQL queries to understand DISTINCT Launch_Site, AVERAGE payload_Mass, Boosters within certain limits, etc.
- Execute SQL queries to understand COUNT of successful Mission_Outcome and RANK successful Mission_Outcomes, etc.

# Data Analysis with SQL

## All Launch Site Names

### What

- Find the names of the unique launch sites
- Present your query result with a short explanation here

### Illustration

```
# Task 1: Display the names of the unique launch sites
query = 'SELECT DISTINCT "Launch_Site" FROM SPACEXTBL'
cur.execute(query)

# Fetch the results
names = cur.fetchall()

# Print the results
print("Unique Launch Sites:")
for site in names:
    print(site)
```

```
Unique Launch Sites:
('CCAFS LC-40',)
('VAFB SLC-4E',)
('KSC LC-39A',)
('CCAFS SLC-40',)
```

# Data Analysis with SQL

## Launch Site Names Begin with 'KSC'

### What

- Find 5 records where launch sites' names start with `KSC`
- Present your query result with a short explanation here

### Illustration

```
# Display 5 records where launch sites begin with the string 'KSC'
query = 'SELECT * FROM SPACEXTBL WHERE "Launch_Site" LIKE "KSC%" LIMIT 5'
cur.execute(query)

# Fetch the results
records = cur.fetchall()

# Print the results
print("Records where Launch Sites begin with 'KSC':")
for record in records:
    print(record)
```

```
Records where Launch Sites begin with 'KSC':
('2017-02-19', '14:39:00', 'F9 FT B1031.1', 'KSC LC-39A', 'SpaceX CRS-10', 2490, 'LEO (ISS)', 'NASA (CRS)', 'Succe
ss', 'Success (ground pad)')
('2017-03-16', '6:00:00', 'F9 FT B1030', 'KSC LC-39A', 'EchoStar 23', 5600, 'GTO', 'EchoStar', 'Success', 'No atte
mpt')
('2017-03-30', '22:27:00', 'F9 FT  B1021.2', 'KSC LC-39A', 'SES-10', 5300, 'GTO', 'SES', 'Success', 'Success (dron
e ship)')
('2017-05-01', '11:15:00', 'F9 FT B1032.1', 'KSC LC-39A', 'NROL-76', 5300, 'LEO', 'NRO', 'Success', 'Success (grou
nd pad)')
('2017-05-15', '23:21:00', 'F9 FT B1034', 'KSC LC-39A', 'Inmarsat-5 F4', 6070, 'GTO', 'Inmarsat', 'Success', 'No a
ttempt')
```

# Data Analysis with SQL

## Total Payload Mass

### What

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here

### Illustration

```
# SQL query to display the total payload mass carried by boosters launched by NASA (CRS)
query = 'SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "Customer" = "NASA (CRS)"'
cur.execute(query)

# Fetch the result
total_payload_mass = cur.fetchall()

# Print the total payload mass
print("Total payload mass carried by boosters launched by NASA (cRS):", total_payload_mass)
```

```
Total payload mass carried by boosters launched by NASA (cRS): [(45596,)]
```

# Data Analysis with SQL

## Average Payload Mass by F9 v1.1

### What

- Calculate the average payload mass carried by booster version F9 v1.1
- Present your query result with a short explanation here

### Illustration

```
# SQL query to display average payload mass carried by booster version F9 v1.1
query = query = 'SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "Booster_Version" = "F9 v1.1"'
cur.execute(query)

# Fetch the result
avg_payload_mass = cur.fetchall()

# Print the total payload mass
print("Average payload mass carried out by booster version F9 v1.1:", avg_payload_mass)
```

```
Average payload mass carried out by booster version F9 v1.1: [(2928.4,)]
```

# Data Analysis with SQL

## First Successful Ground Landing Date

### What

- Find the dates of the first successful landing outcome on drone ship.
- Present your query result with a short explanation here

