



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Timur Dautov
28 March 2023



Outline

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

Executive Summary

Summary of methodologies

- Data Collection (API, Web Scrapping)
- Data Wrangling
- Exploratory Data Analysis (EDA) with SQL
- Exploratory Data Analysis (EDA) with Data Visualization
- Interactive Visual Analytics with Folium
- Dashboard with Plotly Dash
- Machine Learning Predictive Analysis

Summary of all results

- Exploratory Data Analysis Results
- Interactive Visual Analytics Results
- Predictive Analysis Results

Introduction

- Project background and context

Space X 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 Space X 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 Space X for a rocket launch.

- Problems you want to find answers

- What factors influence the success of a rocket first stage landing?
- What is the relationship between the various analyzed variables?
- What impact do these relationships have on the success of the landing of the first stage of a rocket?
- What conditions must be created to ensure the best landing performance of the first stage of the rocket?

Section 1

Methodology

Methodology

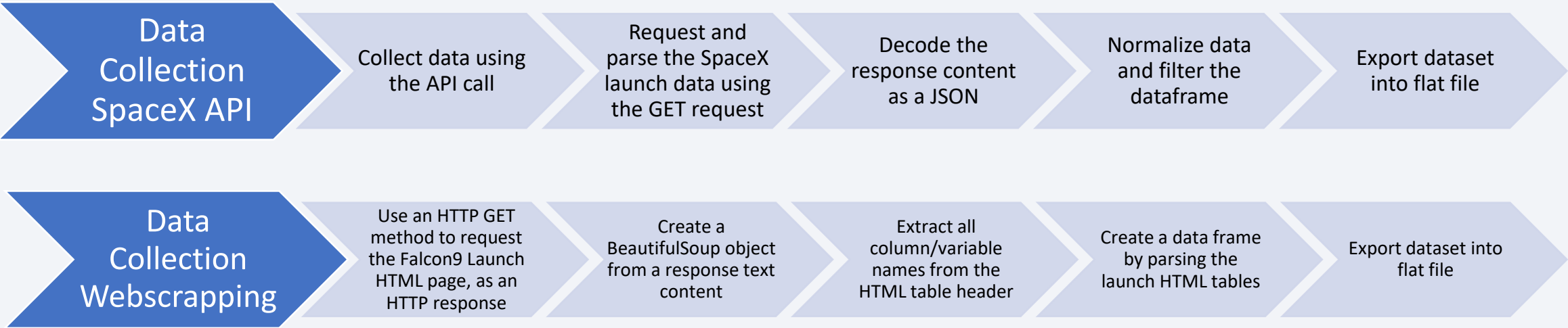
Executive Summary

- Data collection methodology:
 - SpaceX REST API was used for Data collection and Web scraping was done from Wikipedia page.
- Perform data wrangling
 - Data was processed using one-hot encoding for categorical 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
 - Cross Validation was used for tuning and Confusion Matrix was used for evaluating

Data Collection

Data sets were collected by the several ways:

- First, is the SpaceX REST API (api.spacexdata.com/v4/), which is used to get information about rockets, cores, launchpads, payloads.
- Second, is the Web scraping Falcon 9 and Falcon Heavy Launches Records from Wikipedia using BeautifulSoup.



Data Collection – SpaceX API

1. Requesting rocket launch data from SpaceX API.

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

```
response = requests.get(spacex_url)
```

2. Decode the response content as a Json and turn it into a Pandas dataframe.

```
data = pd.json_normalize(response.json())  
data
```

4. Clean data by applying custom functions (ex.).

```
# Call getLaunchSite  
getLaunchSite(data)
```

6. Filter the dataframe to only include Falcon 9 launches.

```
data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']
```

7. Dealing with Missing Values.

```
# Calculate the mean value of PayloadMass column  
avg_PayloadMass = data_falcon9['PayloadMass'].astype('float').mean(axis=0)  
# Replace the np.nan values with its mean value  
data_falcon9['PayloadMass'].replace(np.nan, avg_PayloadMass, inplace=True)
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

8. Export dataset to a CSV.

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

3. Use the API again to get information about the launches.

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.  
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]
```

