IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Zuqaqambe Ngomti
18-06-2025

Outline

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

Executive Summary

- The study was developed as means to understand the land sites of SpaceX ships and if they fail or are successes, we then understood such by producing maps in of the current launch sites with the statistics of each.
- The results I achieved is that the tree prediction provides more accuracy than the other predictive models that I implemented.

Introduction

- Project background and context
- Problems you want to find answers

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was sourced online using an API.
- Perform data wrangling
  - Data was processed by calculating the number of launch sites we have and how each had performed.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Data was sourced online using an API.
- You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API

- https://github.com/Mac-Zuch/IBM/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

From the `rocket` column we would like to learn the booster name.

```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

Data Collection - Scraping

- The full code can be found on the github lab which shows the whole process and outcomes.

- https://github.com/Mac-Zuch/IBM/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

Data Wrangling

- This is how the data was processed.
- Calculate the number of launches on each site
- Calculate the number and occurrence of each orbit
- Calculate the number and occurence of mission outcome of the orbits
- Create a landing outcome label from Outcome column

EDA with Data Visualization

- The charts that were plotted were,
    1. Scatter plot – To see the relationship of the variable's vs the flight number.
    2. Bar chart – the relationship between the success rate of each orbit type.
    3. Line chart – for the yearly trend.
- https://github.com/Mac-Zuch/IBM/blob/main/edadataviz.ipynb

EDA with SQL

- **This is the list of all queries that were performed, due to space constraints you can use the link to my github to see the full extent of the code.**
- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first succesful landing outcome in ground pad was acheived.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.
- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.
- 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.
- https://github.com/Mac-Zuch/IBM/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- The following are map objects that I created are markers, circles, lines, etc. Just to name a few.
- I added the elements above for better readability of the map and to provide context.
- https://github.com/Mac-Zuch/IBM/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Pie chart and a scatter plot.
- To see the relationships between the variables.
- https://github.com/Mac-Zuch/IBM/blob/main/Dash.ipynb

Predictive Analysis (Classification)

- I built the models using tree classifier, svn and knn and logical regression.
- You need present your model development process using key phrases and flowchart
- https://github.com/Mac-Zuch/IBM/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

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

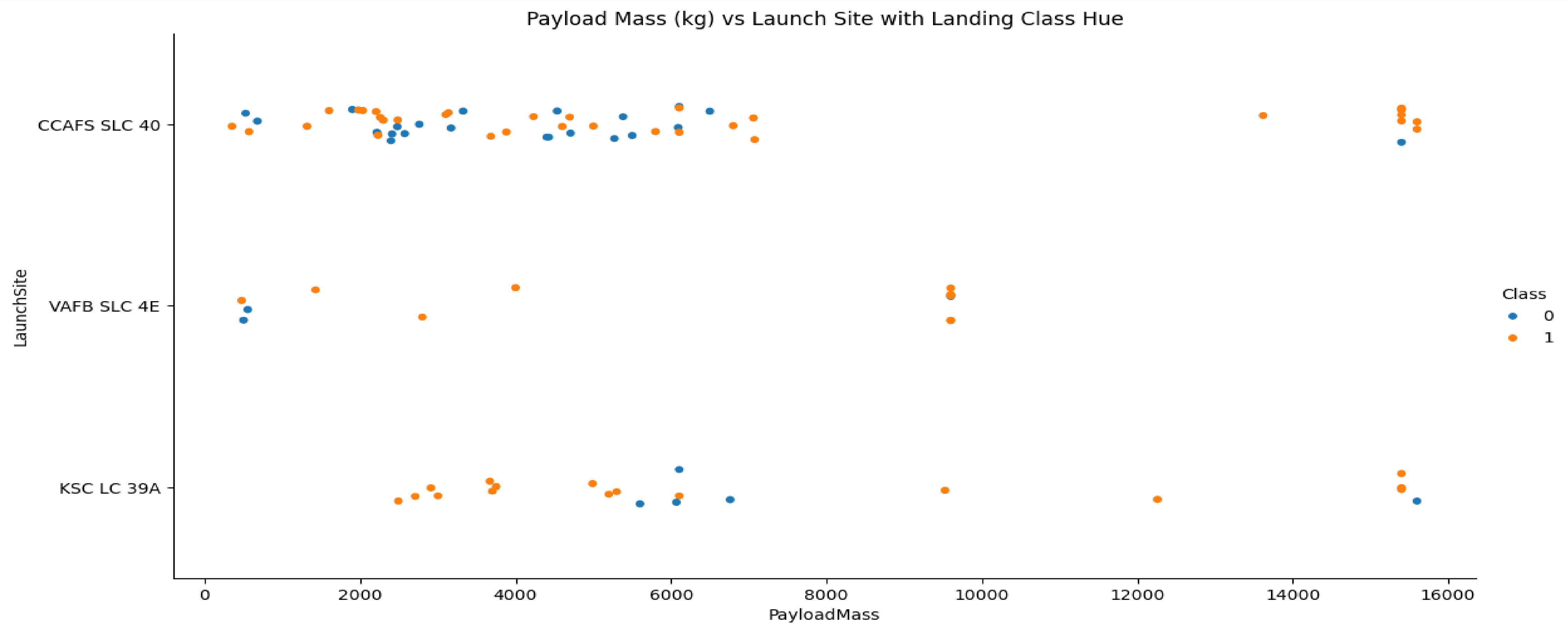
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

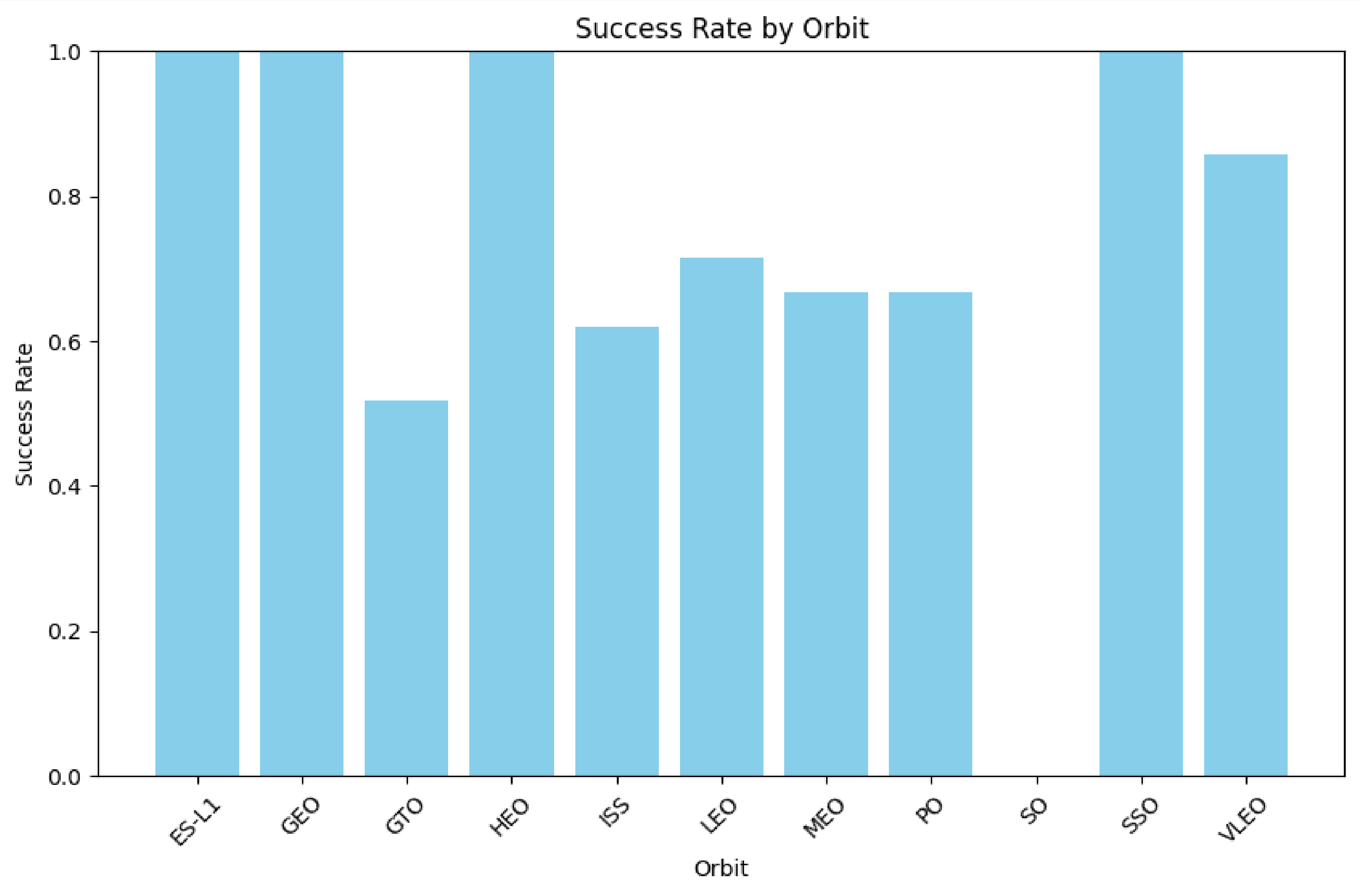
The scatter plot shows that for SLS 40, over 40 flights were a success but still recorded the most failures

Payload vs. Launch Site



There has been a few flights that have had a mass payload over 8000, and even then only 2 failures were recorded .

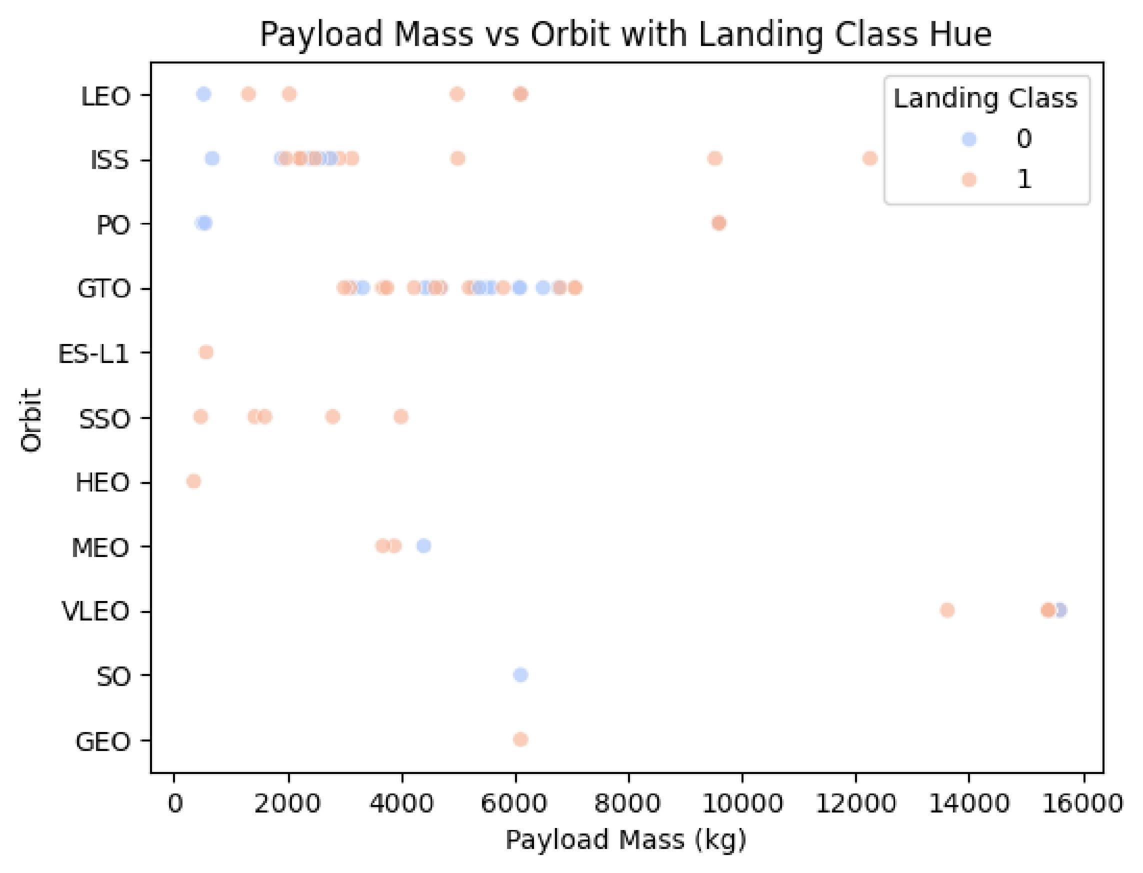
Success Rate vs. Orbit Type



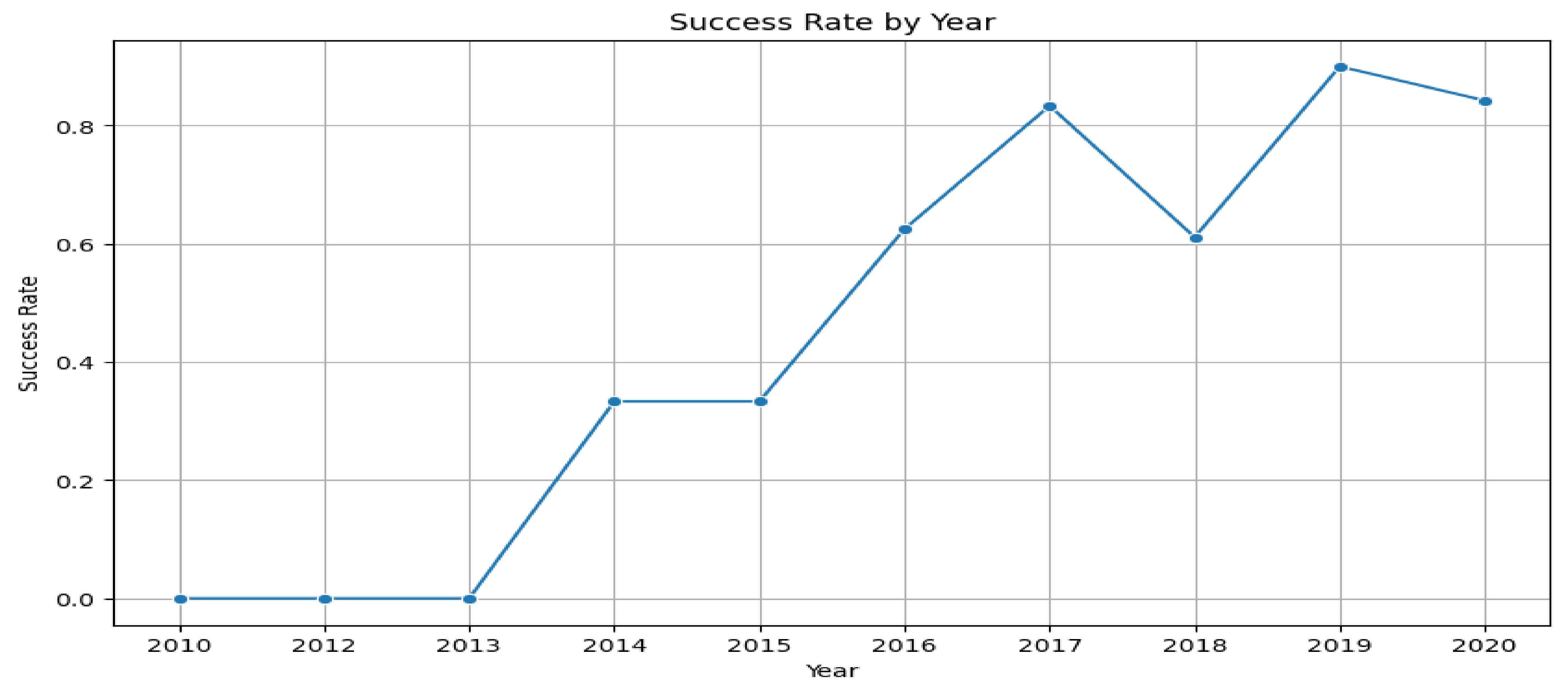
4 Orbits have a perfect Success rate.

Flight Number vs. Orbit Type

Payload vs. Orbit Type



Launch Success Yearly Trend



2019 Recorded the best success rate whereas there has been an increase from 2013 unit 2017.

All Launch Site Names

- Find the names of the unique launch sites
    1. CCAFS LC-40
    2. VAFB SLC-4E
    3. KSC LC-39A
    4. CCAFS SLC-40
- %sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;
    - The above query will provide you with the unique launch sites.

Launch Site Names Begin with 'CCA'

- %sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;
