IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Bongane
Mutesse

Outline

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

Executive Summary

- **Summary of Methodologies:**

This project followed the following steps:

- Data Collection
- Data Wrangling
- Exploratory Data Analysis
- Interactive Visual Analytics
- Predictive Analysis (Classification)

- **Summary of Results:**

The following outputs and visualizations were produced:

1. Exploratory Data Analysis (EDA) results
2. Geospatial analytics
3. Interactive dashboard
4. Predictive analysis of classification models

Introduction

- SpaceX launches Falcon 9 rockets at a cost of around $62m. This is considerably cheaper than other providers (which usually costs above $165m).
- The cheaper launches from SpaceX are because SpaceX can land, and then re-use the first stage of the rocket.
- In order to achieve the cost reduction in Space launches we need to make predictions on whether the first stage will land successfully, we can then determine the cost of a launch, and use this information to assess whether or not an alternate company should bid against SpaceX for a rocket launch.
- This project aim to use Data Analytics to predict if the SpaceX Falcon 9 first stage landing will be successful.

Section 1

Methodology

Methodology

1. Data Collection
   - Made GET requests to the SpaceX REST API
   - Web Scraping
2. Data Wrangling
   - Used the .fillna() method to remove NaN values
   - Used the .value_counts() method to determine the following:
     - Number of launches on each site
     - Number and occurrence of each orbit
     - Number and occurrence of mission outcome per orbit type
     - Created a landing binary outcome label that shows the following:

       0 when the booster did not land successfully

       1 when the booster did land successfully
3. Exploratory Data Analysis
   - Used SQL queries to manipulate and evaluate the SpaceX dataset
   - Used Pandas and Matplotlib to visualize relationships between variables, and determine patterns

4. Interactive Visual Analytics
   - Geospatial analytics using Folium
   - Created an interactive dashboard using Plotly Dash

5. Data Modelling and Evaluation
   - Used Scikit-Learn to:
     - Pre-process (or standardize) the data
     - Split the data into training and testing data using train_test_split()
     - Trained different classification models
     - Found hyperparameters using GridSearchCV()
   - Plotted confusion matrices for each classification model
   - Assessed the accuracy of each classification model

Data Collection

- Data collection process involved a combination of API requests from SpaceX REST API (https://api.spacexdata.com/v4/rockets/) and Web Scraping data from a table in SpaceX's Wikipedia entry(https://en.wikipedia.org/wiki/List_of_Falcon/_9/_and_Falcon_Heavy_launches).
- The two data collection methods were used in order to get complete information about the launches for a more detailed analysis.
- Data Columns were obtained by using SpaceX REST API: FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude,
- Latitude Data Columns were obtained through Wikipedia Web Scraping: Flight No., Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Version Booster, Booster landing, Date, Time

Data Collection – SpaceX API

Requesting rocket launch data from SpaceX API

Decoding the response content using .json() and turning it into a dataframe using .json_normalize()

Requesting needed information about the launches from SpaceX API by applying custom functions

Constructing data we have obtained into a dictionary

Creating a dataframe from the dictionary

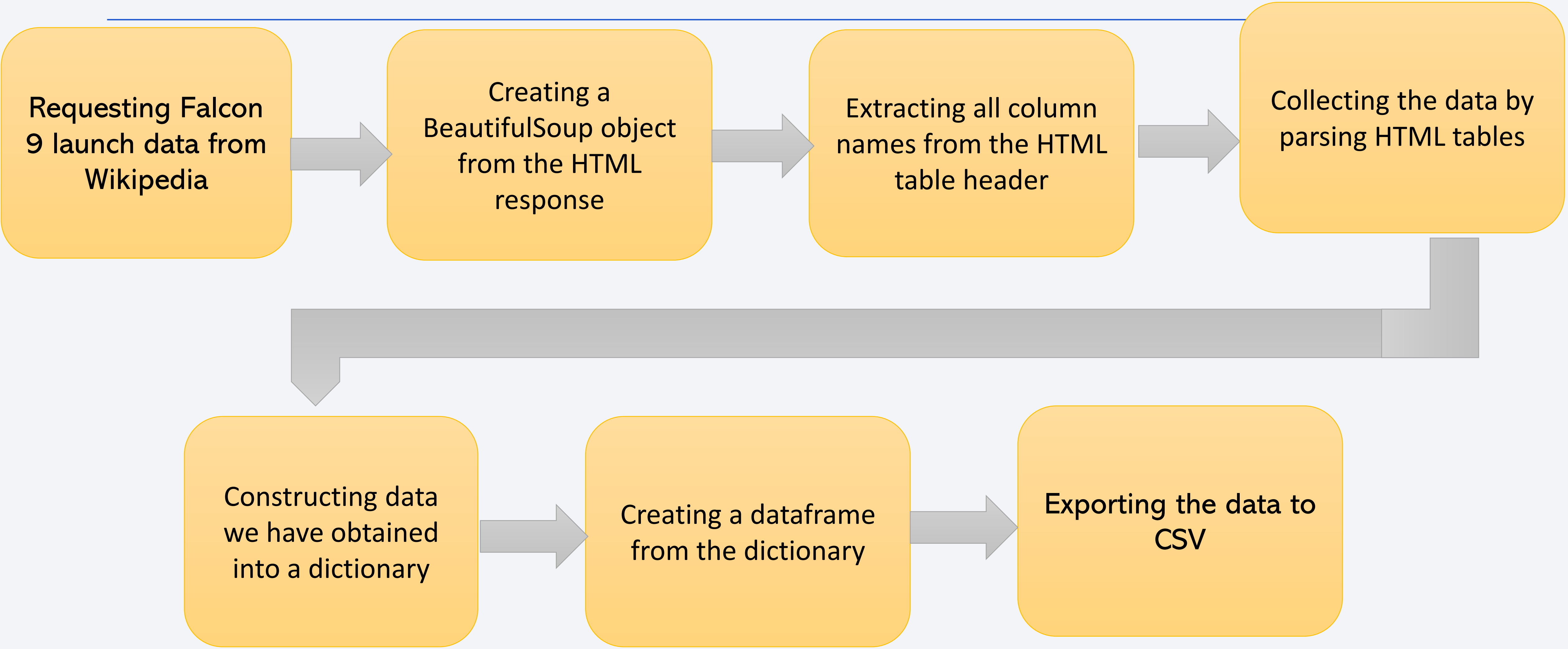
Filtering the dataframe to only include Falcon 9 launches

Replacing missing values of Payload Mass column with calculated .mean() for this column

Exporting the data to CSV

GitHub URL: https://github.com/BMutesse/testrepo/blob/main/jupyter-labs-spacex-data-collection-api.ipynbas an external reference and peer-review purpose

Data Collection - Scraping



GitHub URL: https://github.com/BMutesse/testrepo/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

- In the data set, there are several different cases where the booster did not land successfully.
- Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean.
- True RTLS means the mission outcome was successfully landed to a ground pad while False RTLS means the mission outcome was unsuccessfully landed to a ground pad.
- True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.
- We mainly convert those outcomes into Training Labels with 1 for booster successfully landing, and 0 if it was unsuccessful.

Perform exploratory Data Analysis and determine Training Labels

Calculate the number of launches on each site

Calculate the number and occurrence of each orbit

Calculate the number and occurrence of mission outcome per orbit type

Create a landing outcome label from Outcome column

Export the data to CSV

GitHub URL: https://github.com/BMutesse/testrepo/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

The following Charts were plotted:

- Flight Number vs. Payload Mass,
- Flight Number vs. Launch Site,
- Payload Mass vs. Launch Site,
- Orbit Type vs. Success Rate,
- Flight Number vs. Orbit Type,
- Payload Mass vs Orbit Type and

- The Success Rate Yearly Trend Scatter plots showing the relationship between variables were also plotted.
- If a relationship exists, they could be used in machine learning model.
- Bar charts showing comparisons among discrete categories. The goal was to show the relationship between the specific categories being compared and a measured value.
- Line charts showing trends in data over time (time series).