5. Construct our dataset using the data we have obtained by combining the columns into a dictionary.

```
launch_dict = {'FlightNumber': list(data['flight_number']),  
               'Date': list(data['date']),  
               'BoosterVersion': BoosterVersion,  
               'PayloadMass': PayloadMass,  
               'Orbit': Orbit,  
               'LaunchSite': LaunchSite,  
               'Outcome': Outcome,  
               'Flights': Flights,  
               'GridFins': GridFins,  
               'Reused': Reused,  
               'Legs': Legs,  
               'LandingPad': LandingPad,  
               'Block': Block,  
               'ReusedCount': ReusedCount,  
               'Serial': Serial,  
               'Longitude': Longitude,  
               'Latitude': Latitude}
```


Data Collection – Scraping

1. Request the Falcon9 Launch Wiki page from its URL.

```
# use requests.get() method with the provided static_url
requests.get(static_url)
# assign the response to a object
html = requests.get(static_url).text
```

4. Create an empty dictionary with keys from the extracted column names.

```
launch_dict= dict.fromkeys(column_names)

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

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

6. Create a dataframe and export dataset to a CSV.

```
df=pd.DataFrame(launch_dict)

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

2. Create a BeautifulSoup object from the HTML response.

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html, 'html.parser')
```

3. Extract all column/variable names from the HTML table header.

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
for row in first_launch_table.find_all('th'):
    # Iterate each th element and apply the provided extract_column_from_header() to get a column name
    name = extract_column_from_header(row)
    # Append the Non-empty column name ('if name is not None and len(name) > 0') into a list called column_names
    if (name != None and len(name) > 0):
        column_names.append(name)
```

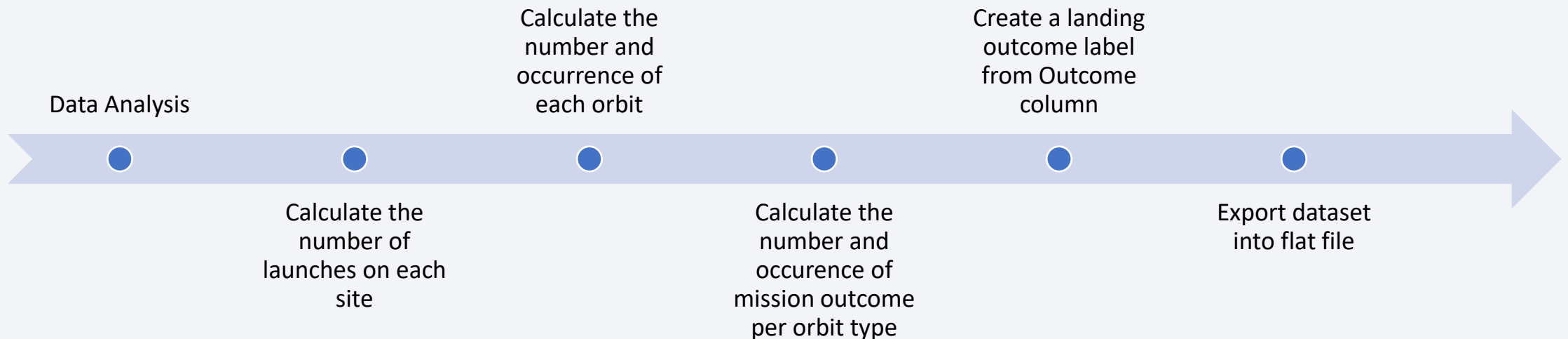
5. Fill up the launch_dict with launch records extracted from table rows (ex. part of code).

```
row=rows.find_all('td')
#if it is number save cells in a dictionary
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])
```

GitHub URL of the completed web scraping notebook: [Data Collection with Webscraping](#)

Data Wrangling

Data wrangling were done by performing Exploratory Data Analysis (EDA) on a dataset. It is represented by the following workflows:



Data Wrangling

1. Calculate the number of launches on each site.

```
# Apply value_counts() on column LaunchSite
df.value_counts('LaunchSite')
```

2. Calculate the number and occurrence of each orbit.

```
# Apply value_counts on Orbit column
df.value_counts('Orbit')
```

3. Calculate the number and occurrence of mission outcome per orbit type.

```
# landing_outcomes = values on Outcome column
landing_outcomes = df.value_counts('Outcome')
landing_outcomes
```

4. Create a set of outcomes where the second stage did not land successfully.

```
bad_outcomes = set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

5. Create a landing outcome label from Outcome column.