- Below is the result of the sql query above, and it give you the 5 launch site names that begin with CCA.

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- %sql SELECT SUM("Payload_Mass__kg_") AS Total_Payload_Mass FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)';
- Present your query result with a short explanation here

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

- %sql SELECT AVG("Payload_Mass__kg_") AS Avg_Payload_Mass FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1';
- Present your query result

Avg_Payload_Mass
2928.4

First Successful Ground Landing Date

- %sql SELECT MIN("Date") AS First_Successful_Ground_Landing FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)';
- Present your query result

First_Successful_Ground_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- %sql SELECT DISTINCT "Booster_Version" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (drone ship)' AND "Payload_Mass__kg_" > 4000 AND "Payload_Mass__kg_" < 6000;

- Present your query result

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- %sql SELECT "Mission_Outcome", COUNT(*) AS Total FROM SPACEXTABLE GROUP BY "Mission_Outcome";
- Present your query result

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

Boosters Carried Maximum Payload

- %sql SELECT "Booster_Version", "Payload_Mass__kg_" FROM SPACEXTABLE WHERE "Payload_Mass__kg_" = (SELECT MAX("Payload_Mass__kg_") FROM SPACEXTABLE);

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

	Month	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

- %sql SELECT substr("Date", 6, 2) AS Month,"Landing_Outcome","Booster_Version","Launch_Site" FROM SPACEXTABLE WHERE "Landing_Outcome" LIKE '%Failure (drone ship)%' AND substr("Date", 1, 4) = '2015';

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

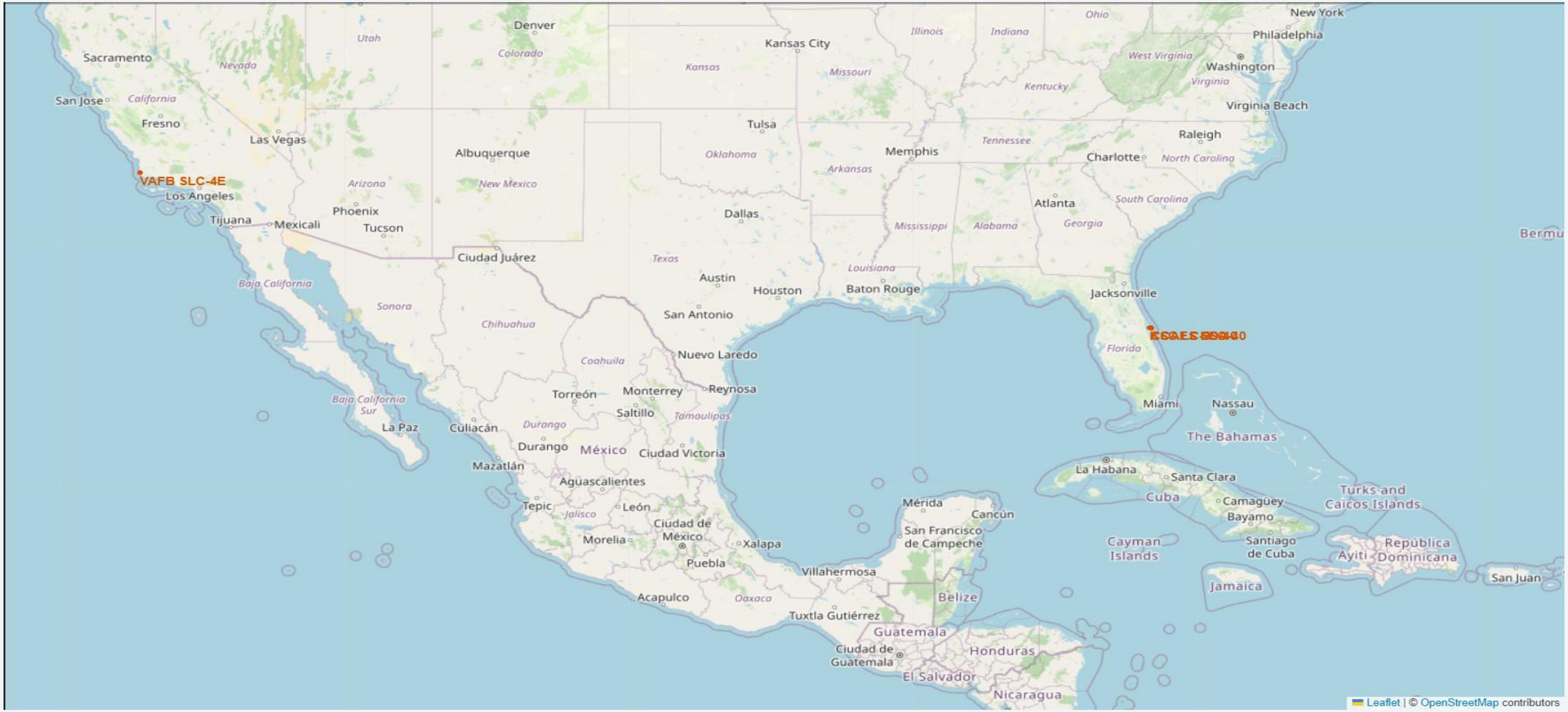
- %sql SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count FROM SPACEXTABLE WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY "Landing_Outcome" ORDER BY Outcome_Count DESC;

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

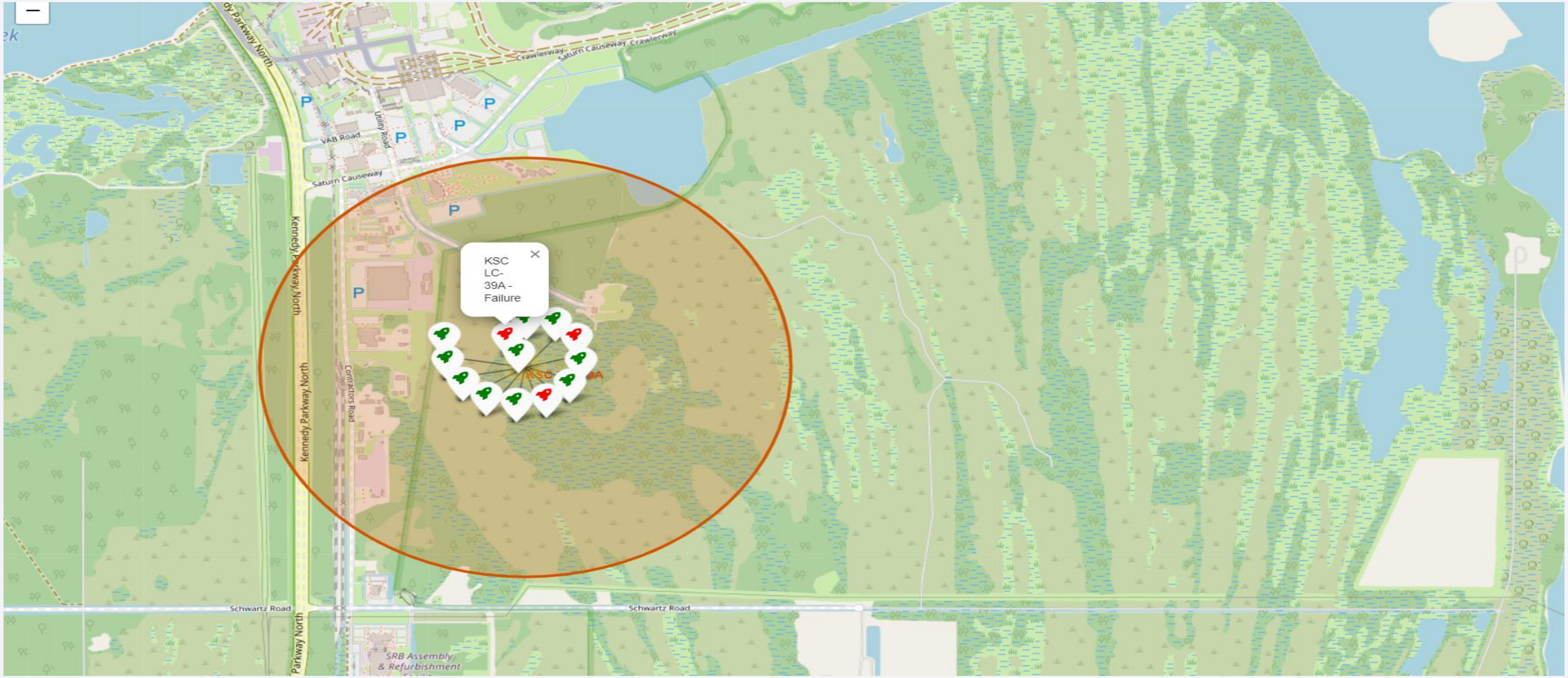
Section 3

Launch Sites Proximities Analysis