### Illustration

```
# SQL query to list the date where the succesful landing outcome in drone ship was achieved
query = query = 'SELECT min("Date") FROM SPACEXTBL WHERE  "Landing_Outcome" = "Success (drone ship)"'
cur.execute(query)

# Fetch the result
succesfull_landing = cur.fetchall()

# Print the total payload mass
print("First date where succesfull landing in drone shipe:", succesfull_landing)
```

```
Firste date where succesfull landing in drone shipe [('2016-04-08',)]
```

# Data Analysis with SQL

## Successful Drone Ship Landing with Payload between 4000 and 6000

### What

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
- Present your query result with a short explanation here

### Illustration

```
# SQL query to list the names of the boosters which have success in ground pad and have payload mass greater thar
query = query = 'SELECT "Booster_Version" FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (ground pad)" AND "PA
cur.execute(query)

# Fetch the result
boosters_ground_pad = cur.fetchall()

# Print the total payload mass
print("Boosters success in ground pad payload > 4000 and < 6000:", boosters_ground_pad)
```

```
Boosters success in ground pad payload > 4000 and < 6000 [('F9 FT B1032.1',), ('F9 B4 B1040.1',), ('F9 B4 B1043.
1',)]
```

# Data Analysis with SQL

## Total Number of Successful and Failure Mission Outcomes

### What

- Calculate the total number of successful and failure mission outcomes
- Present your query result with a short explanation here

### Illustration

```
# SQL query to count successful and failed mission outcomes
query = 'SELECT "Mission_Outcome", COUNT(*) AS "Total" FROM SPACEXTBL GROUP BY "Mission_Outcome"'

cur.execute(query)

# Fetch all results
total_mission_outcomes = cur.fetchall()

# Print the results
print("Total number of successful and failed mission outcomes:")
for outcome in total_mission_outcomes:
    print(f"{outcome[0]}: {outcome[1]}")
```

```
Total number of successful and failed mission outcomes:
Failure (in flight): 1
Success: 98
Success : 1
Success (payload status unclear): 1
```

# Data Analysis with SQL

## Boosters Carried Maximum Payload

### What

- List the names of the booster which have carried the maximum payload mass
- Present your query result with a short explanation here

### Illustration

```
# SQL query to list all the booster_versions that have carried the maximum payload mass.
query = 'SELECT "Booster_Version" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FRO

cur.execute(query)

# Fetch all results
booster_version_max_payload = cur.fetchall()

# Print the results
print("Booster_versions that have carried the maximum payload mass:", booster_version_max_payload)
```

```
Booster_versions that have carried the maximum payload mass: [('F9 B5 B1048.4',), ('F9 B5 B1049.4',), ('F9 B5 B105
1.3',), ('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 ',)]
```

# Data Analysis with SQL

## 2017 Launch Records

### What

- List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Present your query result with a short explanation here

### Illustration

41.5 KB

```
# SQL query to get the required records for failed landings on a drone ship in 2015
query = '''
SELECT
    CASE substr("Date", 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
    END AS month_name,
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Success (ground pad)'
    AND "Date" LIKE "2017%"
'''

cur.execute(query)

# Fetch all results
success_ground_pad_2017 = cur.fetchall()

# Print the results
print("Month Name | Landing Outcome | Booster Version | Launch Site")
print("-----------------------------------------------------------")
for record in success_ground_pad_2017:
    print(f"{record[0]} | {record[1]} | {record[2]} | {record[3]}")
```

```
Month Name | Landing Outcome | Booster Version | Launch Site
-----------------------------------------------------------
February | Success (ground pad) | F9 FT B1031.1 | KSC LC-39A
May | Success (ground pad) | F9 FT B1032.1 | KSC LC-39A
June | Success (ground pad) | F9 FT B1035.1 | KSC LC-39A
August | Success (ground pad) | F9 B4 B1039.1 | KSC LC-39A
September | Success (ground pad) | F9 B4 B1040.1 | KSC LC-39A
December | Success (ground pad) | F9 FT  B1035.2 | CCAFS SLC-40
```