EDA with SQL

The following SQL queries were performed:

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string ‘CCA’
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date when the first successful landing outcome in ground pad was achieved
- Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster versions which have carried the maximum payload mass
- Listing the failed landing outcomes in drone ship, their booster versions and launch site names for the months in year 2015
- Ranking the count of landing outcomes between the date 2010-06-04 and 2017-03-20 in descending order

Build an Interactive Map with Folium

The following steps were taken to visualize the launch data on an interactive map:

1. Mark all launch sites on a map
   - Initialise the map using a Folium Map object
   - Add a folium.Circle and folium.Marker for each launch site on the launch map.
2. Mark the success/failed launches for each site on a map
   - As many launches have the same coordinates, it makes sense to cluster them together.
   - Before clustering them, assign a marker colour of successful (class = 1) as green, and failed (class = 0) as red.
   - To put the launches into clusters, for each launch, add a folium.Marker to the MarkerCluster() object.
   - Create an icon as a text label, assigning the icon_color as the marker_colour determined previously.
3. Calculate the distances between a launch site to its proximities
   - To explore the proximities of launch sites, calculations of distances between points can be made using the Lat and Long values.
   - After marking a point using the Lat and Long values, create a folium.Marker object to show the distance.
   - To display the distance line between two points, draw a folium.PolyLine and add this to the map.

GitHub URL: https://github.com/BMutesse/testrepo/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

The following plots were added to a Plotly Dash dashboard to have an interactive visualisation of the data:

1. Pie chart (px.pie()) showing the total successful launches per site
   - This makes it clear to see which sites are most successful
   - The chart could also be filtered (using a dcc.Dropdown() object) to see the success/failure ratio for an individual site

2. Scatter graph (px.scatter()) to show the correlation between outcome (success or not) and payload mass (kg)
   - This could be filtered (using a RangeSlider() object) by ranges of payload masses
   - It could also be filtered by booster version

Predictive Analysis (Classification)

Model Development

- To prepare the dataset for model development:
  - Perform necessary data transformations (standardise and pre-process)
  - Split data into training and test data sets, using `train_test_split()`
  - Decide which type of machine learning algorithms are most appropriate
- For each chosen algorithm:
  - Create a `GridSearchCV` object and a dictionary of parameters
  - Fit the object to the parameters
  - Use the training data set to train the model

Model Evaluation

- For each chosen algorithm:
  - Using the output GridSearchCV object:
    - Check the tuned hyperparameters (best_params_)
    - Check the accuracy (score and best_score_)
    - Plot and examine the Confusion Matrix

Finding the Best Classification Model

- Review the accuracy scores for all chosen algorithms
- The model with the highest accuracy score is determined as the best performing model

Results

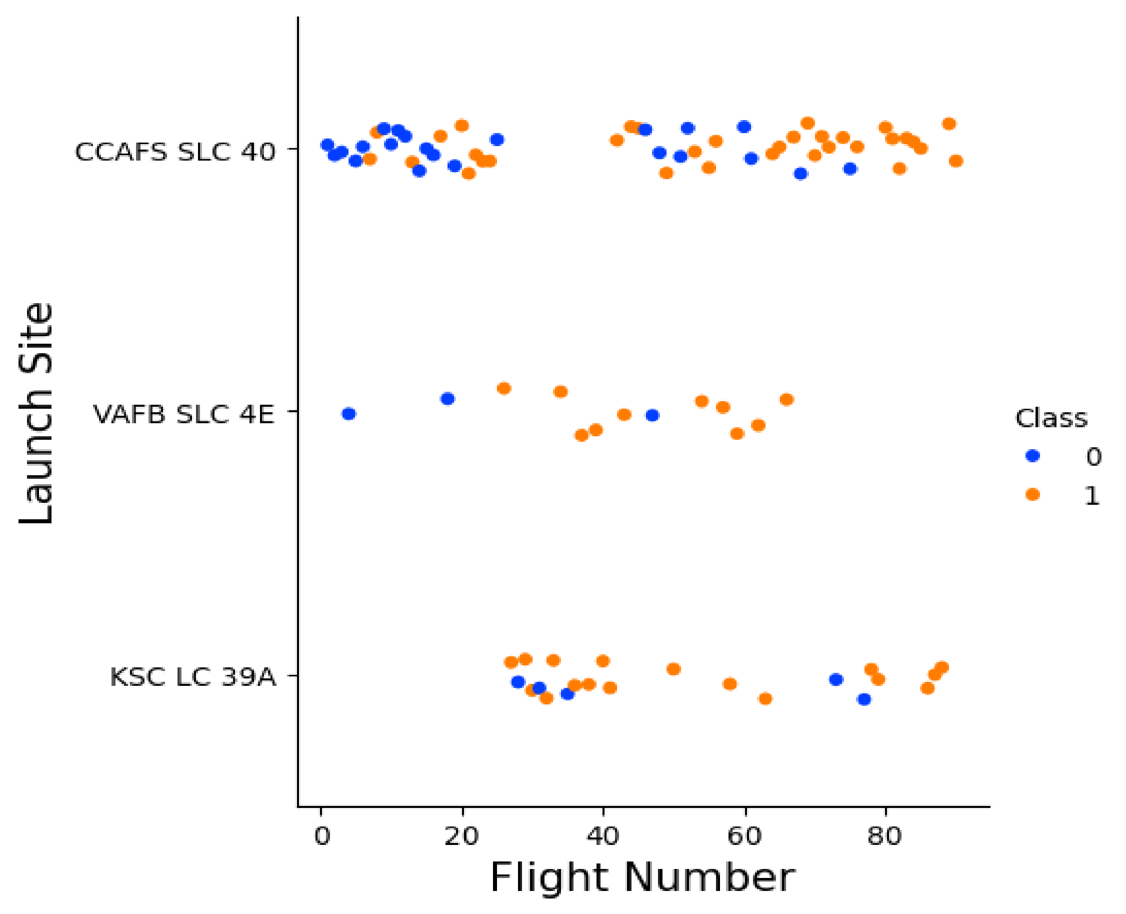
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

