IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Trung Bui
November 14th, 2025

Outline

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

Executive Summary

- Data gathered from SpaceX API
- Visualized relationships between features and mission outcome
- Built an interactive dashboard
- Identified 4 key features affecting launch mission outcome
- Applied 4 ML classification method to model the launch outcome prediction
- Determined the most accurate classification – Decision Tree

Introduction

- Gathering SpaceX launch missions' data from public sources
- Creating data dashboards
- Analyzing data to determine the cost of each launch
- Train ML models to predict if the first stage is reused

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Use SpaceX REST API to get data on past launches
  - Use BeautifulSoup to scrap data
- Perform data wrangling
  - Visualize data, clean, and deal with missing data,
  - Calculate and generate needed features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Split the data into training and test sets
  - Apply different ML models to select the best accuracy score, and review the confusion matrix

Data Collection

- Describe how data sets were collected.
- Request rocket launch data using SpaceX API (link)
- Turn JSON file format into a Pandas dataframe
- Call a series of functions using the API to extract the necessary information using identification numbers from the downloaded launch data
- Filter data to include only Falcon 9 launches
- Check for missing data and replace missing data with means

Data Collection – SpaceX API

- Request the rocket launch data from SpaceX API

  static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'

  response = requests.get(static_json_url)

- Decode the response content as JSON, then turn it into a Pandas dataframe, and deal with missing data,

  data_json=response.json()

  data = pd.json_normalize(data_json)

  data_falcon9['PayloadMass']=data_falcon9['PayloadMass'].replace(np.nan,mean_payload)

Data Collection – Scraping

- Request the Falcon 9 and Falcon Heavy Launch data

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

  response = requests.get(static_url)

- Use BeautifulSoup to scrape data

  soup = BeautifulSoup(response.text,"html.parser“)

- Create a Dataframe from a dictionary of features, save to a CSV file for later use

  df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })

  df.to_csv('spacex_web_scraped.csv', index=False)

Data Wrangling

- Request the Falcon 9 and Falcon Heavy Launch data

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

  response = requests.get(static_url)

- Use BeautifulSoup to scrape data

  soup = BeautifulSoup(response.text,"html.parser“)

- Create a Dataframe from a dictionary of features, save to a CSV file for later use

  df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })

  df.to_csv('spacex_web_scraped.csv', index=False)

EDA with Data Visualization

- Visualize features to help determine their relationships with the success rate
  - Flight Number and Launch Site
  - Payload Mass and Launch Site
  - Success Rate and Orbit Type
  - Flight Number and Orbit Type
  - Payload Mass and Orbit Type
  - Plot the Launch Success Yearly Trend

EDA with SQL

- The following SQL queries were performed in the EDA stage
  - Display unique launch sites' names
  - Display 5 records of launch sites beginning with 'CCA'
  - Display the total payload mass carried by boosters launched by NASA (CSR)
  - Display average payload mass carried by booster version F9 v1.1
  - List the date of the first successful landing in ground pad
  - List names of boosters with success in drone ship, and have payload mass between 4000 and 6000 kg
  - List the total number of successful and failure mission outcomes
  - List all booster versions that have carried the maximum payload using subquery
  - List records with months in 2015 that have failure in drone ship outcomes
  - Rank the count of landing outcomes between 2010-06-03 and 2017-03-20 in descending order

Build an Interactive Map with Folium

- Add map objects
  - Circle for launch sites
  - Colored markers for success/failure launch at each site
  - Add lines from each site's nearest coastline, railways, highways, and cities
  - Calculate and add distances to those lines
- Visualized maps help to identify the optimal location to build a launch site

Build a Dashboard with Plotly Dash

- The following features were added to the Dashboard
  - Launch Site Drop-down input
  - Add a callback function to render a Success pie chart based on the selected site
  - Add a Range slide to select payload
  - Add a callback function to render a Success payload scatter plot
- Those plots help identify a correlation between Payload and Success for all sites

Predictive Analysis (Classification)

- Create a Class column – the Y variable
- Standardize data with preprocessing.StandardScaler()
- Split data X and Y into training and test data (80/20 split)
- Estimate parameters, calculate accuracy, and plot confusion matrix for
  - Logistic regression
  - Support Vector Machine
  - Decision Tree
  - K-Nearest Neighbor
- Select the method that performs best on the test data – Decision Tree