```
# 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)
```

```
{'False ASDS', 'False Ocean', 'False RTL5', 'None ASDS', 'None None'}
```

6. Determine the success rate.

```
df["Class"].mean()

0.6666666666666666
```

7. Export dataset to a CSV.

```
df.to_csv("dataset_part\2.csv", index=False)
```

GitHub URL of the completed data wrangling notebook: [Data Wrangling](#)

EDA with Data Visualization

Next charts were plotted:

Scatter graphs:	Payload Mass vs Flight Number
	Launch Site vs Flight Number
	Launch Site vs Payload Mass
	Orbit Type vs Flight Number
	Orbit Type vs Payload Mass

Bar Chart:	Success Rate vs Orbit Type
-------------------	----------------------------

Line Chart:	Success Rate vs Year
--------------------	----------------------

Scatter graphs were used for determination of the correlation (patterns) between different variables. This graphs were used to find variables (factors) which are lead to maximum probability of success in landing outcome.

Bar chart was used to represent relation between two categorical variables.

Line chart was used to represent a success rate trending line of outcomes over the years. This helps to predict the result of the launch outcomes in the future.

EDA with SQL

SQL queries were written and executed to solve some tasks, namely:

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 the names of the booster_versions which have carried the maximum payload mass.

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 successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

GitHub URL of the completed EDA with Data Visualization notebook: [EDA with SQL](#)

Build an Interactive Map with Folium

In this part of our research we have created and added to a folium map the following objects:

- Mark all launch sites on a map.
- Add a circle for each launch site in data frame.
- Mark the success/failed launches for each site on the map.
- Simplify a map containing many markers having the same coordinate with marker clusters.
- Calculate the distances between a launch site to its proximities.