Scatter plot of Flight Number vs. Launch Site

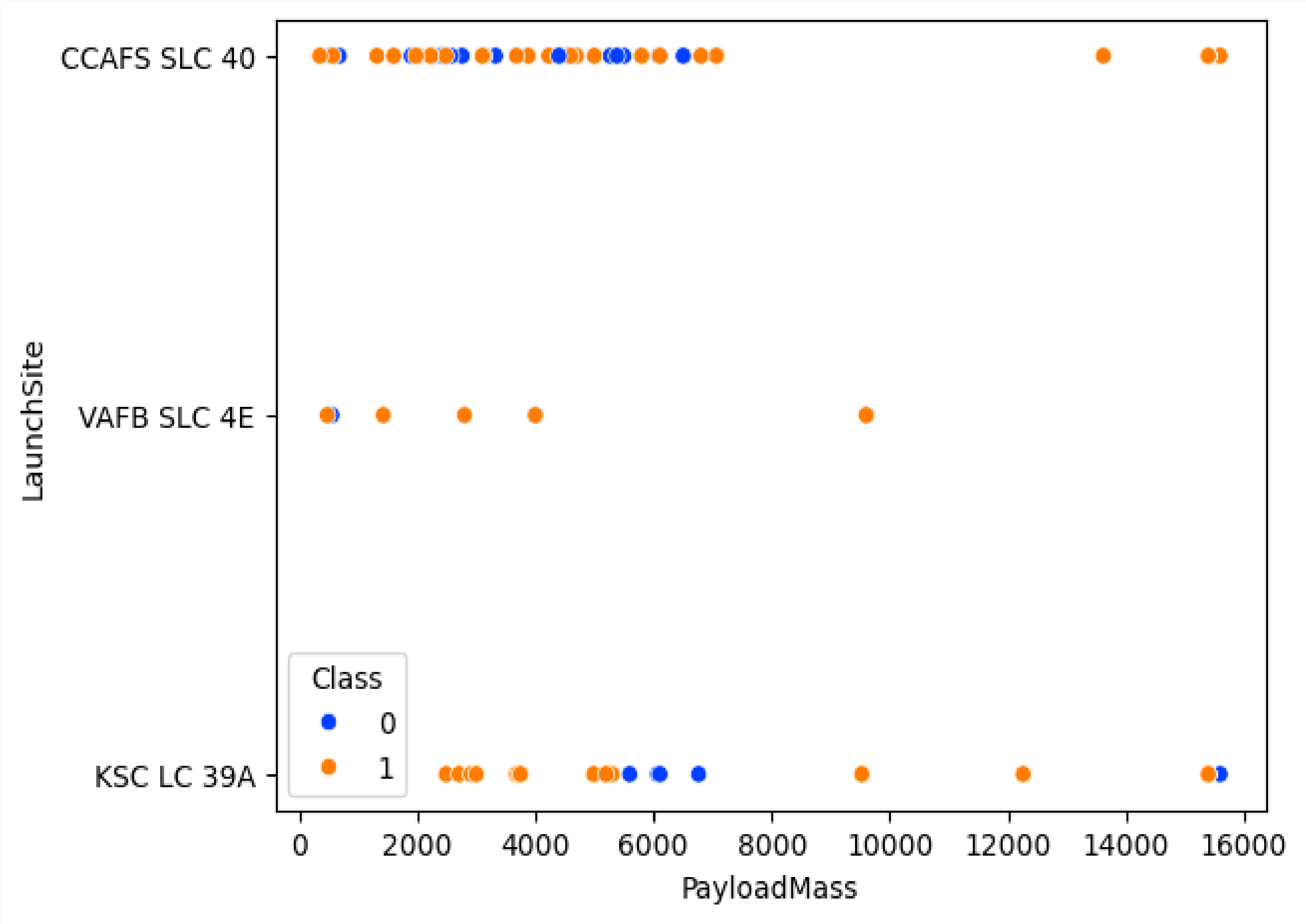


The scatter plot of Launch Site vs. Flight Number shows that:

- As the number of flights increases, the rate of success at a launch site increases.
- Most of the early flights (flight numbers < 30) were launched from CCAFS SLC 40, and were generally unsuccessful.
- The flights from VAFB SLC 4E also show this trend, that earlier flights were less successful.
- No early flights were launched from KSC LC 39A, so the launches from this site are more successful.
- Above a flight number of around 30, there are significantly more successful landings (Class = 1).

Payload vs. Launch Site

Scatter plot of Payload vs. Launch Site

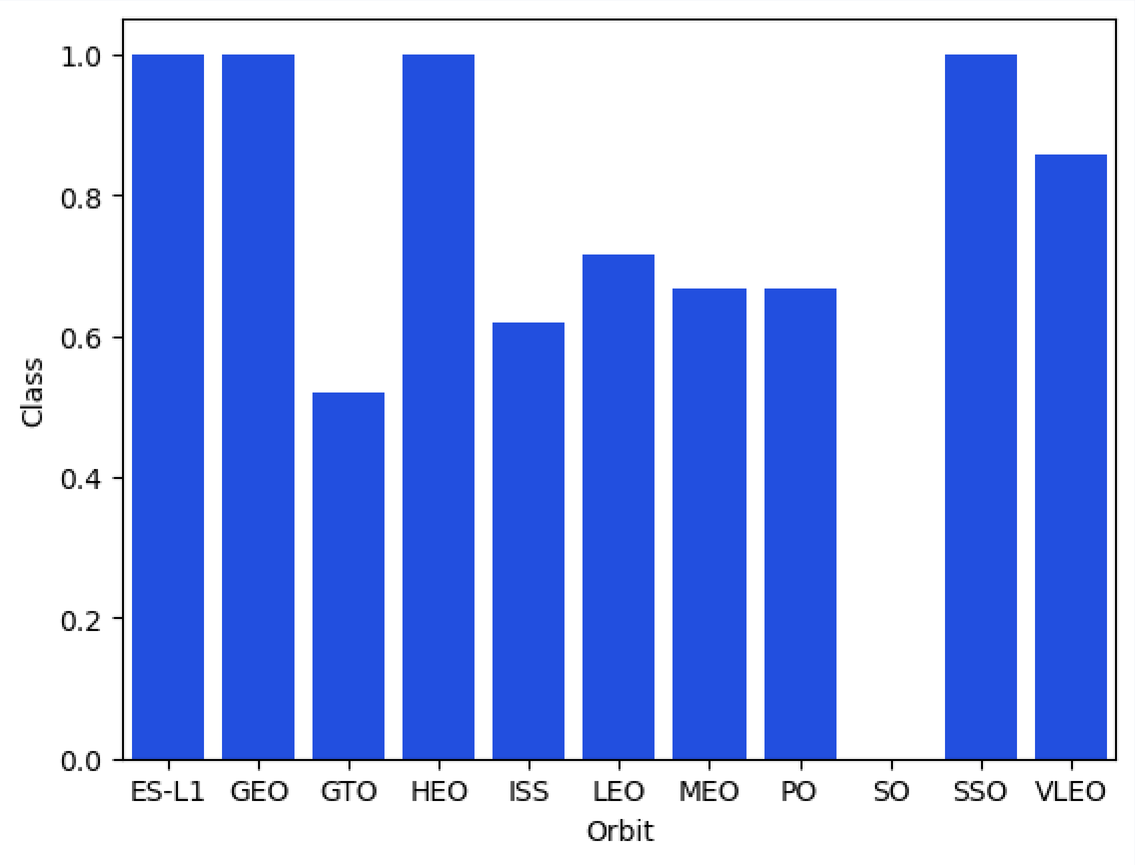


The scatter plot of Launch Site vs. Payload Mass shows that:

- Above a payload mass of around 7000 kg, there are very few unsuccessful landings, but there is also far less data for these heavier launches.
- There is no clear correlation between payload mass and success rate for a given launch site.
- All sites launched a variety of payload masses, with most of the launches from CCAFS SLC 40 being comparatively lighter payloads (with some outliers).

Success Rate vs. Orbit Type

Bar chart for the success rate of each orbit type



The bar chart of Success Rate vs. Orbit Type shows that the following orbits have the highest (100%) success rate:

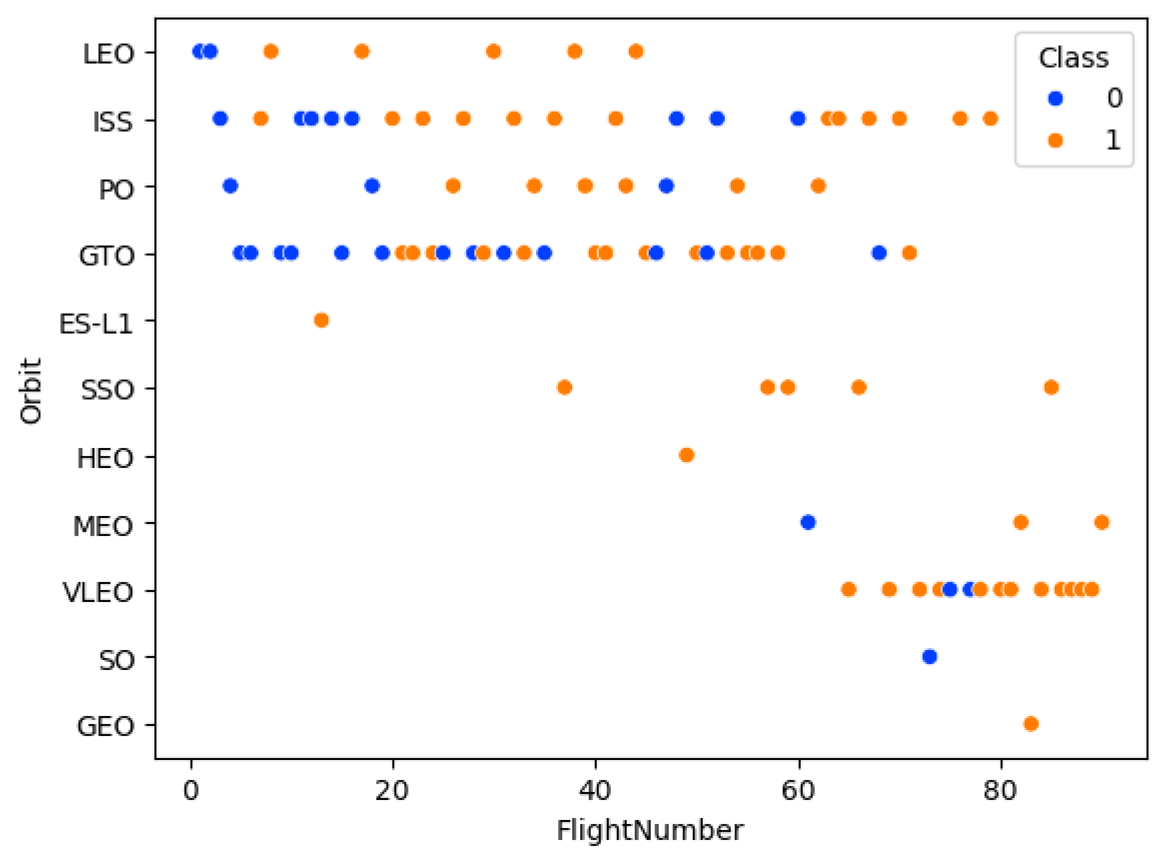
- ES-L1 (Earth-Sun First Lagrangian Point)
- GEO (Geostationary Orbit)
- HEO (High Earth Orbit)
- SSO (Sun-synchronous Orbit)

The orbit with the lowest (0%) success rate is:

- SO (Heliocentric Orbit)

Flight Number vs. Orbit Type

Scatter plot of Flight number vs. Orbit type

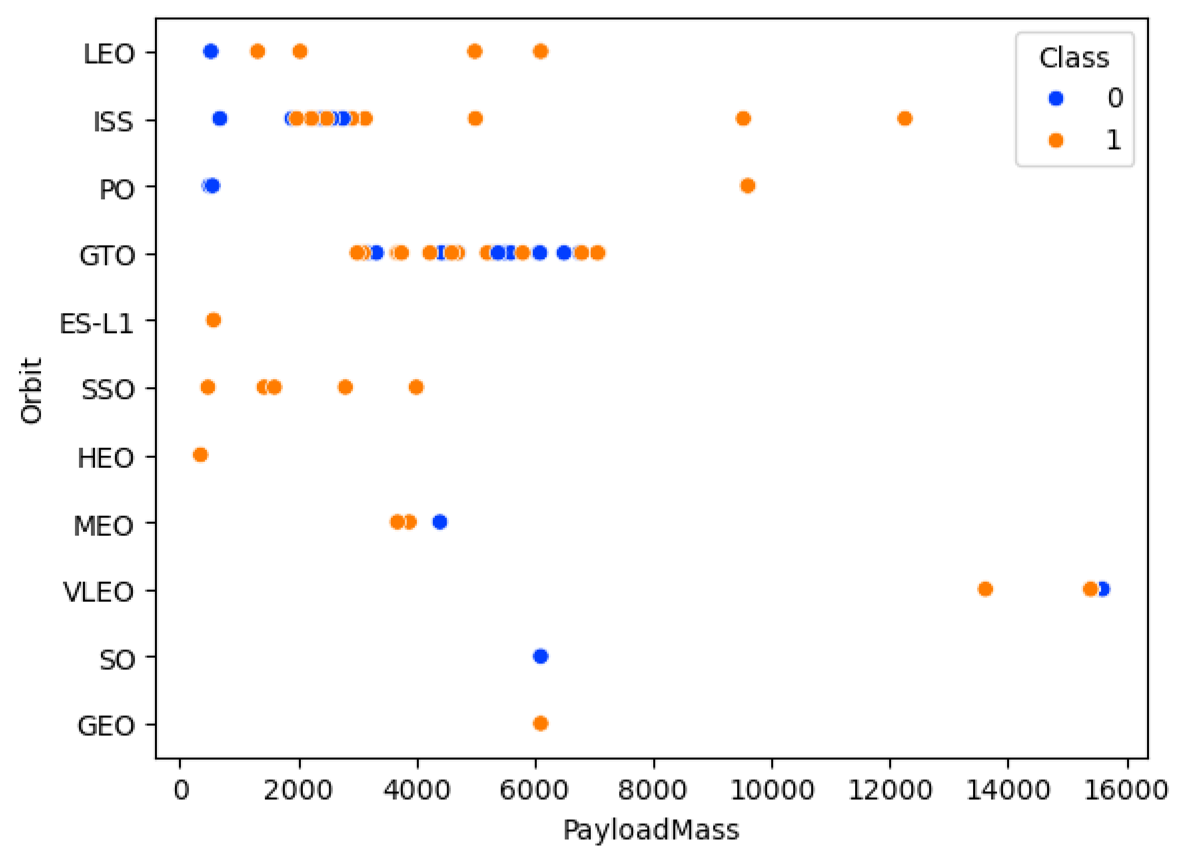


This scatter plot of Orbit Type vs. Flight number shows a few useful points that the previous plots did not, such as:

- The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
- The 100% success rate in SSO is more impressive, with 5 successful flights.
- There is little relationship between Flight Number and Success Rate for GTO.
- Generally, as Flight Number increases, the success rate increases. This is most extreme for LEO, where unsuccessful landings only occurred for the low flight numbers (early launches).