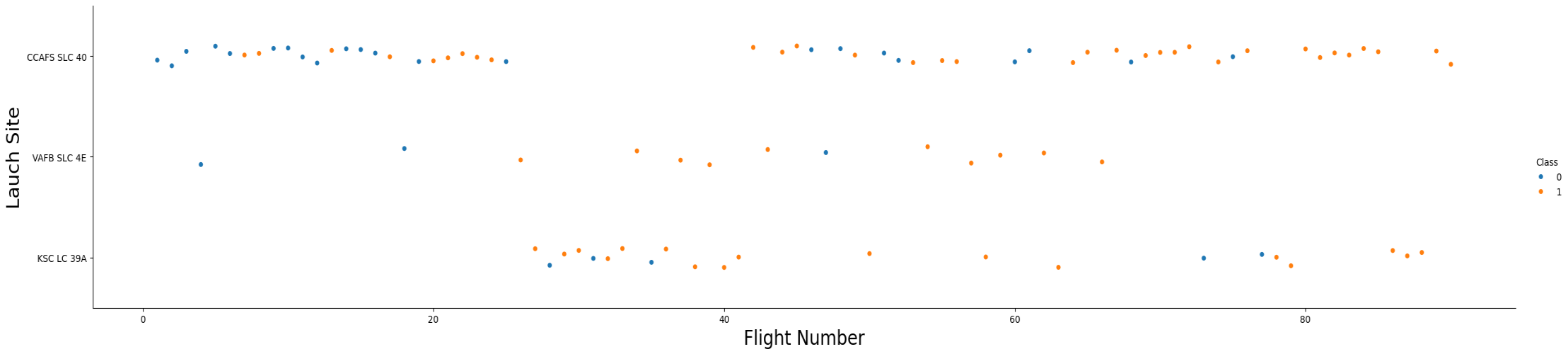
Results

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

Section 2

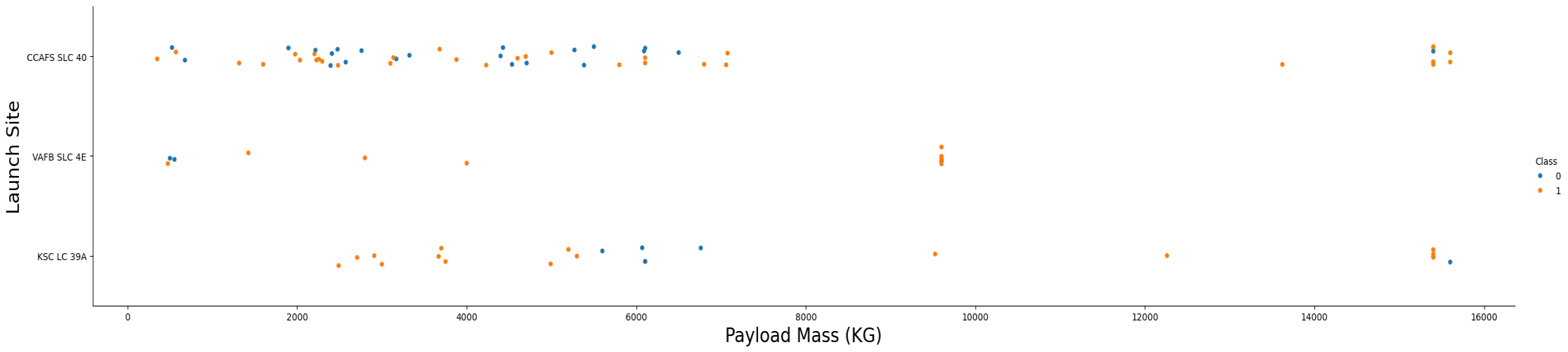
Insights drawn from EDA

Flight Number vs. Launch Site



Earlier flights had more failures across launch sites, although most were launched from CCAFS SLC 40.
Success rate has increased dramatically after the first 20 flights.
The VAFB SLC 4E has the most success rate among the launch sites.

Payload vs. Launch Site

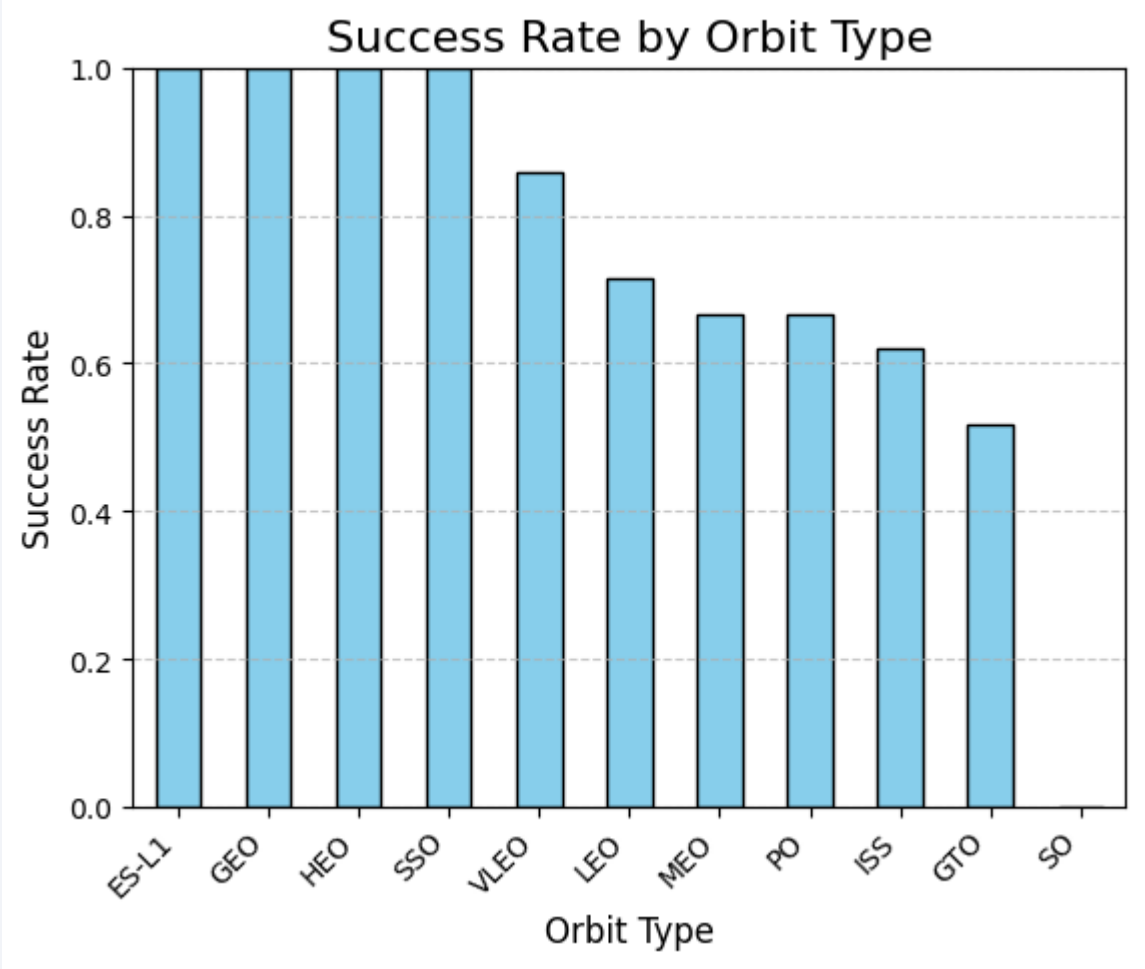


Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavy payload mass(greater than 10000).
Most of heavy payload launches greater than 8000kg were successful. CCAFS SLC 40 has the most launches less than 8000kg and highest rates of failure.
VAFB SLC 4E has the most success with launches less than 10000kg but also has the least number of launches.

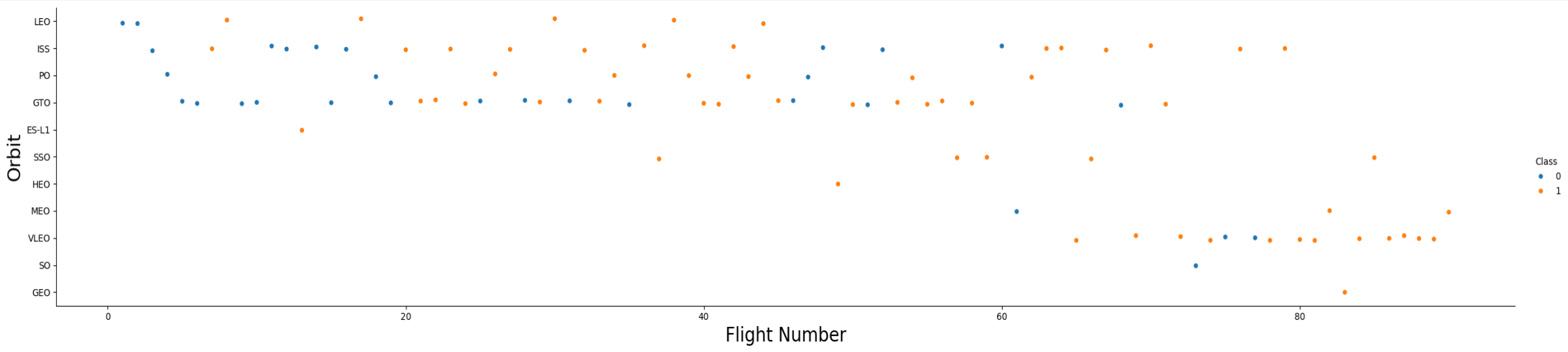
Success Rate vs. Orbit Type

The following four orbits have the 100% success rate:

- ES-L1
- GEO
- HEO
- SSO

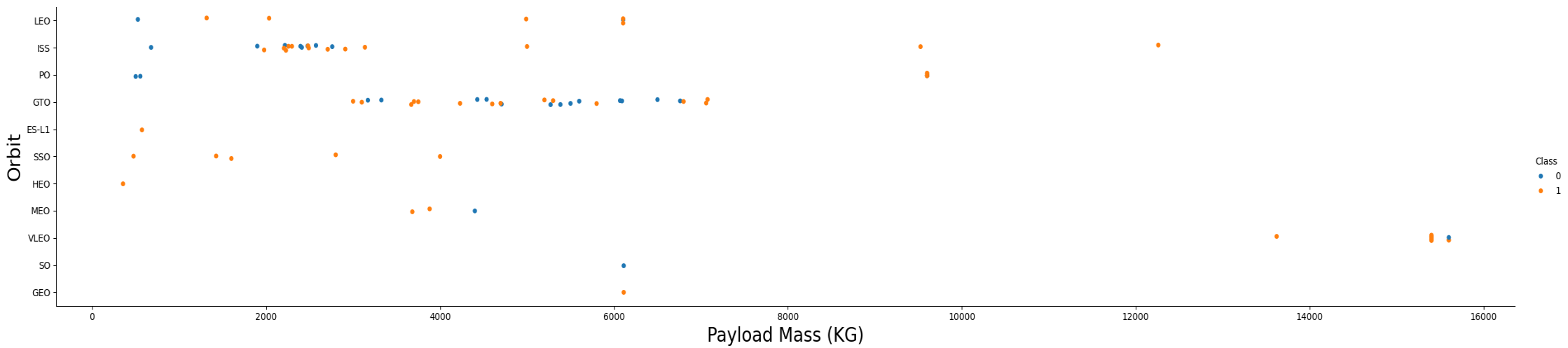


Flight Number vs. Orbit Type



You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

Payload vs. Orbit Type

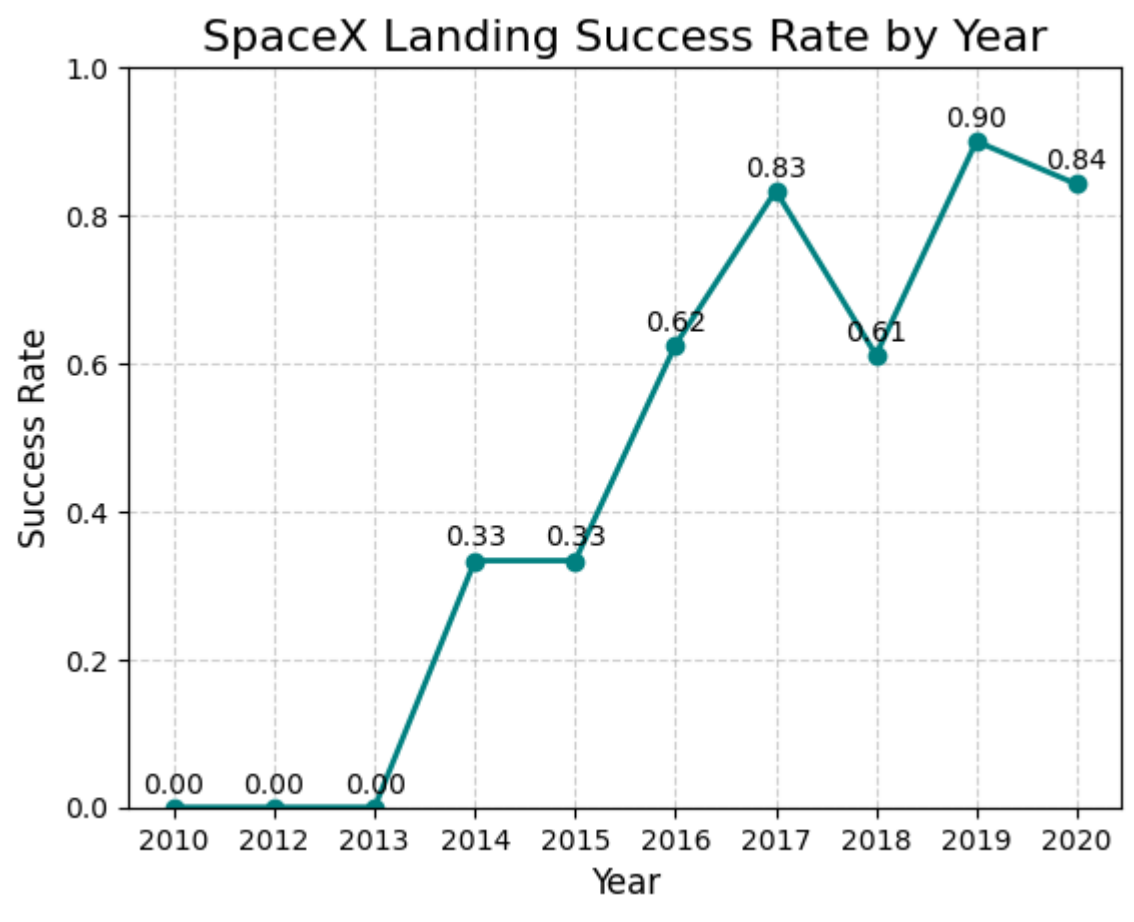


With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend

you can observe that the sucess rate since 2013 kept increasing till 2020



All Launch Site Names

```
# SQL query
query = """
SELECT DISTINCT "Launch_Site"
FROM SPACEXTABLE;
"""

# Execute query and load results into Pandas DataFrame
unique_sites_df = pd.read_sql_query(query, con)

# Display the result
print(unique_sites_df)
```

	Launch_Site
0	CCAFS LC-40
1	VAFB SLC-4E
2	KSC LC-39A
3	CCAFS SLC-40

- “DISTINCT” is used to select unique launch site names from the SpaceXTable table

Launch Site Names Begin with 'CCA'

```
# SQL query to get 5 records where launch site begins with 'CCA'
query = """
SELECT *
FROM SPACEXTABLE
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5;
"""

# Execute the query and load results into Pandas DataFrame
cca_sites_df = pd.read_sql_query(query, con)

# Display the result
print(cca_sites_df)
```

```
         Date Time (UTC) Booster_Version  Launch_Site  \
0  2010-06-04   18:45:00  F9 v1.0  B0003  CCAFS LC-40
1  2010-12-08   15:43:00  F9 v1.0  B0004  CCAFS LC-40
2  2012-05-22    7:44:00  F9 v1.0  B0005  CCAFS LC-40
3  2012-10-08    0:35:00  F9 v1.0  B0006  CCAFS LC-40
4  2013-03-01   15:10:00  F9 v1.0  B0007  CCAFS LC-40

                                                Payload  PAYLOAD_MASS__KG_  \
0                Dragon Spacecraft Qualification Unit                  0
1  Dragon demo flight C1, two CubeSats, barrel of...                   0
2                                Dragon demo flight C2                525
3                                          SpaceX CRS-1                500
4                                          SpaceX CRS-2                677

       Orbit         Customer Mission_Outcome      Landing_Outcome
0        LEO           SpaceX         Success  Failure (parachute)
1  LEO (ISS)  NASA (COTS) NRO         Success  Failure (parachute)
2  LEO (ISS)      NASA (COTS)         Success           No attempt
3  LEO (ISS)       NASA (CRS)         Success           No attempt
4  LEO (ISS)       NASA (CRS)         Success           No attempt
```