# Data Analysis with SQL

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

### What

- 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
- Present your query result with a short explanation here

### Illustration

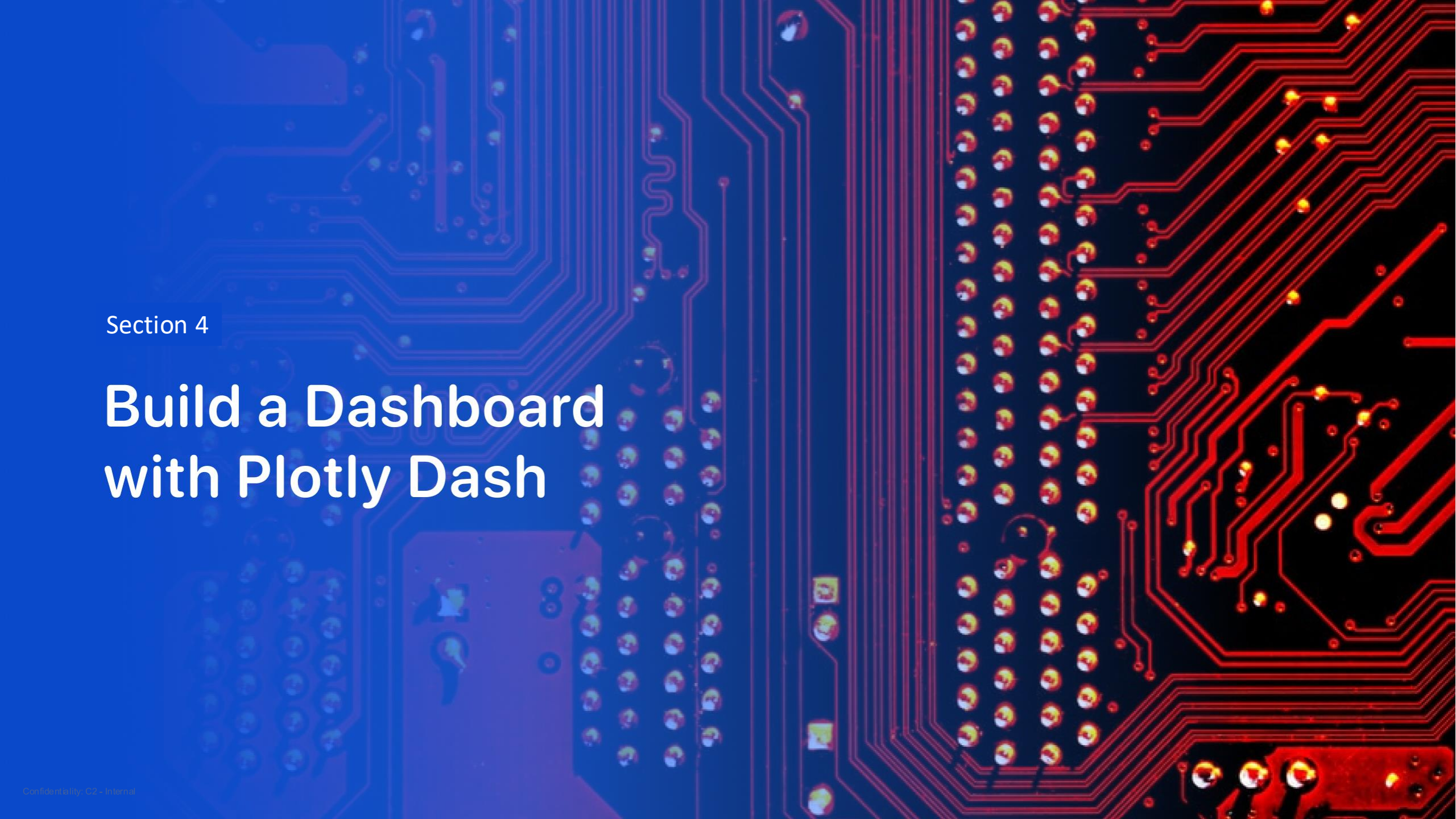
```
# SQL query to rank the count of landing outcomes
query = '''
SELECT "Landing_Outcome", COUNT(*) AS outcome_count
FROM SPACEXTBL
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY outcome_count DESC;
'''

# Execute the query
cur.execute(query)

# Fetch all results
rank_landing_outcomes = cur.fetchall()

# Display the results
for row in rank_landing_outcomes:
    print(row)
```

```
('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 4