- Mark down a point on the closest coastline and calculate the distance between the coastline point and the launch site.
- Draw a PolyLine between a launch site to the selected coastline point
- Draw a line between a launch site to its closest city, railway, highway

The launch success rate may depend on many factors such as payload mass, orbit type, and so on. It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories. Finding an optimal location for building a launch site certainly involves many factors and we could discover some of the factors by analyzing the existing launch site locations. So, we use Folium to visualize geo-spatial data and to find some geographical patterns about launch sites.

GitHub URL of the completed EDA with Data Visualization notebook: [EDA with Folium](#)

Build a Dashboard with Plotly Dash

As a result, the following objects were built:

- The pie chart with the total success launches by Sites.
- Scatter graph with correlation between payload mass and success for all Sites.

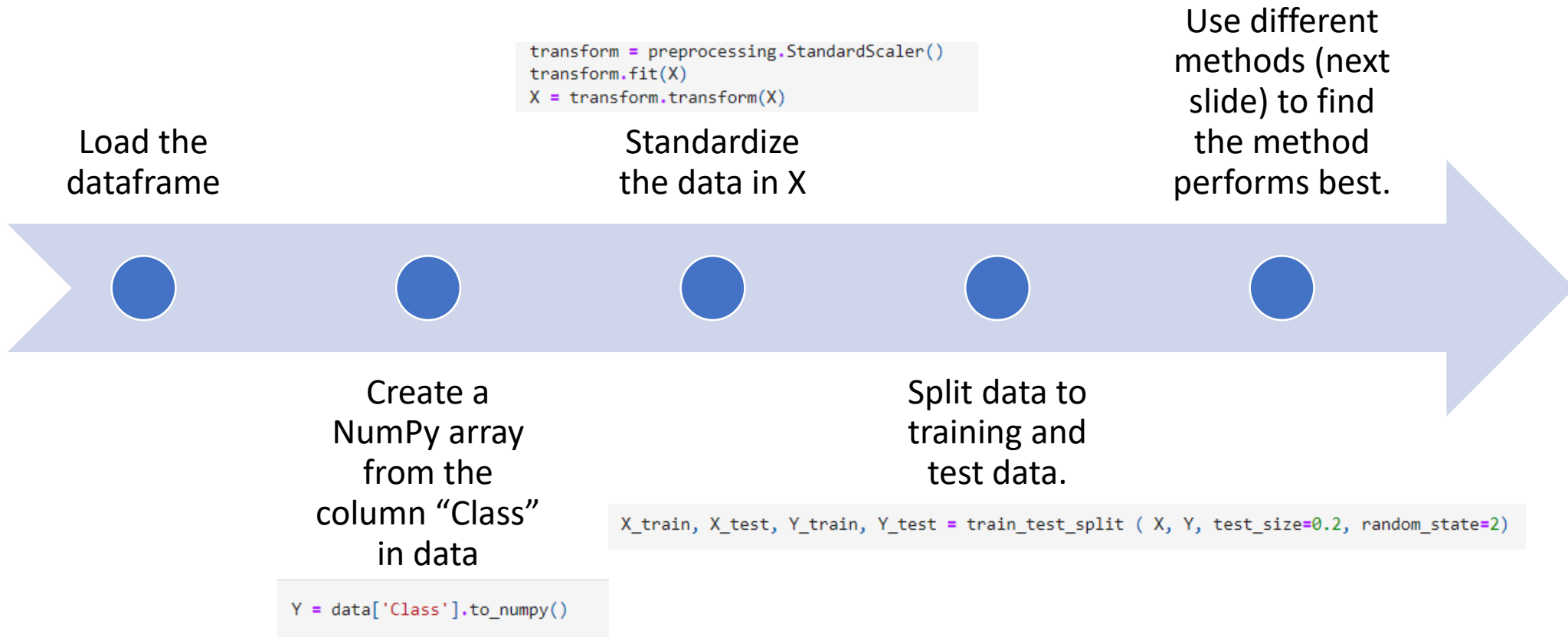
This plots and interactions were built to perform interactive visual analytics on SpaceX launch data in real-time. We obtained insights, which helped us to answer important questions about launches, namely:

- Which site has the largest successful launches?
- Which site has the highest launch success rate?
- Which payload range(s) has the highest launch success rate?
- Which payload range(s) has the lowest launch success rate?
- Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest launch success rate?

GitHub URL of the completed Dashboard Application with Plotly Dash: [Plotly Dash Lab](#)

Predictive Analysis (Classification) - 1

The flowchart how we built, evaluated, improved, and found the best performing classification model presented below:



Predictive Analysis (Classification) -2

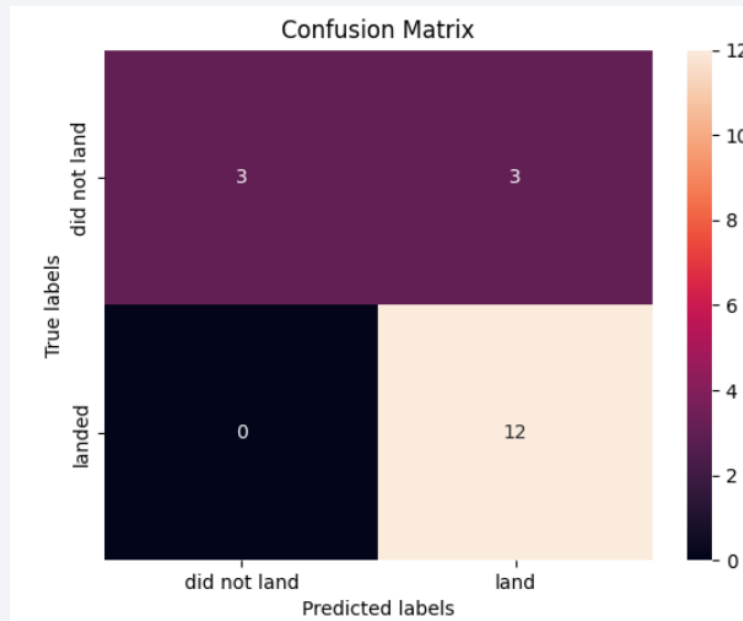
Methods that have been used and their confusion matrix:

Logistic regression
object.

Support vector machine
object.

Decision tree classifier
object.

K-nearest neighbors
object.



Find the method performs best:

```
print('Accuracy for LR:', logreg_cv.score(X_test, Y_test))
print('Accuracy for SVM:', svm_cv.score(X_test, Y_test))
print('Accuracy for DT:', tree_cv.score(X_test, Y_test))
print('Accuracy for KNN:', knn_cv.score(X_test, Y_test))
```

Accuracy for LR: 0.8333333333333334
Accuracy for SVM: 0.8333333333333334
Accuracy for DT: 0.8333333333333334
Accuracy for KNN: 0.8333333333333334

GitHub URL of the completed Predictive Analysis: [Predictive Analysis with ML](#)

Results

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

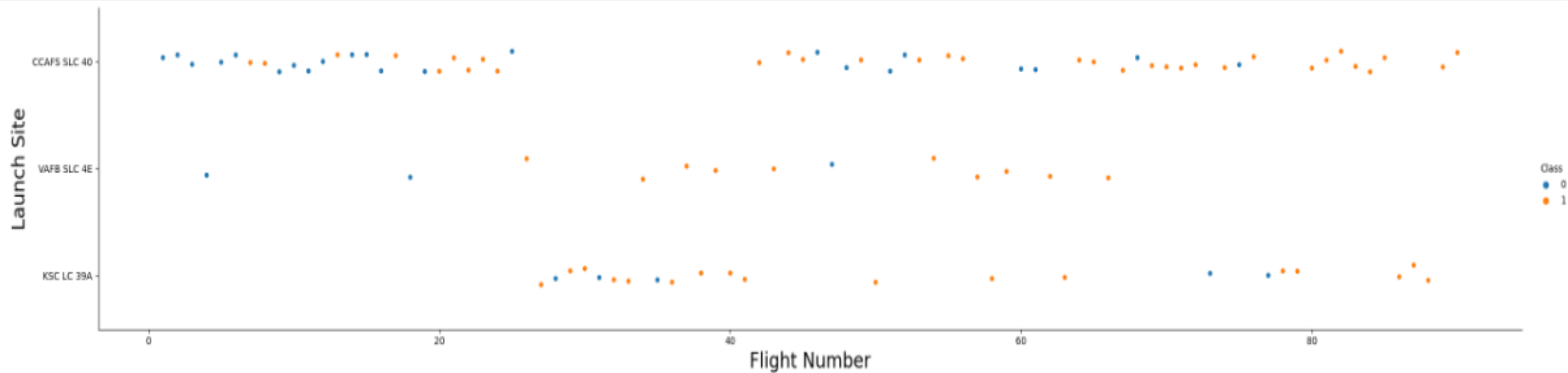
The background of the slide is an abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks in shades of red and cyan. A faint, light blue grid pattern is also visible, particularly in the lower-left quadrant. The overall effect is dynamic and technological.

Section 2

Insights drawn from EDA

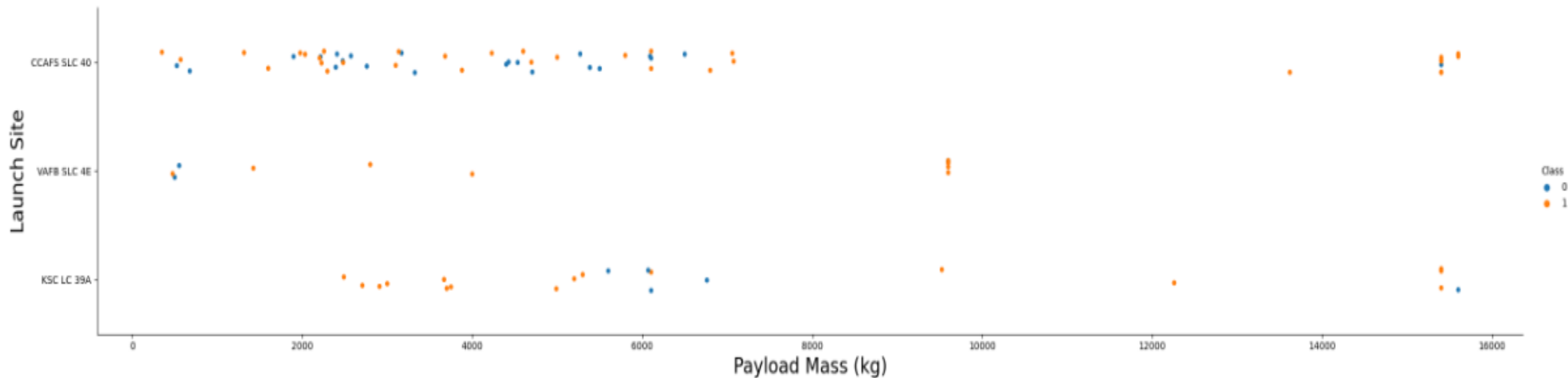
Flight Number vs. Launch Site