Launch Site Locations

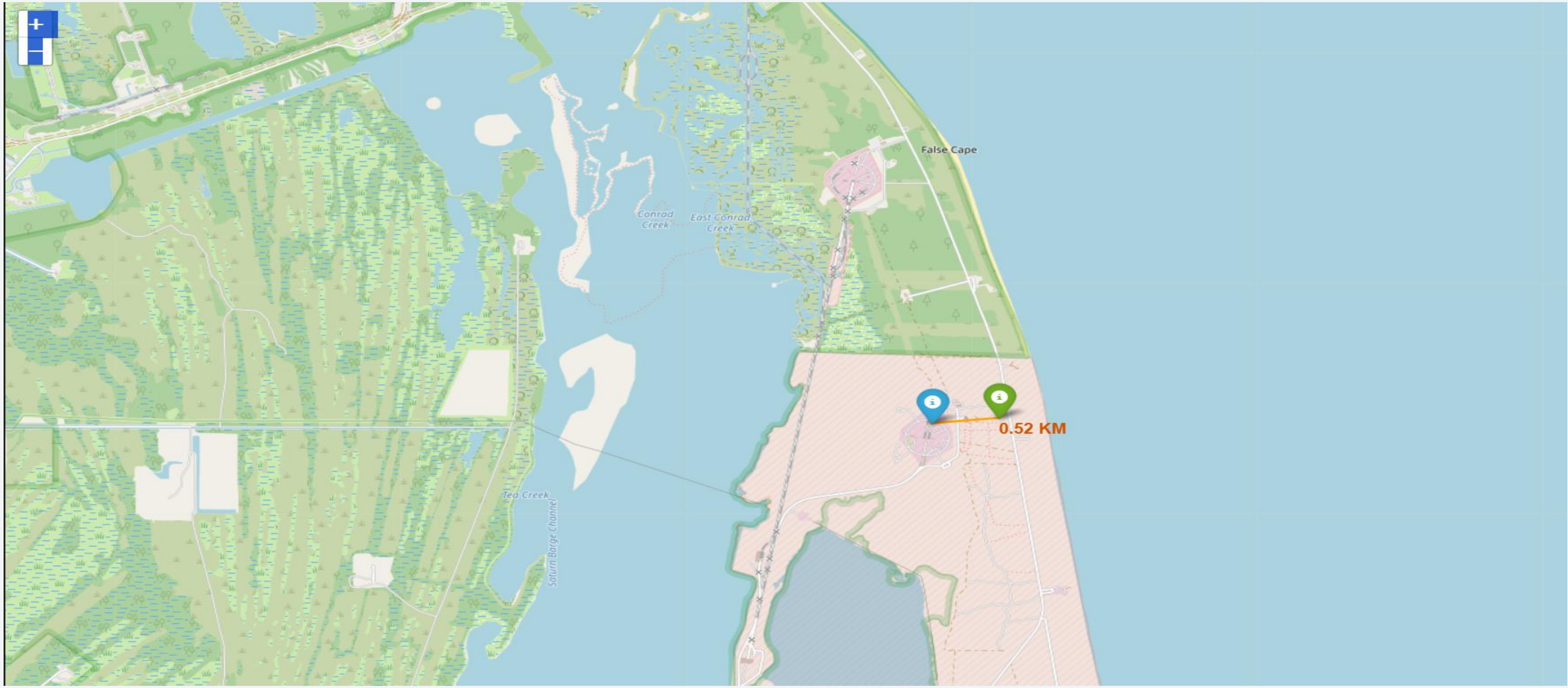


Marker Map



Launch site with markers for success and failures experienced in the site.

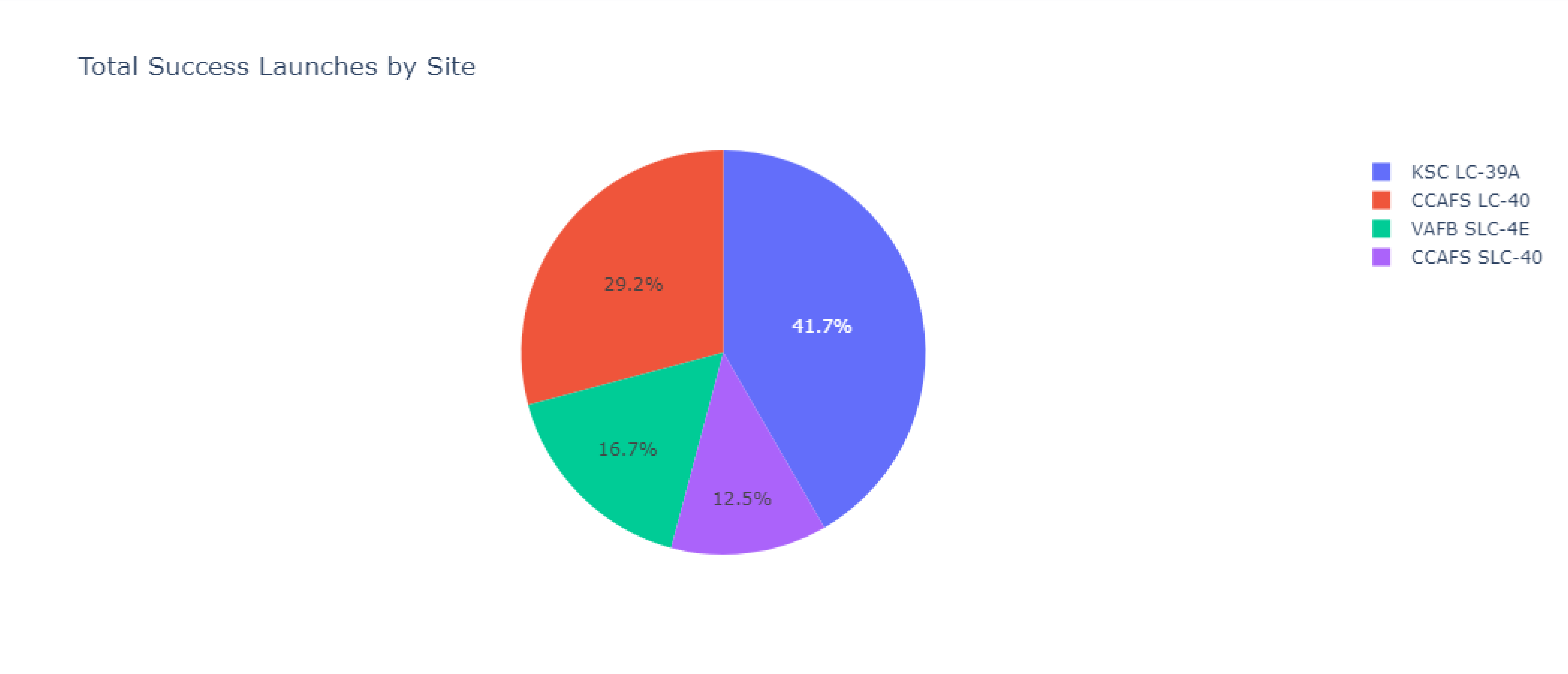
Distance Map



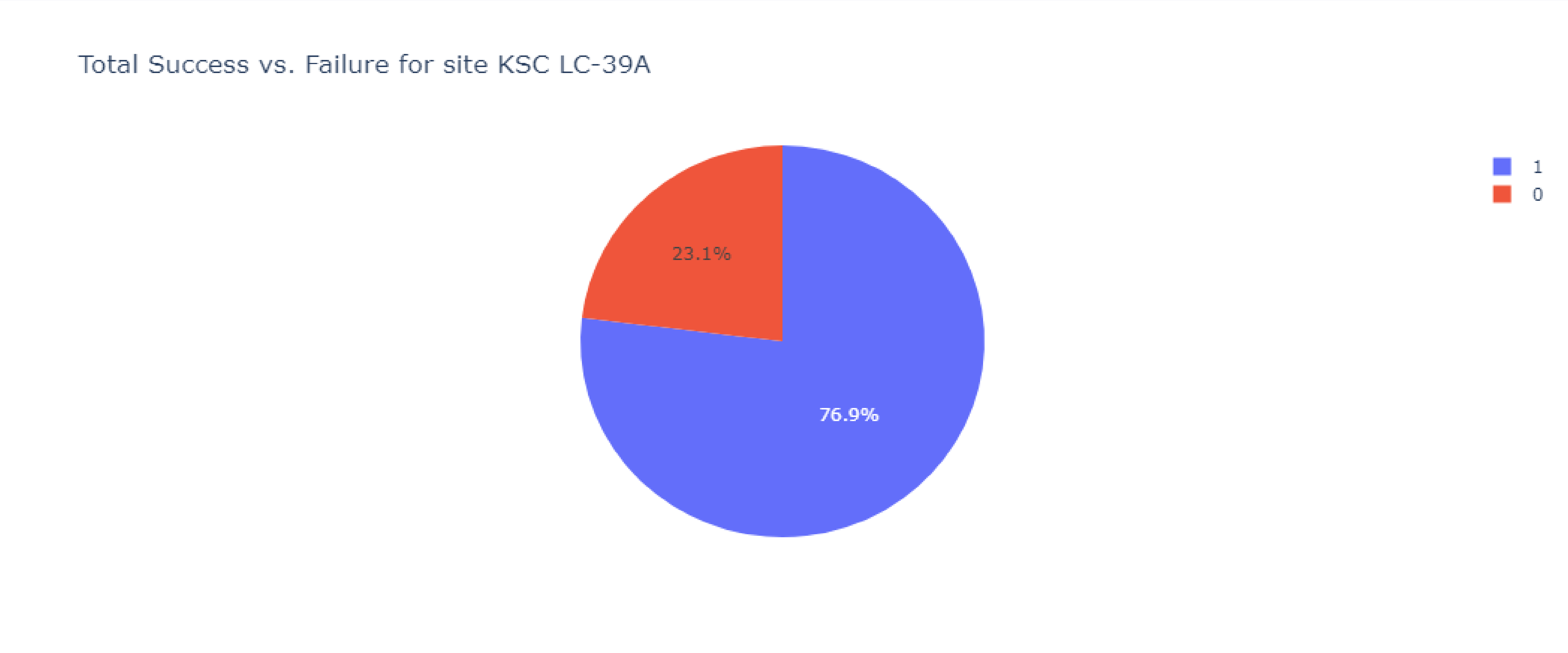
Section 4

Build a Dashboard with Plotly Dash

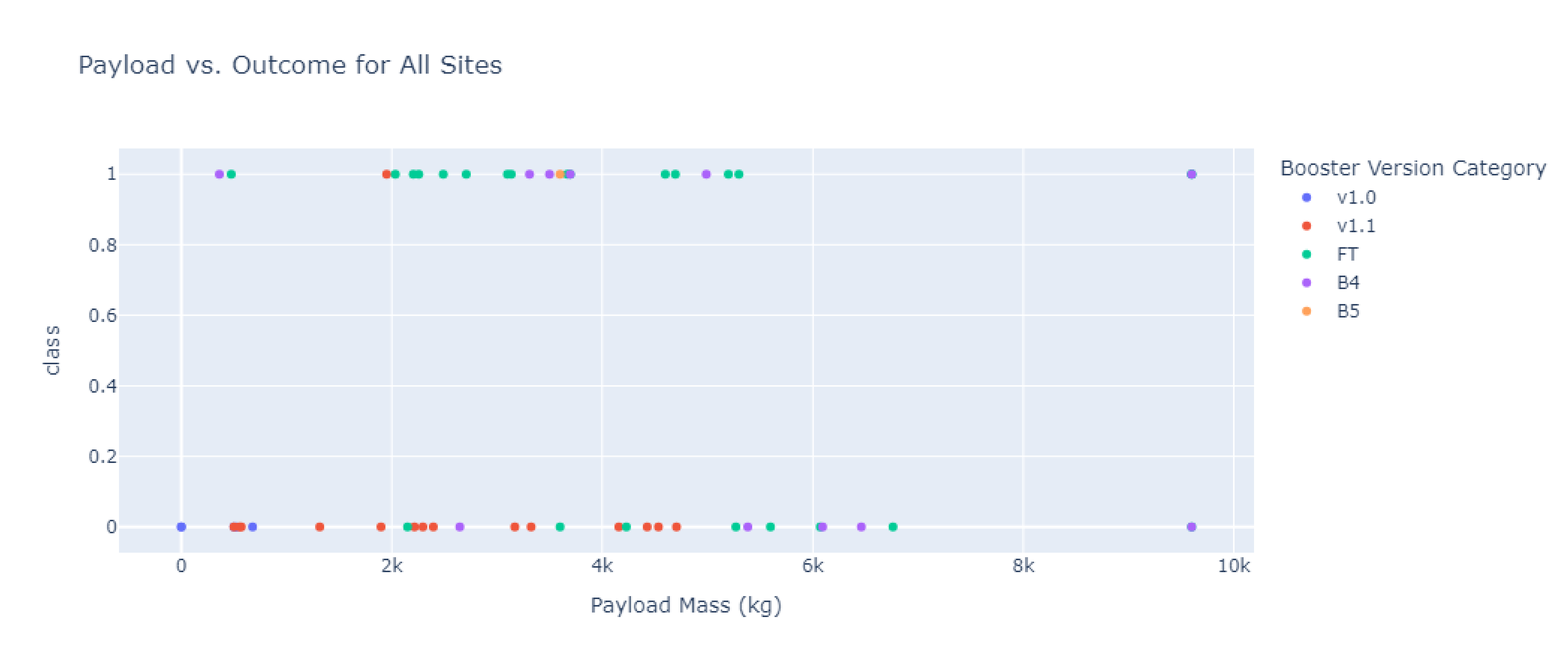
Pie Chart Success rate



Pie chart of the highest launch success



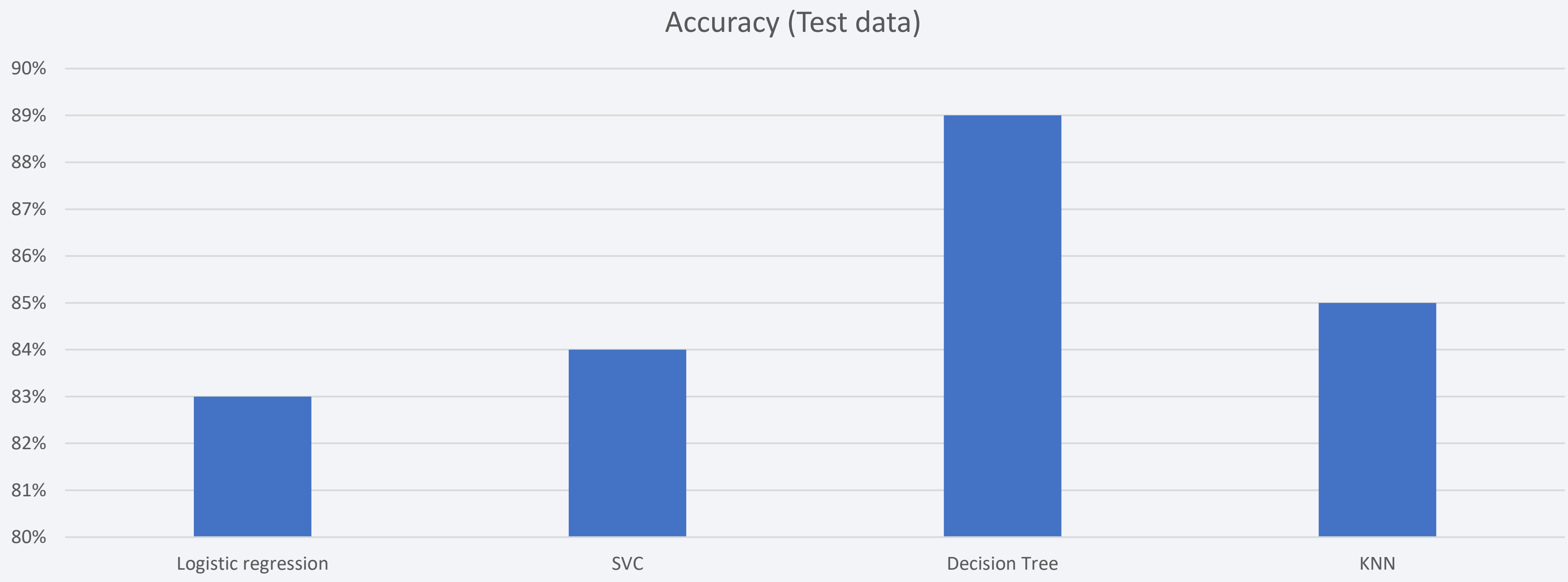
Payload vs Launch Outcome



Section 5

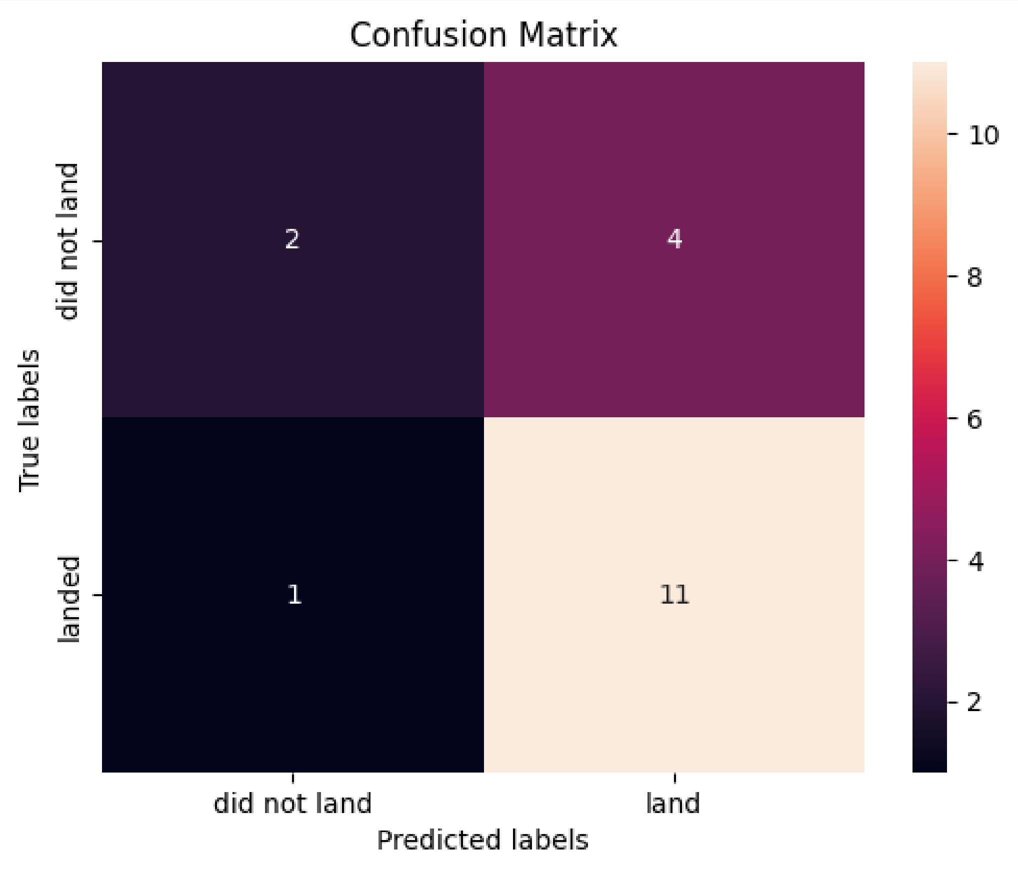
Predictive Analysis (Classification)

Classification Accuracy



Decision Tree has the highest accuracy amongst the other models.

Confusion Matrix



Decision Tree Matrix showing 11 True Positives.

Conclusions

- We can conclude that there have been more successes than failures in launch sites especially with the recommended payload mass as seen in the above slides.
- All models show an accuracy percentage of above 80% which is a good sign.
- The GitHub depository will provide the necessary needed python scripts which have the in-depth code used.

Appendix