# Build a Dashboard with Plotly Dash

# Dashboarding with Ploty Dash

## What

- *Data dashboarding* to visualize relation between launch site, payload mass, and success rate via IBM dataset (data7, python, Github)

## How

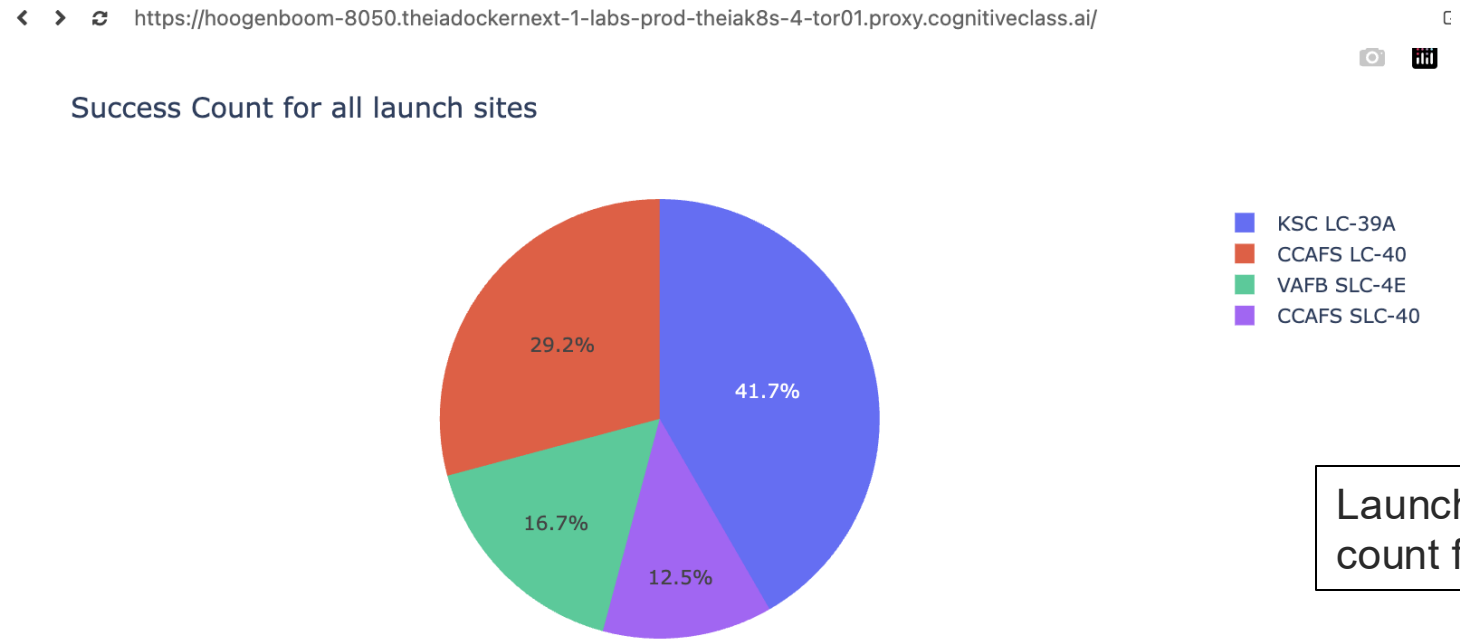
- Add a Launch Site Drop-down Input Component
- Add a callback function to render success-pie-chart based on selected site dropdown
- Add a Range Slider to Select Payload
- Add a callback function to render the success-payload-scatter-chart scatter plot

# Dashboarding with Ploty Dash

## What

- Show the screenshot of launch success count for all sites, in a pie chart
- Explain the important elements and findings on the screenshot

## Illustration



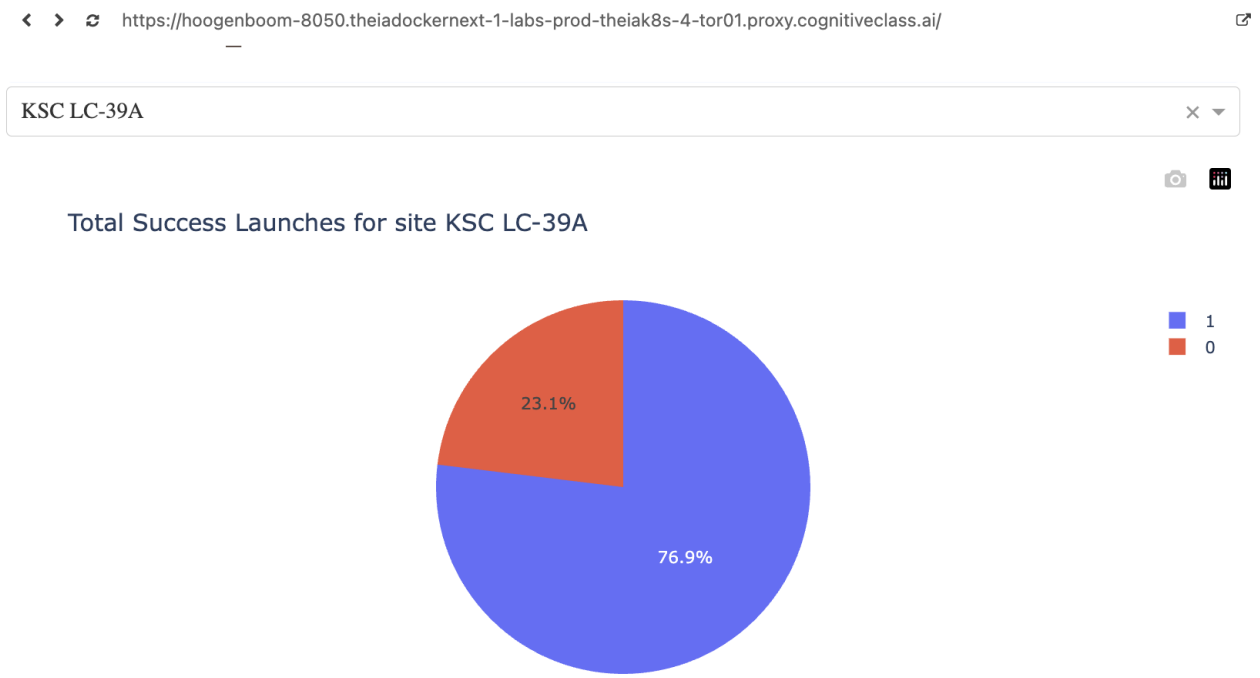

Launch Site KSC LC-39A has highest count for successful launches

# Dashboarding with Ploty Dash

## What

- Show the screenshot of the pie chart for the launch site with highest launch success ratio