- More flights at the launch site, more success rate at the launch site.



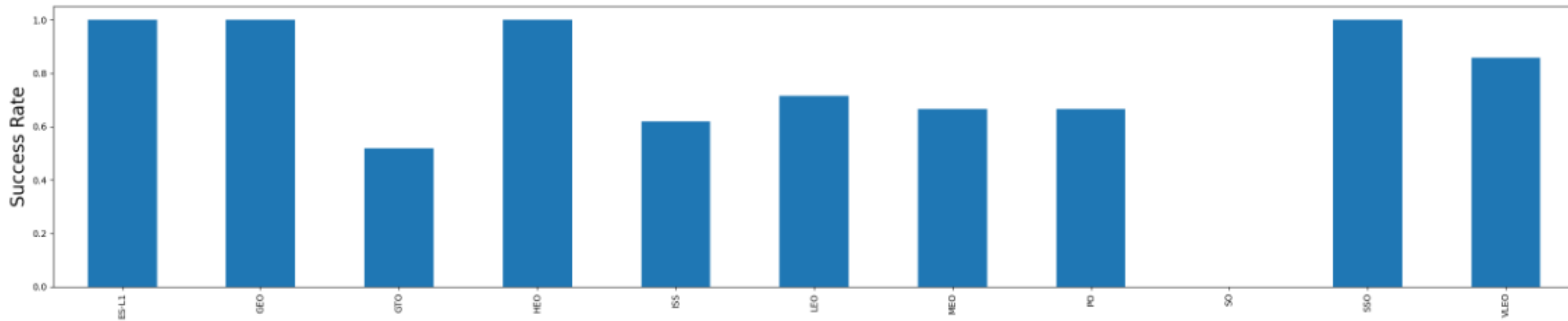
Payload vs. Launch Site

- The greater the payload mass, higher the success rate



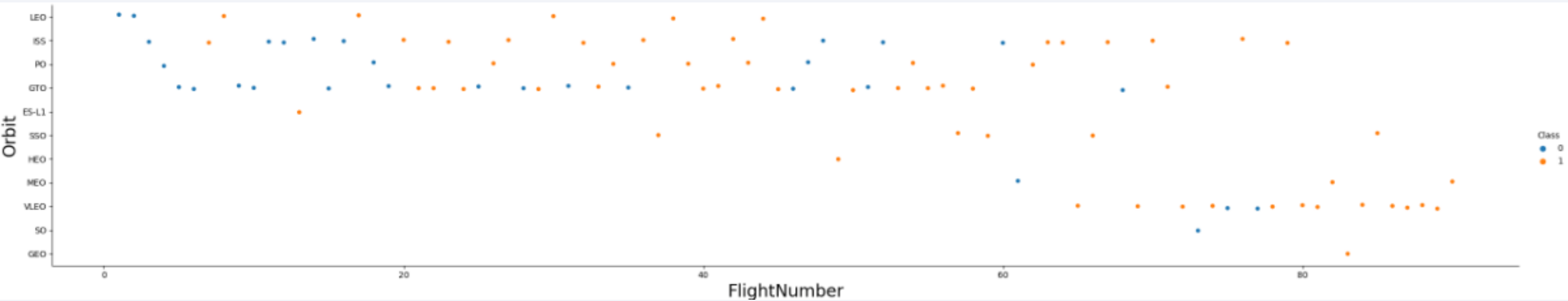
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, SSO have the highest success rates.



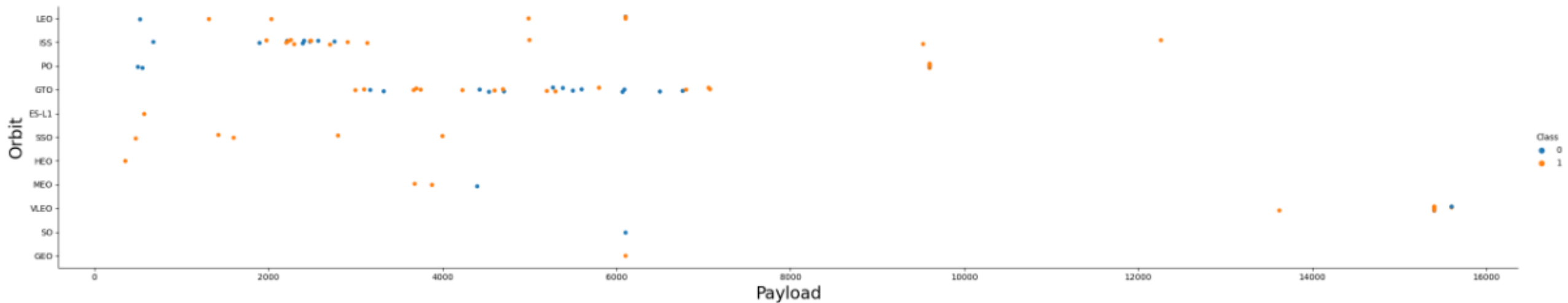
Flight Number vs. Orbit Type

- LEO, ISS, VLEO success rate increases with number of flights.
- There is no relation between flight number and GTO orbit.



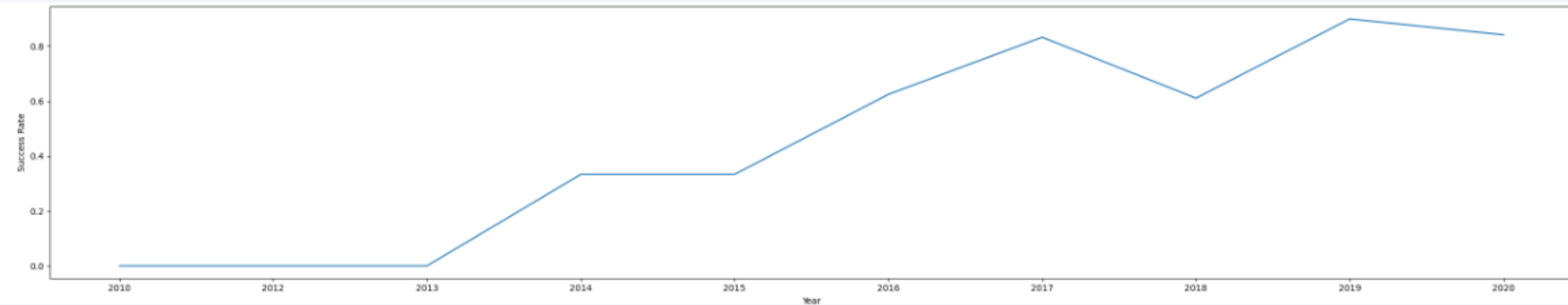
Payload vs. Orbit Type

- Heavy payloads have positive influence on LEO, ISS orbits.
- Heavy payloads have negative influence on MEO, GTO, VLEO orbits.



Launch Success Yearly Trend

- Success rate has been increasing since 2013 to 2020 but it was a decreasing in 2018.



All Launch Site Names

Names of the unique launch sites. We use «DISTINCT» keyword to get unique values from the column «Launch_Site» from table «SPACEXTBL».