- Like ‘CCA%’ is used to select site names begin with ‘CCA’ only
- LIMIT 5 is used to get 5 records

Total Payload Mass

```
# SQL query to get total payload mass for NASA (CRS)
query = """
SELECT SUM("PAYLOAD_MASS__KG_") AS total_payload_mass
FROM SPACEXTABLE
WHERE "Customer" = 'NASA (CRS)';
"""

# Execute the query and load results into Pandas DataFrame
nasa_payload_df = pd.read_sql_query(query, con)

# Display the result
print(nasa_payload_df)
```

```
   total_payload_mass
0               45596
```

- SUM() function is used to calculate total payload, limit to 'NASA (CRS)' customer

Average Payload Mass by F9 v1.1

```
# SQL query to get average payload mass carried by booster version F9 v1.1
query = """
SELECT AVG("PAYLOAD_MASS__KG_") AS avg_payload_mass
FROM SPACEXTABLE
WHERE "Booster_Version" = 'F9 v1.1';
"""

# Execute the query and load results into Pandas DataFrame
avg_v11_payload_df = pd.read_sql_query(query, con)

# Display the result
print(avg_v11_payload_df)
```

```
   avg_payload_mass
0            2928.4
```

- AVG() function is used to calculate average payload, for Booster version ‘F9 v1.1’

First Successful Ground Landing Date

```
# SQL query to get first ground pad landing success
query = """
SELECT MIN("Date") AS first_ground_pad_success
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (ground pad)';
"""

# Execute the query and load results into Pandas DataFrame
first_ground_pad_success = pd.read_sql_query(query, con)

# Display the result
print(first_ground_pad_success)
```

```
  first_ground_pad_success
0               2015-12-22
```

- MIN() function is used to select the first date with ‘Success (ground pad)’ outcome

Successful Drone Ship Landing with Payload between 4000 and 6000

```
# SQL query to get names of boosters with successful drone ship landing and payload between 4000 and 6000 kg
query = """
SELECT DISTINCT Booster_Version AS unique_booster_names
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (drone ship)' AND (PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000);
"""

# Execute the query and load results into Pandas DataFrame
boosters_names = pd.read_sql_query(query, con)

# Display the result
print(boosters_names)
```

```
  unique_booster_names
0          F9 FT B1022
1          F9 FT B1026
2       F9 FT  B1021.2
3       F9 FT  B1031.2
```

- DISTINCT is used to select unique names of booster versions that have ‘Success (drone ship)’ outcome and had a payload mass greater than 4000 but less than 6000kg

Total Number of Successful and Failure Mission Outcomes

- Count() function is used to calculate the total number of missions
- Group By is used to perform the calculation for each outcome

```
# Count unique values in the mission outcome column
query = """
SELECT "Landing_Outcome", COUNT(*) AS total_missions
FROM SPACEXTABLE
GROUP BY "Landing_Outcome";
"""

# Execute the query and load results into Pandas DataFrame
total_missions = pd.read_sql_query(query, con)

# Display the result
print(total_missions)
```

```
           Landing_Outcome  total_missions
0       Controlled (ocean)               5
1                  Failure               3
2     Failure (drone ship)               5
3      Failure (parachute)               2
4               No attempt              21
5              No attempt                1
6   Precluded (drone ship)               1
7                  Success              38
8     Success (drone ship)              14
9     Success (ground pad)               9
10    Uncontrolled (ocean)               2
```

Boosters Carried Maximum Payload

- Use the subquery to calculate the maximum payload
- Select the booster versions that have carried the maximum payload from the table

```
query = """
SELECT "Booster_Version" As boosters_max_payload
FROM SPACEXTABLE
WHERE "PAYLOAD_MASS__KG_" = (
    SELECT MAX("PAYLOAD_MASS__KG_")
    FROM SPACEXTABLE
);
"""

# Execute the query and load results into Pandas DataFrame
boosters_max_payload = pd.read_sql_query(query, con)

# Display the result
print(boosters_max_payload)
```

```
   boosters_max_payload
0          F9 B5 B1048.4
1          F9 B5 B1049.4
2          F9 B5 B1051.3
3          F9 B5 B1056.4
4          F9 B5 B1048.5
5          F9 B5 B1051.4
6          F9 B5 B1049.5
7         F9 B5 B1060.2
8         F9 B5 B1058.3
9          F9 B5 B1051.6
10         F9 B5 B1060.3
11        F9 B5 B1049.7
```

2015 Launch Records

- Use Substr() functions to get Month and Year as new columns

- Select only records in Year 2015 and ‘Failure (drone ship)’ outcome

```
query = """
SELECT
    substr("Date", 6, 2) AS Month,
    substr("Date", 0, 5) AS Year,
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM SPACEXTABLE
WHERE substr("Date", 1, 4) = '2015'
  AND "Landing_Outcome" = "Failure (drone ship)";
"""

# Execute the query and load results into Pandas DataFrame
failure_records_2015= pd.read_sql_query(query, con)

# Display the result
print(failure_records_2015)
```

	Month	Year	Landing_Outcome	Booster_Version	Launch_Site
0	01	2015	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
1	04	2015	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- COUNT() function is used
- GROUP BY to group by landing outcome
- DESC for descending order
- For dates between ‘2010-06-04’ and ‘2017-03-20’