- Explain the important elements and findings on the screenshot

## Illustration




Launch Site KSC LC-39A has highest share [76,9%] for successful [1] launches

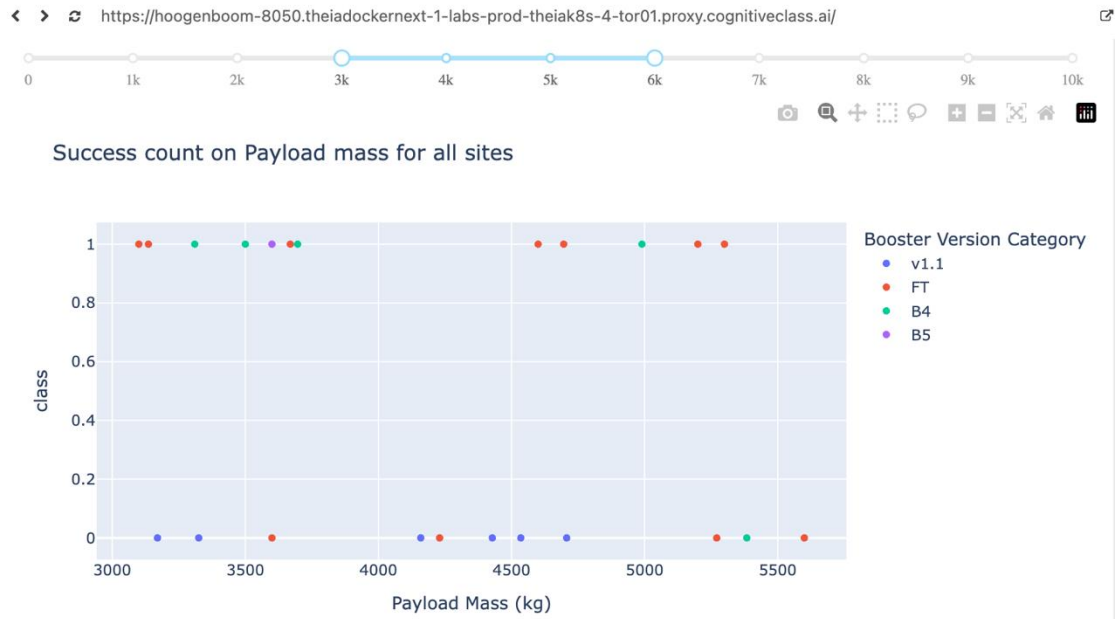
# Dashboarding with Ploty Dash

## What

- Screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in range slider
- Explain important elements and findings on screenshot, such as which payload range or booster version have largest success rate, etc.

## Illustration

Booster Version FT has most successful [1] launches

Section 5

# Predictive Analysis (Classification)

# Predictive Analysis (Classification)

## What

- *Predictive modelling* via LR, SVM, decision tree, and KNN to understand scoring accuracy via IBM dataset (data5, Github)