Payload vs. Orbit Type

Scatter plot of payload vs. orbit type

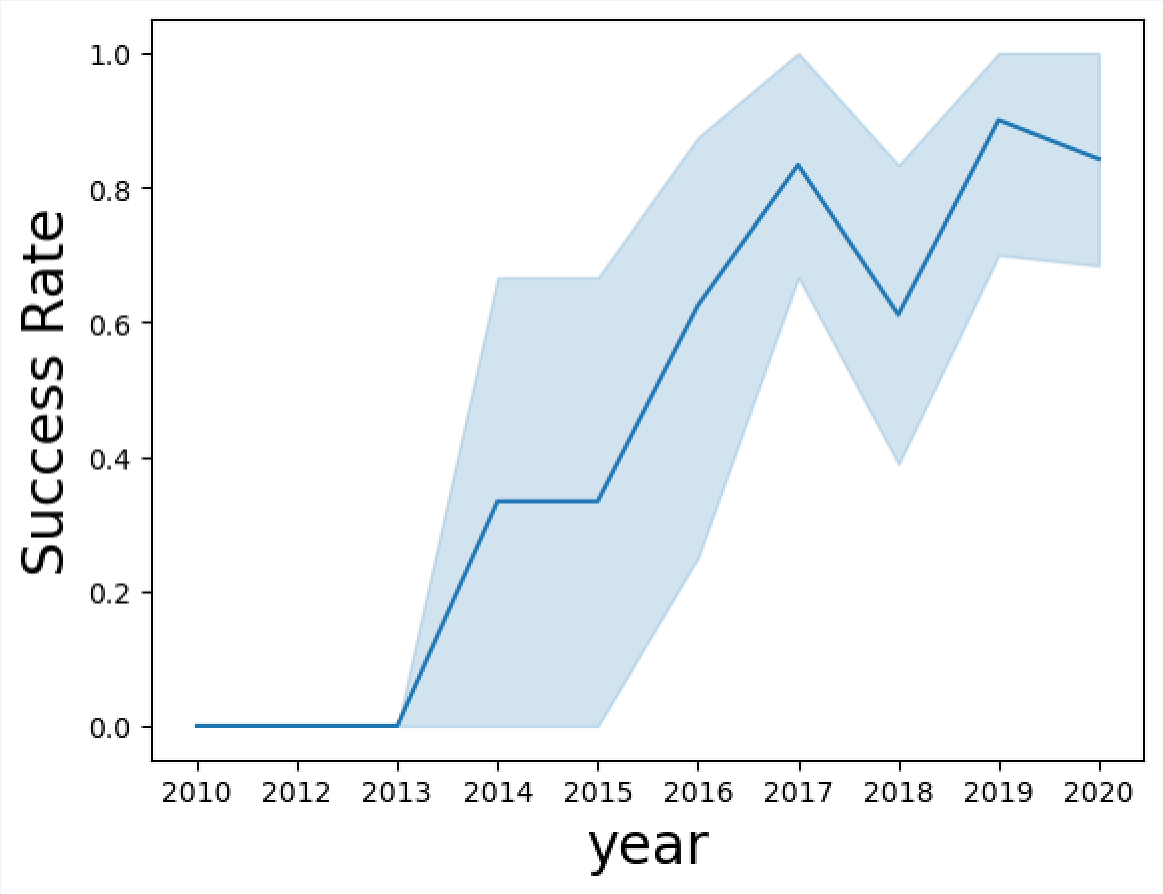


This scatter plot of Orbit Type vs. Payload Mass shows that:

- The following orbit types have more success with heavy payloads:
  - PO (although the number of data points is small)
  - ISS
  - LEO
- For GTO, the relationship between payload mass and success rate is unclear.
- VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.

Launch Success Yearly Trend

Line chart of yearly average success rate



The line chart of yearly average success rate shows that:

- Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
- After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
- After 2016, there was always a greater than 50% chance of success.

All Launch Site Names

```
%sql SELECT UNIQUE(LAUNCH_SITE) FROM SPACEXTBL;
```

Launch_site:

CCAFS LC – 40
CCAFS SLC – 40
KSC LC – 39A
VAFB SLC – 4E

The word UNIQUE returns only unique values from the LAUNCH_SITE column of the SPACEXTBL table.

Launch Site Names Begin with 'CCA'

```
%sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

Launch_site:

CCAFS LC – 40
CCAFS LC – 40
CCAFS LC – 40
CCAFS LC – 40
CCAFS LC – 40

LIMIT 5 fetches only 5 records, and the LIKE keyword is used with the wild card 'CCA%' to retrieve string values beginning with 'CCA'.

Total Payload Mass

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL\
WHERE CUSTOMER = 'NASA (CRS)';
```

Total_payload_mass: 45596

- The SUM keyword is used to calculate the total of the LAUNCH column, and the SUM keyword (and the associated condition) filters the results to only boosters from NASA (CRS).

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVERAGE_PAYLOAD_MASS FROM SPACEXTBL\
WHERE BOOSTER_VERSION= 'F9 v1.1';
```

Aver_payload_mass:
2928

- The AVG keyword is used to calculate the average of the PAYLOAD_MASS__KG_ column, and the WHERE keyword (and the associated condition) filters the results to only the F9 v1.1 booster version

First Successful Ground Landing Date

```
%sql SELECT MIN(DATE) AS FIRST_SUCCESSFUL_GROUND_LANDING FROM SPACEXTBL\
WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

fist_successful_ground_landing:
2015-12-22

- The MIN keyword is used to calculate the minimum of the DATE column, i.e. the first date, and the WHERE keyword (and the associated condition) filters the results to only the successful ground pad landings.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL \
WHERE (LANDING__OUTCOME = 'Success (drone ship)') AND (PAYLOAD_MASS__KG_
BETWEEN 4000 AND 6000);
```

Booster_version:

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

- The WHERE keyword is used to filter the results to include only those that satisfy both conditions in the brackets (as the AND keyword is also used). The BETWEEN keyword allows for 4000 < x < 6000 values to be selected.

Total Number of Successful and Failure Mission Outcomes

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

Mission_outcome	Total
Failure (inflight):	1
Success:	99
Success (payload status unclear)	1

- The COUNT keyword is used to calculate the total number of mission outcomes, and the GROUPBY keyword is also used to group these results by the type of mission outcome.

Boosters Carried Maximum Payload

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

Booster_version:

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

- A subquery is used here. The SELECT statement within the brackets finds the maximum payload, and this value is used in the WHERE condition. The DISTINCT keyword is then used to retrieve only distinct /unique booster versions.

2015 Launch Records

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL \
    WHERE (LANDING__OUTCOME = 'Failure (drone ship)') AND (EXTRACT(YEAR FROM
DATE) = '2015');
```

Booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

The WHERE keyword is used to filter the results for only failed landing outcomes, AND only for the year of 2015.

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

```
%sql SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS \
TOTAL_NUMBER FROM SPACEXTBL \
    WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
    GROUP BY LANDING__OUTCOME \
    ORDER BY TOTAL_NUMBER DESC;
```

landing_outcome	total_number
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (paratute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

- The WHERE keyword is used with the BETWEEN keyword to filter the results to dates only within those specified. The results are then grouped and ordered, using the keywords GROUP BY and ORDER BY, respectively, where DESC is used to specify the descending order.

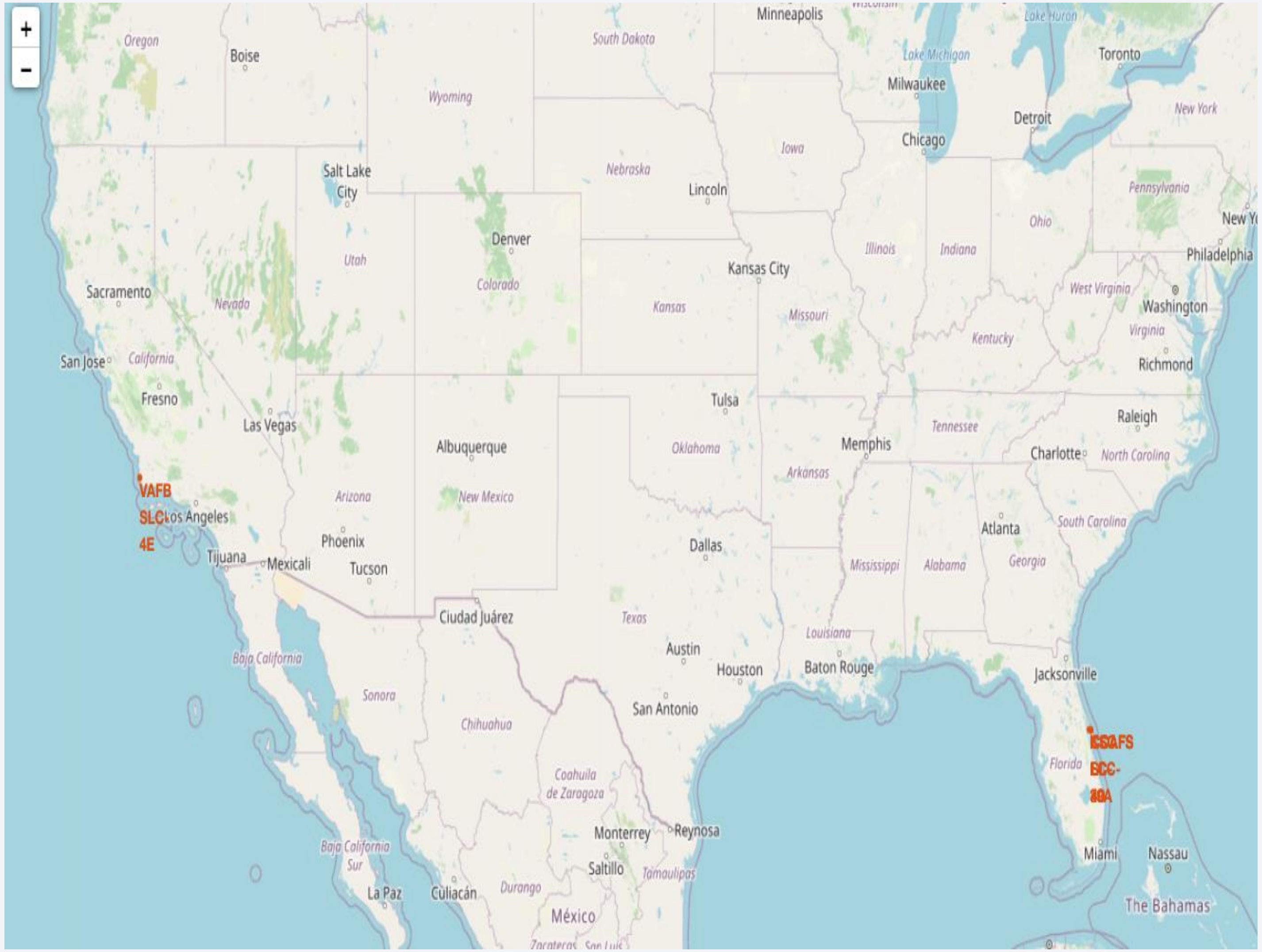
Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