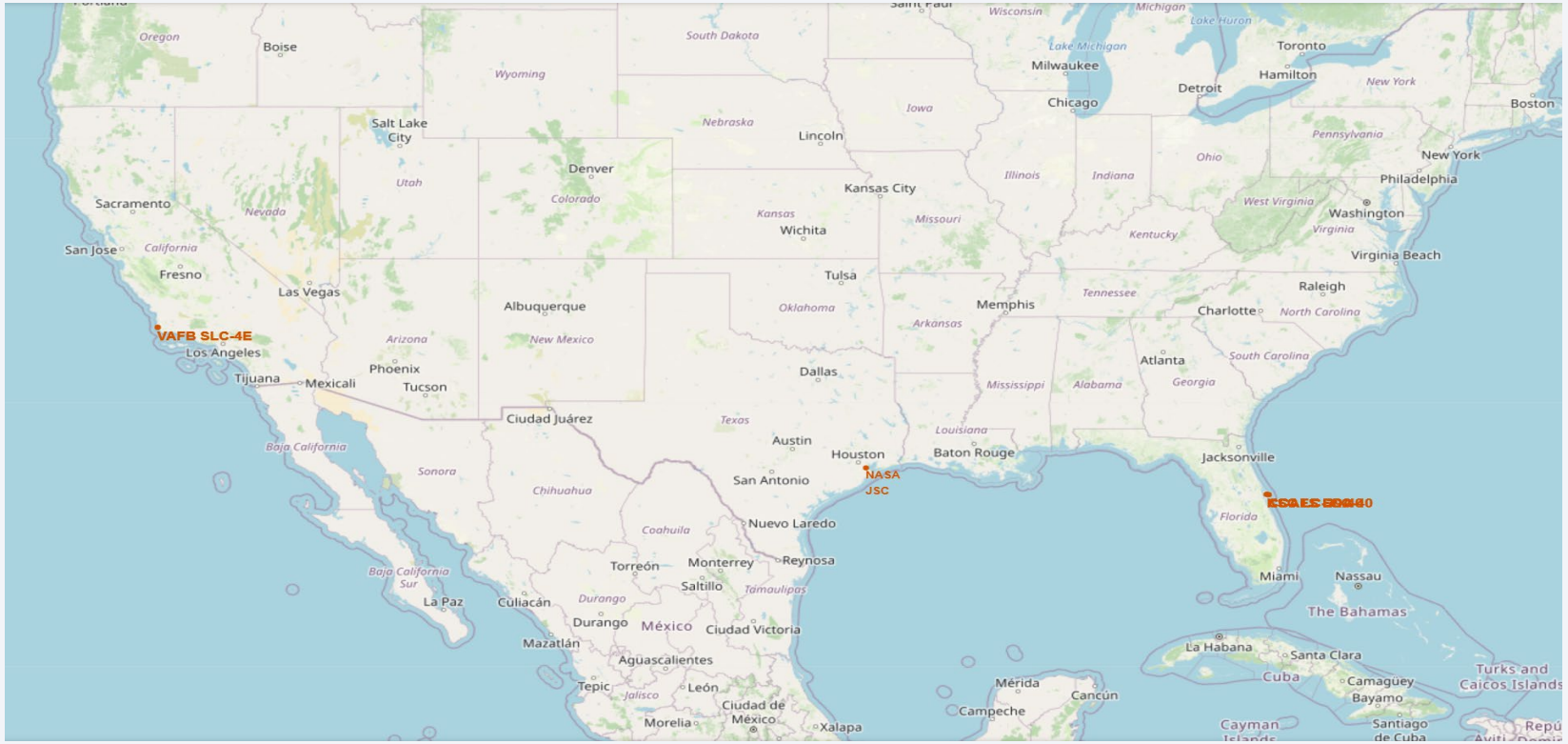
```
query = """
SELECT "Landing_Outcome", COUNT(*) AS outcome_count
FROM SPACEXTABLE
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY outcome_count DESC;
"""

# Execute the query and load results into Pandas DataFrame
landing_counts= pd.read_sql_query(query, con)

# Display the result
print(landing_counts)
```

```
          Landing_Outcome  outcome_count
0              No attempt             10
1   Success (drone ship)               5
2   Failure (drone ship)               5
3   Success (ground pad)               3
4     Controlled (ocean)               3
5   Uncontrolled (ocean)               2
6    Failure (parachute)               2
7  Precluded (drone ship)              1
```