## How

- Load external data set into dataframe
- Standardized the data, split our data in training and test data (X – launch orbit details, Y – launch success or failure)
- Built different Logistic Regression, SVM, Decision Tree, K-Nearest Neighbor machine learning models and tune different hyperparameters using GridSearch CV.
- Identify most accurate classification machline learning model via Training Data (Decision tree)
- Validate most accurate classification machine learning model on Test Data and illustrate via Confusion Matric FP/ TP

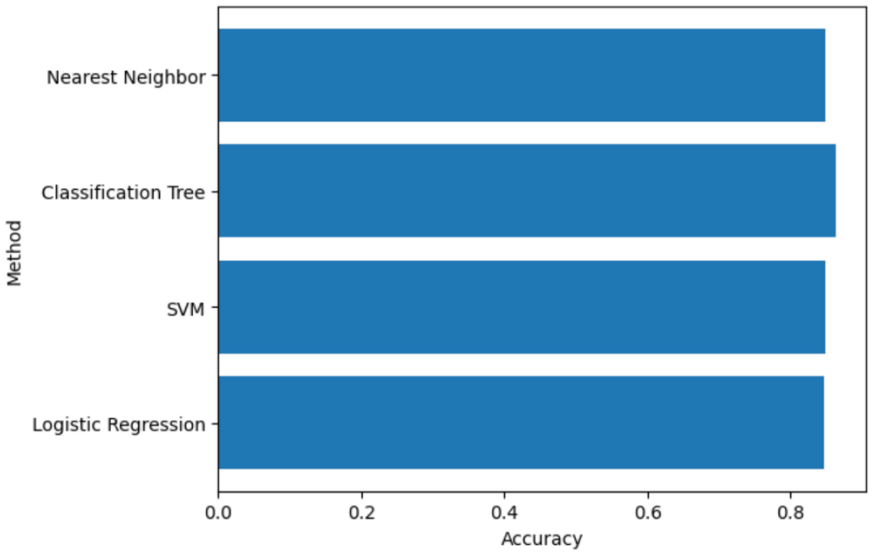
# Predictive Analysis

## Classification Accuracy

### What

- Visualize the built model accuracy for all built classification models, in a bar chart
- Find which model has the highest classification accuracy

### Illustration