Discussion:

- Most of Launch sites are in proximity to the Equator line. The Earth is moving faster at the equator than any other place on the surface of the Earth. Anything on the surface of the Earth at the equator is already moving at 1670 km/hour. If a ship is launched from the equator it goes up into space, and it is also moving around the Earth at the same speed it was moving before launching. This is because of inertia. This speed will help the spacecraft keep up a good enough speed to stay in orbit.
- All launch sites are in very close proximity to the coast. Launching rockets towards the ocean minimizes the risk of having any debris dropping or exploding near people.



<Folium Map Screenshot 2>

Discussion:

- From the colour-labeled markers we should be able to easily identify which launch sites have relatively high success rates.

- Green Marker = Successful Launch

- Red Marker = Failed Launch

- Launch Site KSC LC-39A has a very high Success Rate.

<Folium Map Screenshot 3>

Discussion:

- From the visual analysis of the launch site KSC LC-39A we can clearly see that it is:

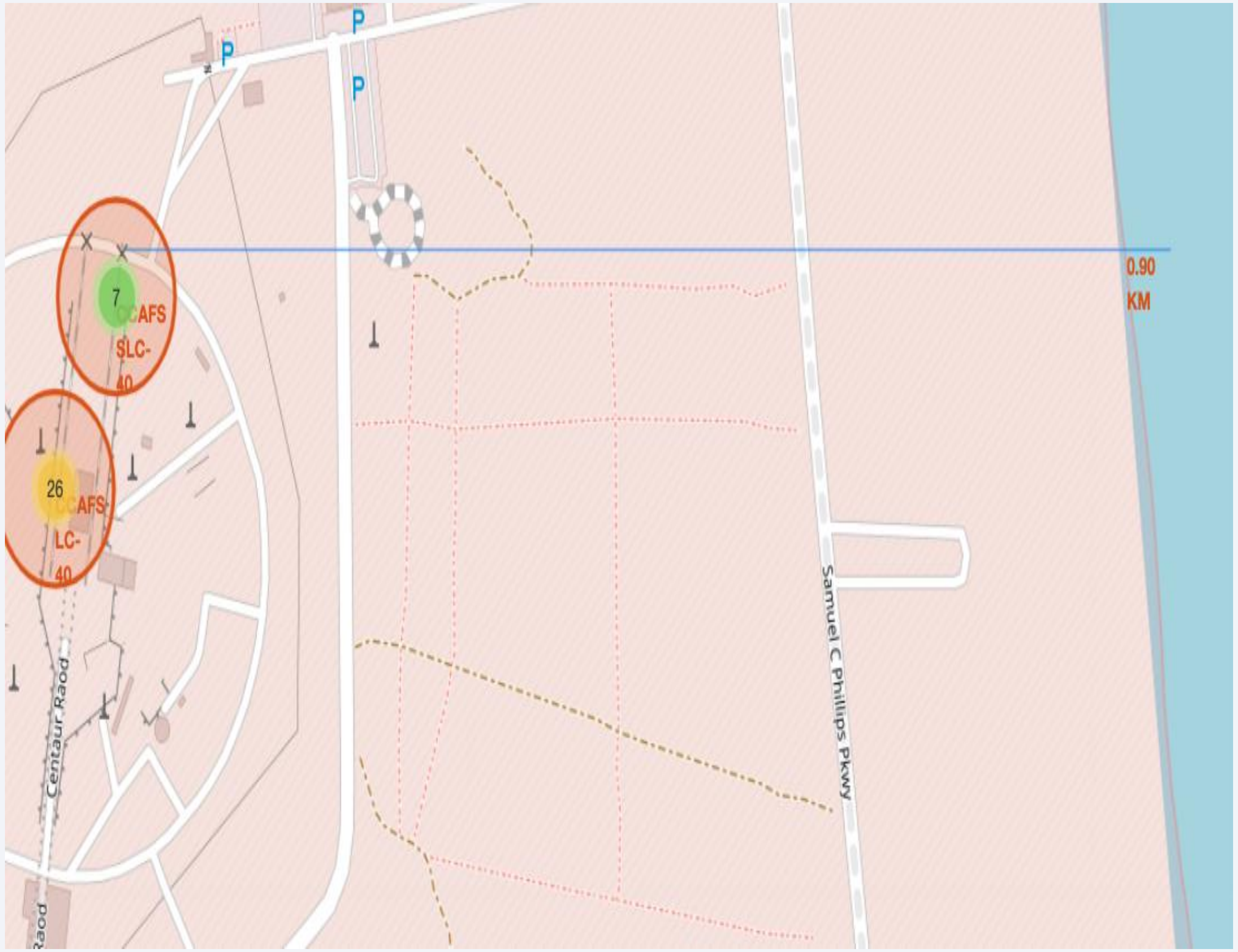
- relatively close to railway (15.23 km)

- relatively close to highway (20.28 km)

- relatively close to coastline (14.99 km)

- relatively close to its closest city, Titusville (16.32 km).