Section 3

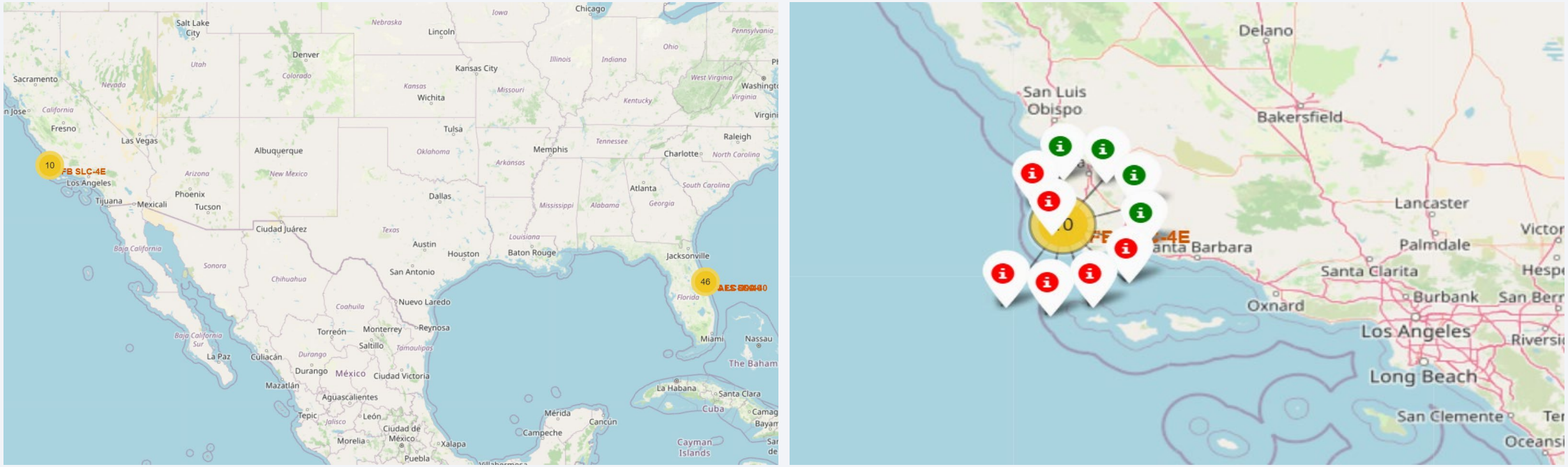
Launch Sites Proximities Analysis

Map All Launch Sites



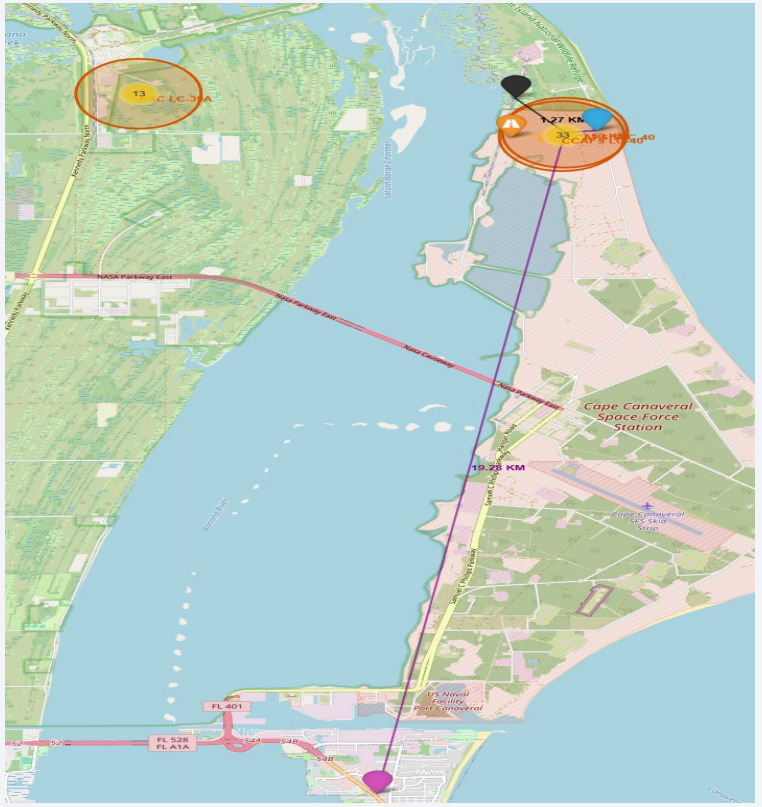
- Launch sites are very close to the coastlines, either Atlantic or Pacific ocean
- Sites are not near but not too far from major cities, for example Miami or Los Angeles

Success/Failure by site

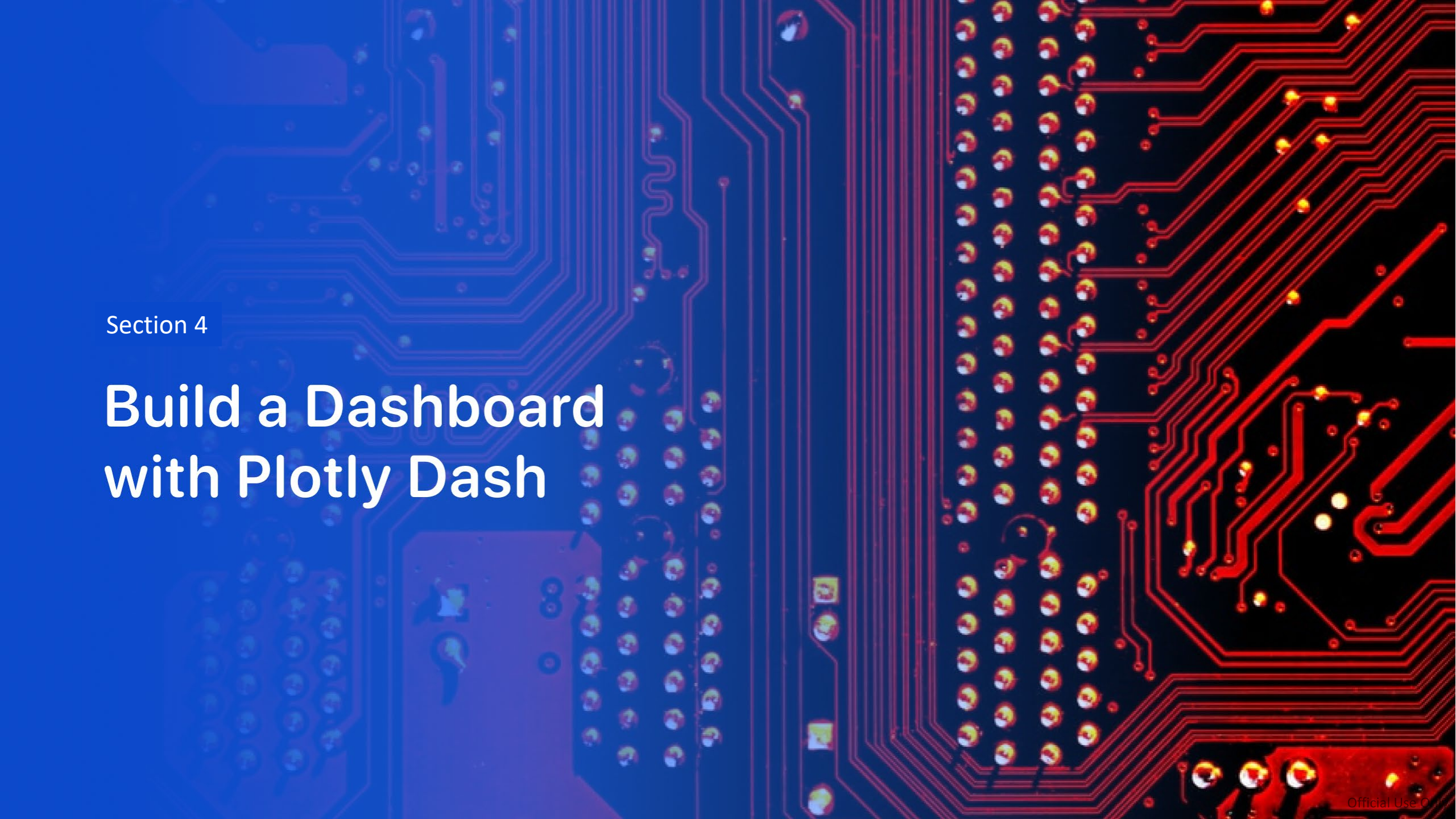


- The map shows the success/failure data for each site.
- E.g., VAFB SLC 4E, as shown above, has a low success rate.