```python
import pandas as pd
import dash
from dash import dcc, html
from dash.dependencies import Input, Output
import plotly.express as px
# Load SpaceX data
spacex_df = pd.read_csv("C:/Users/ZNgomti/Documents/spacex_launch_dash.csv")

min_payload = spacex_df['Payload Mass (kg)'].min()
max_payload = spacex_df['Payload Mass (kg)'].max()

app = dash.Dash(__name__)

# Dropdown options
site_options = [{'label': 'All Sites', 'value': 'ALL'}] + \
               [{'label': site, 'value': site} for site in spacex_df['Launch Site'].unique()]

app.layout = html.Div(children=[
    html.H1('SpaceX Launch Records Dashboard', style={'textAlign': 'center', 'color': '#503D36'}),

    # TASK 1: Launch Site Dropdown
    dcc.Dropdown(id='site-dropdown',
                 options=site_options,
                 value='ALL',
                 placeholder='Select a Launch Site here',
                 searchable=True),

    html.Br(),

    # TASK 2: Pie Chart Output
    html.Div(dcc.Graph(id='success-pie-chart')),

    html.Br(),

    html.P("Payload range (Kg):"),

    # TASK 3: Payload RangeSlider
    dcc.RangeSlider(id='payload-slider',
                    min=0, max=10000, step=1000,
                    marks={i: f'{i}' for i in range(0, 10001, 2500)},
                    value=[min_payload, max_payload]),

    # TASK 4: Scatter Plot Output
    html.Div(dcc.Graph(id='success-payload-scatter-chart'))
])

# TASK 2: Callback for Pie Chart
@app.callback(Output(component_id='success-pie-chart', component_property='figure'),
              Input(component_id='site-dropdown', component_property='value'))
def get_pie_chart(entered_site):
    if entered_site == 'ALL':
        fig = px.pie(spacex_df, values='class', names='Launch Site',
                     title='Total Success Launches by Site')
    else:
        filtered_df = spacex_df[spacex_df['Launch Site'] == entered_site]
        class_counts = filtered_df['class'].value_counts().reset_index()
        class_counts.columns = ['class', 'count']
        fig = px.pie(class_counts, values='count', names='class',
                     title=f'Total Success vs. Failure for site {entered_site}')
    return fig

# TASK 4: Callback for Scatter Plot
@app.callback(Output(component_id='success-payload-scatter-chart', component_property='figure'),
              [Input(component_id='site-dropdown', component_property='value'),
               Input(component_id="payload-slider", component_property="value")])
def get_scatter_plot(entered_site, payload_range):
    filtered_df = spacex_df[(spacex_df['Payload Mass (kg)'] >= payload_range[0]) &
                            (spacex_df['Payload Mass (kg)'] <= payload_range[1])]

    if entered_site == 'ALL':
        fig = px.scatter(filtered_df, x='Payload Mass (kg)', y='class',
                         color='Booster Version Category',
                         title='Payload vs. Outcome for All Sites')
    else:
        site_df = filtered_df[filtered_df['Launch Site'] == entered_site]
        fig = px.scatter(site_df, x='Payload Mass (kg)', y='class',
                         color='Booster Version Category',
                         title=f'Payload vs. Outcome for site {entered_site}')
    return fig

if __name__ == '__main__':
    app.run(debug=True)
```

Thank you!