```
Accuracy Logistic Regression 0.8464285714285713
Parameters Logistic Regression {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
Accuracy SVM 0.8482142857142856
Parameters SVM {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
Classification tree 0.8767857142857143
Parameters Classification Tree {'criterion': 'entropy', 'max_depth': 16, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_sp
lit': 10, 'splitter': 'random'}
Nearest neighbor 0.8482142857142858
Parameters Nearest neighbor {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
```

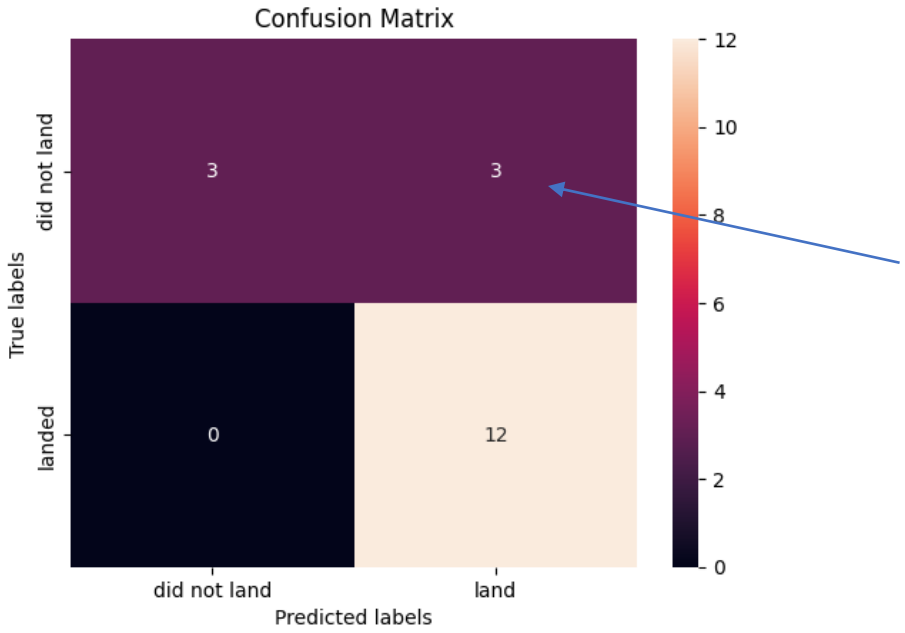
# Predictive Analysis

## Confusion Matrix

### What

- Show the confusion matrix of the best performing model with an explanation

### Illustration




The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes.

Major problem is false positives → Incorrectly predicting successful landing, whereas the actual landing was unsuccesfull

# Capstone Assignment

## Conclusions

- **Data visualization** The higher the number of fights the greater the chance of a successful Launch_Outcome
- **Data visualization** Over the years the success rates increases *(ed: probably due to lessons learned over time)*
- **Data visualization** Orbits (ES-L1, GEO, HEO) SSO, VLEO have the highest average success in Launch_Outcome, however for the Orbits ()  only 1 flight was executed.
- **Data visualization** Over the years the success rates increased
- **Dashboarding** Kenney Space Centre (KSC) LC-39A had the most successful launches [76.9%] of all launch sites
- **Predictive Analysis** The ML model Decision Tree Classifier has the highest accuracy score [87.7%], but still has weakness by incorrectly predicting successful landing, whereas the actual landing was unsuccessful in 3 out of 18 cases.

# Conclusions

- Point 1
- Point 2
- Point 3
- Point 4
- …

# 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!