Distances to nearest locations



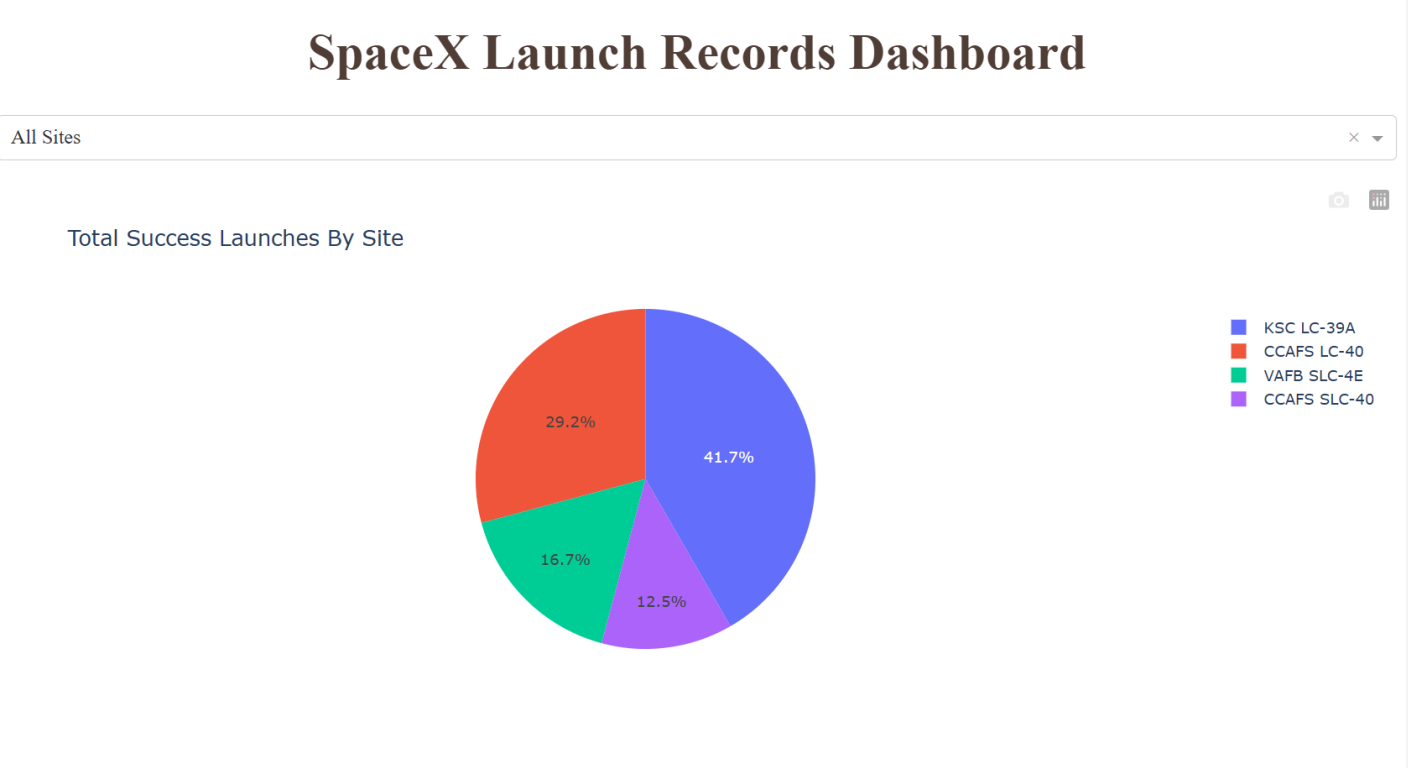
- CCAFS SLC-40 is <1mi from the nearest coastline, railways, and highways, making it easily accessible via various transportation forms
- It is >10mi away from the nearest city, more suitable for rocket launching (environmental, safety)



Section 4

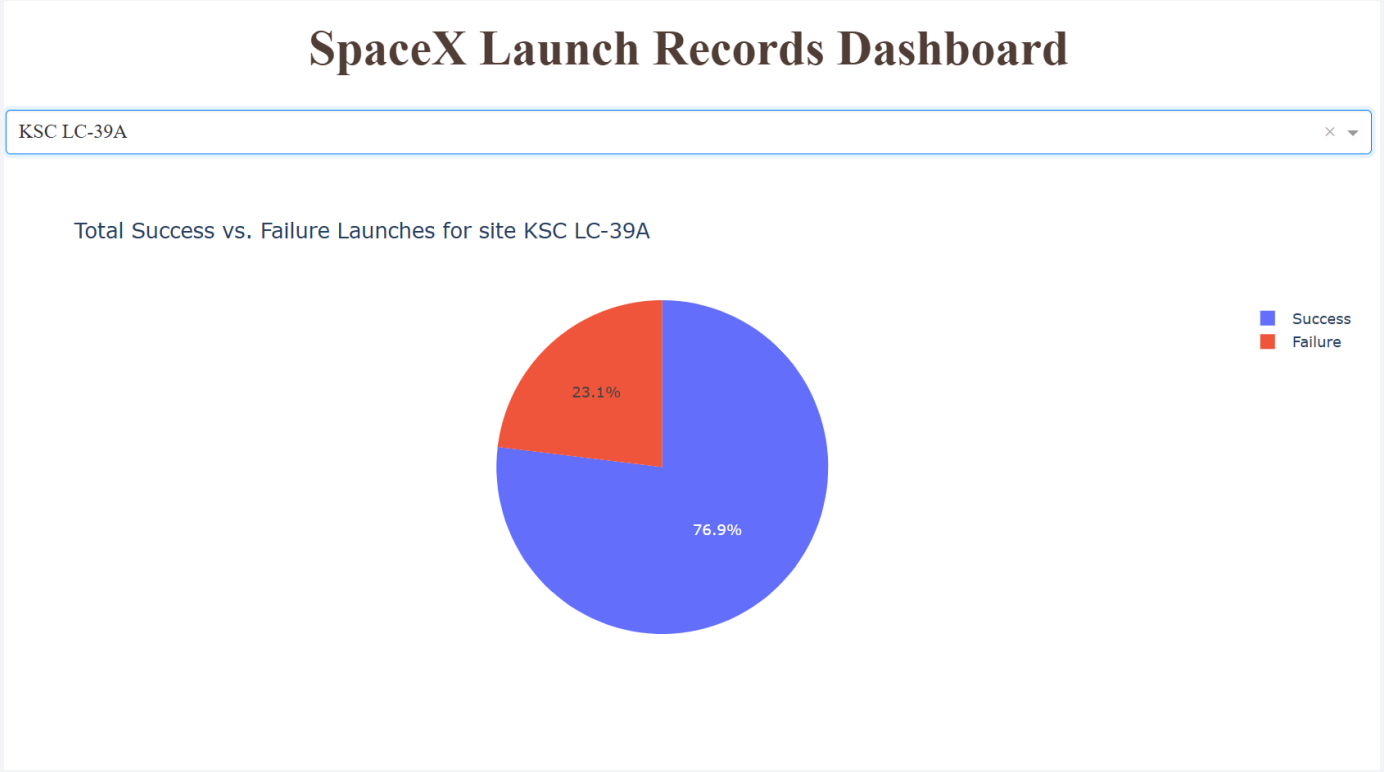
Build a Dashboard with Plotly Dash

Dashboard 1 – All sites



- Site KSC LC-39A has the most successful launches (41.7%), while CCAFS SLC-40 accounts for the least number of success launches (12.5%)

<Dashboard Screenshot 2>



- Site KSC LC-39A accounts for the most successful launches among all sites, AND also has the highest success ratio