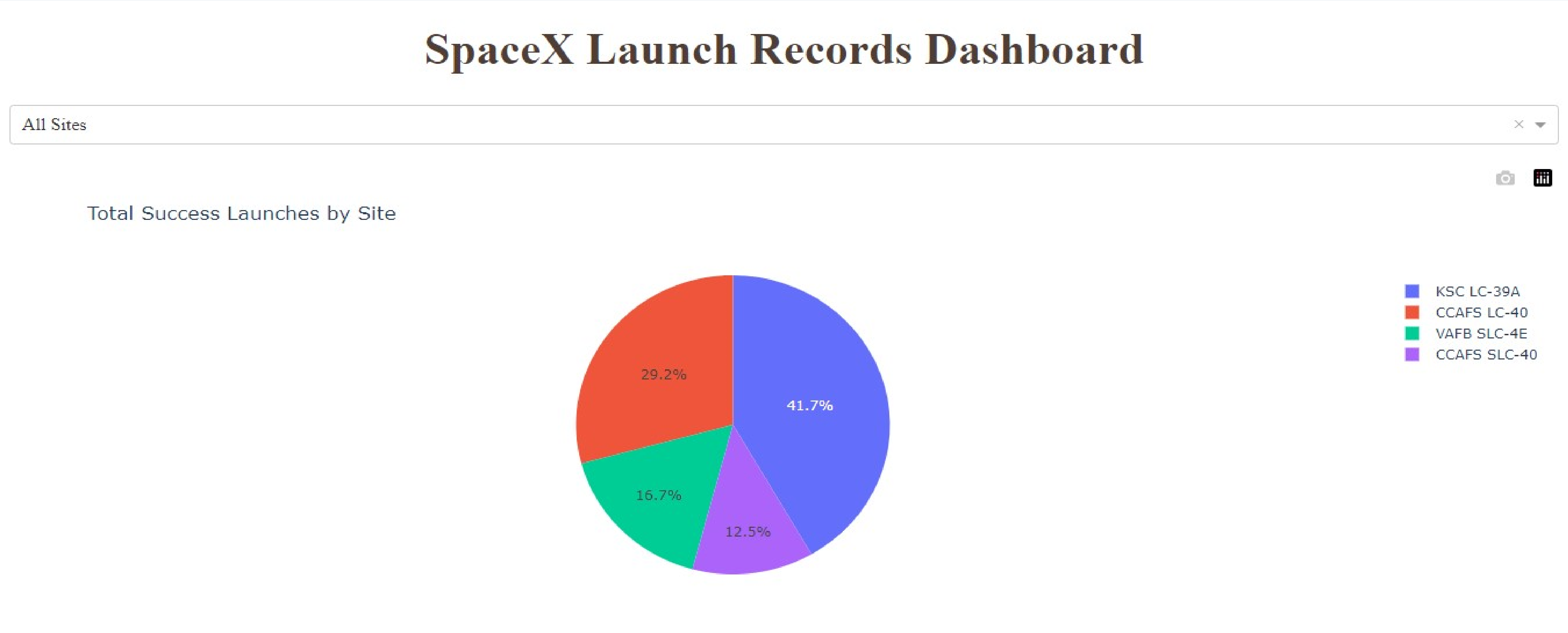
• Failed rocket with its high speed can cover distances like 15-20 km in few seconds. It could be potentially dangerous to populated areas.



Section 4

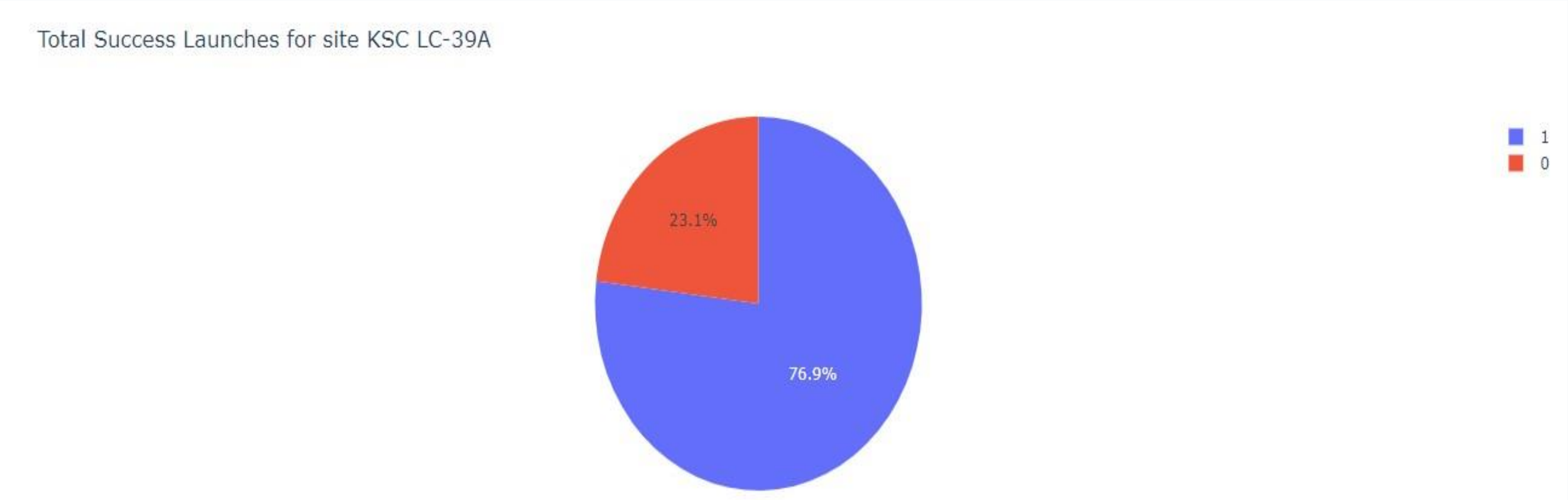
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>



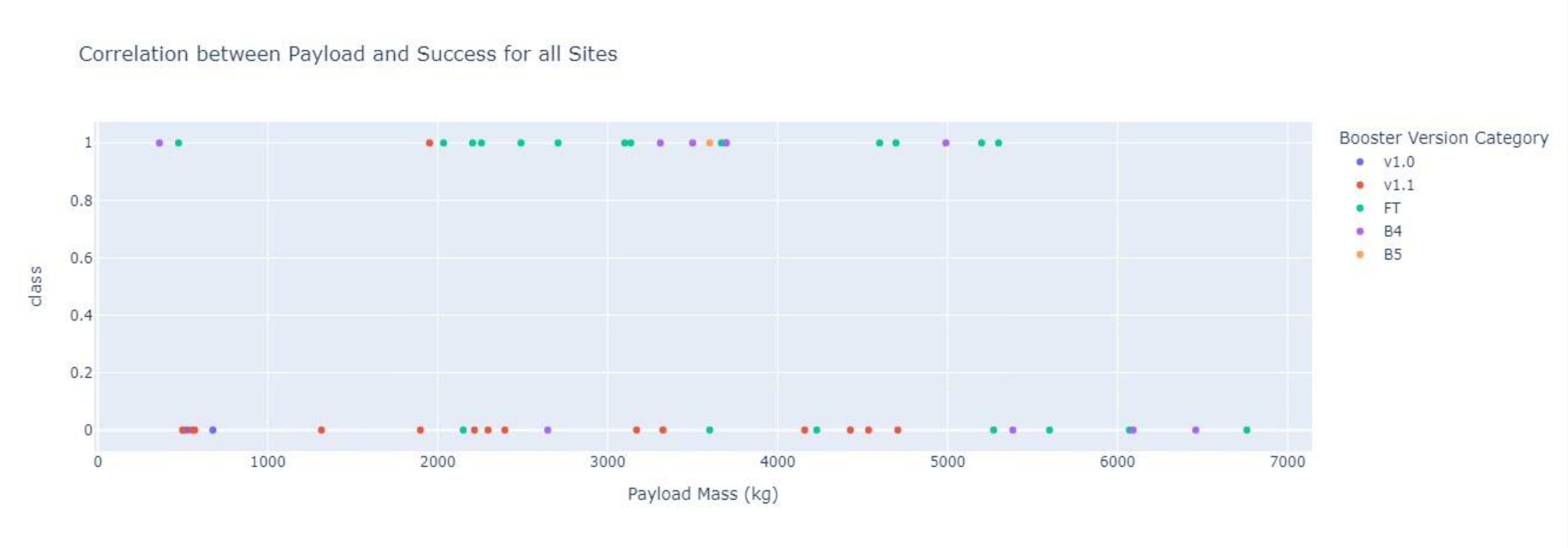
The chart clearly shows that from all the sites, KSC LC-39A has the most successful launches.

<Dashboard Screenshot 2>



KSC LC-39A has the highest launch success rate (76.9%) with 10 successful and only 3 failed landings.

<Dashboard Screenshot 3>



- Plotting the launch outcome vs. payload for all sites can be split into two ranges: a) 0 – 4000 kg (low payloads) and b) 4000 – 10000 kg (massive payloads). From the plot, it can be shown that the success for massive payloads is lower than that for low payloads.

- It is also worth noting that some booster types (v1.0 and B5) have not been launched with massive payloads.

Section 5

Predictive Analysis (Classification)

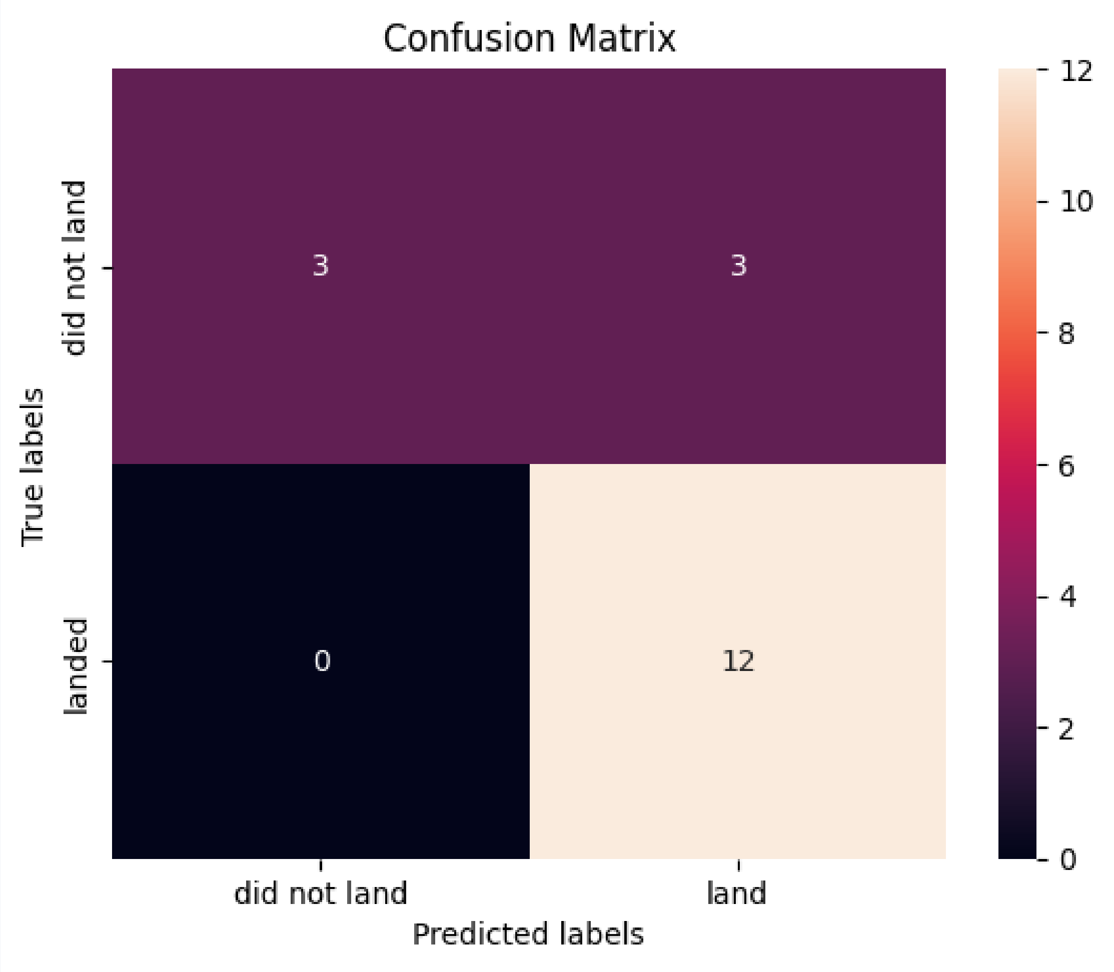
Classification Accuracy

Table of Scores and Accuracy of the Test Set

	LogReg	SVM	Tree	KNN
Jaccard_Score	0.8000	0.8000	0.8000	0.8000
F1_score	0.8889	0.8889	0.8889	0.8889
Accuracy	0.8333	0.8333	0.8333	0.8333

Based on the scores of the Test Set, we can not confirm which method performs best.

Confusion Matrix



- Examining the confusion matrix, we see that logistic regression can distinguish between the different classes.
- We observe that the major problem is false positives.

Conclusions

- Decision Tree Model is the best algorithm for this dataset.
- Launches with a low payload mass show better results than launches with a larger payload mass.
- Most of launch sites are in proximity to the Equator line and all the sites are in very close proximity to the coast.
- The success rate of launches increases over the years.
- KSC LC-39A has the highest success rate of the launches from all the sites.
- Orbits ES-L1, GEO, HEO and SSO have 100% success rate.

Appendix

Python Code for creating interactive Dashboard

```python
# Import required libraries
import pandas as pd
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output
import plotly.express as px
# Read the airline data into pandas dataframe
spacex_df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0

#Minimum and maximum Payload masses for the range slider
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()

# Create a dash application
app = dash.Dash(__name__)


#Get the launch sites from the spacex_df dataframe to use in the dropdown list
launch_sites = []
launch_sites.append({'label': 'All Sites', 'value': 'All Sites'})
for launch_site in spacex_df['Launch Site'].unique().tolist():
    launch_sites.append({'label': launch_site, 'value': launch_site})

# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),
 # TASK 1: Add a dropdown list to enable Launch Site selection
dcc.Dropdown(id='site-dropdown', options = launch_sites, value='All Sites',
                placeholder="Launch Site", searchable=True ),
            html.Br(),
# TASK 2: Add a pie chart to show the total successful launches count for all sites
# If a specific launch site was selected, show the Success vs. Failed counts for the site
html.Div(dcc.Graph(id='success-pie-chart')), #this id will be used in the callback function to chan
            html.Br(),
            html.P("Payload range (Kg):"),

# TASK 3: Add a slider to select payload range
dcc.RangeSlider(id='payload-slider', min=0, max=10000, step=1000,
                marks={0: '0', 2500: '2500', 5000: '5000', 7500: '7500', 10000: '10000'},
                value=[min_payload, max_payload]),

 # TASK 4: Add a scatter chart to show the correlation between payload and launch success
    html.Div(dcc.Graph(id='success-payload-scatter-chart')),
                                ])
```

```python
# TASK 2:
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
@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):
    filtered_df = spacex_df[spacex_df['Launch Site'] == entered_site]
    if entered_site == 'All Sites':
        fig = px.pie(spacex_df, values='class', names='Launch Site',
        title='Total Success Launches by Site')
        return fig
    else:
        # return the outcomes pie chart for a selected site
        site_df = filtered_df.groupby(
            ['Launch Site', 'class']).size().reset_index(name='class count')
        fig = px.pie(site_df,values='class count',names='class',
        title=f'Total Success Launches for site {entered_site}')
        return fig
```

```python
# TASK 4:
# Add a callback function for site-dropdown and payload-slider as inputs,
# success-payload-scatter-chart` as output
@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')])
               #note the 2 inputs, so they are in a list

def get_scatter_chart(entered_site, payload_slider):
    low, high = payload_slider
    slide=(spacex_df['Payload Mass (kg)'] > low) & (spacex_df['Payload Mass (kg)'] < high)
    dropdown_scatter=spacex_df[slide]

#If All sites are selected, render a scatter plot to display all values for variables
# Payload Mass (kg) and class. Point colour is set to the booster version category
    if entered_site == 'All Sites':
        fig = px.scatter(
            dropdown_scatter, x='Payload Mass (kg)', y='class',
            hover_data=['Booster Version'],
            color='Booster Version Category',
            title='Correlation between Payload and Success for all Sites')
        return fig
    else:
        #If a specific site is selected, filter the spacex_df dataframe first,
        # then render a scatter plot to display the same as for all sites.
        dropdown_scatter = dropdown_scatter[spacex_df['Launch Site'] == entered_site]
        fig=px.scatter(
            dropdown_scatter,x='Payload Mass (kg)', y='class',
            hover_data=['Booster Version'],
            color='Booster Version Category',
            title = f'Success by Payload Size for site {entered_site}')
        return fig

# Run the app
if __name__ == '__main__':
    app.run_server()
```

Thank you!