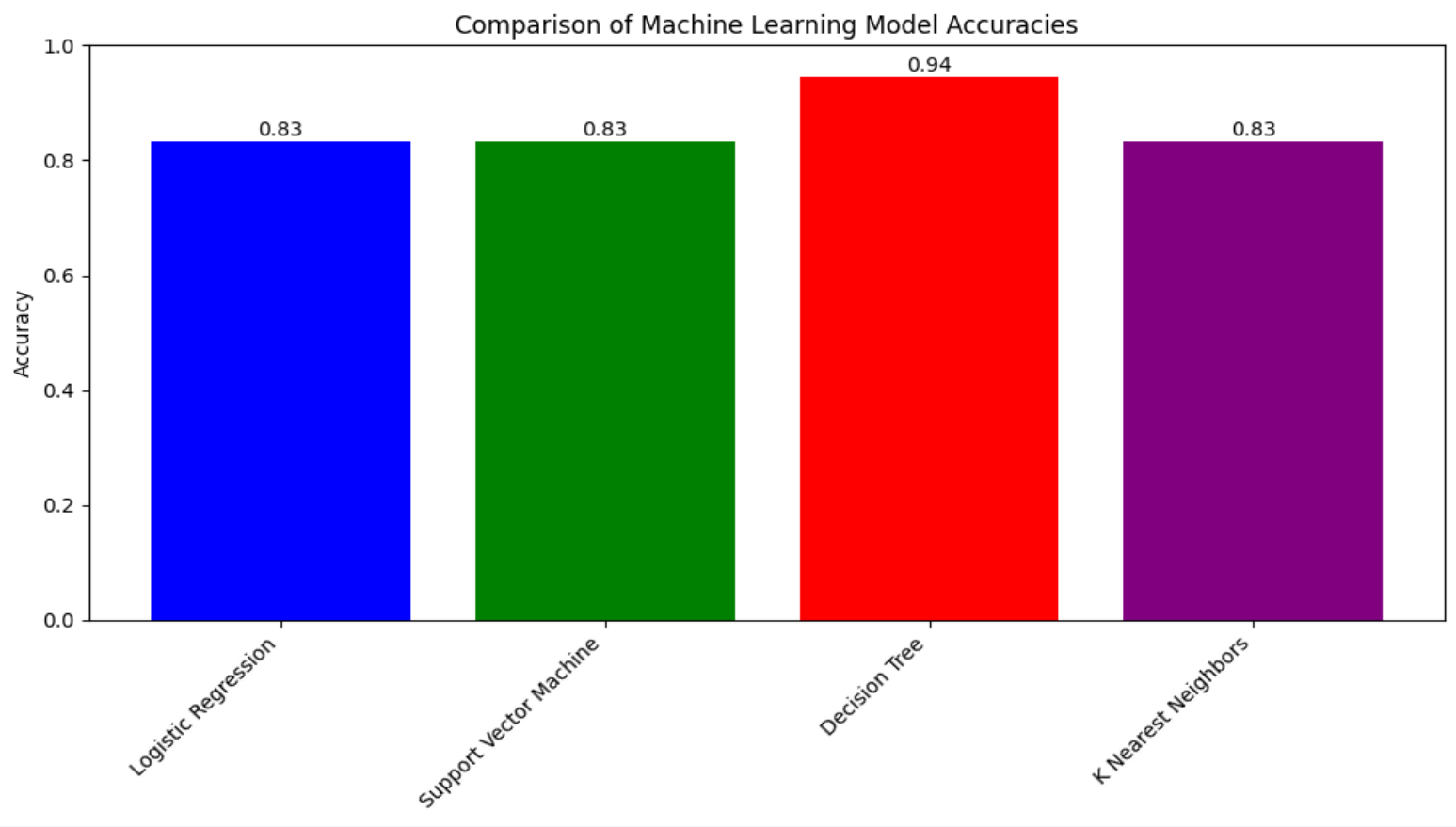
<Dashboard Screenshot 3>

- Booster version FT has the most success with a payload of less than 6000kg
- It appears no success yet for a payload larger than 6000kg

Section 5

Predictive Analysis (Classification)

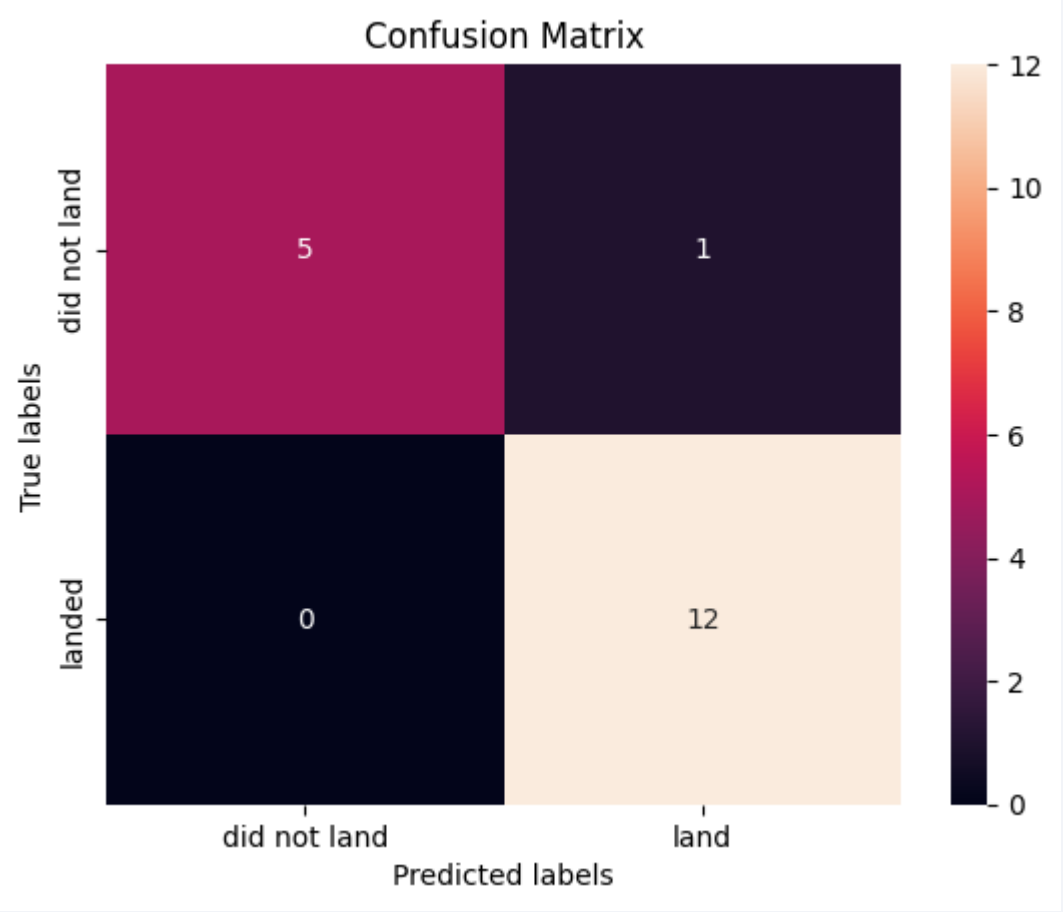
Classification Accuracy



- Decision tree has the highest accuracy among 4 classification methods

Confusion Matrix

- From the Confusion Matrix for the Decision Tree classification, 12 out of 13 predicted landing outcome are true and only one is incorrect.
- This is better than the other classifications: Logistic Regression, Support Vector Machine, and K-Nearest Neighbors



Conclusions

- Data Wrangling points to predictive features of a success mission based on
  - Booster Version
  - Orbit
  - Launch Site
  - Payload Mass
- Decision Tree classification provided the most accurate model

Appendix

- https://github.com/trungbui79/Applied-Data-Science-Capstone-Project

Thank you!