```
%%sql  
SELECT DISTINCT LAUNCH_SITE  
FROM SPACEXTBL;
```

```
* sqlite:///my_data1.db  
Done.
```

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

Launch Site Names Begin with 'CCA'

5 records where launch sites begin with the string 'CCA'. We use «LIMIT» keyword to get 5 records from table «SPACEXTBL», and «LIKE» to find launch sites names started with CCA.

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

```
* sqlite:///my_data1.db
Done.
```

Total Payload Mass

Calculate the total payload carried by boosters from NASA. We use «SUM» keyword to get information from «SPACEXTBL» and choose a customer as «NASA (CRS)».

```
%%sql  
SELECT SUM(PAYLOAD_MASS_KG_) FROM SPACEXTBL  
WHERE Customer = 'NASA (CRS)';
```

```
* sqlite:///my_data1.db
```

```
Done.
```

```
SUM(PAYLOAD_MASS_KG_)
```

```
45596
```


Average Payload Mass by F9 v1.1

Calculate the average payload mass carried by booster version F9 v1.1. We use «AVG» keyword to get average mass of payloads from «SPACEXTBL» and choose a booster version as «F9 v1.1».

```
%%sql  
SELECT AVG(PAYLOAD_MASS_KG_) FROM SPACEXTBL  
WHERE Booster_Version = 'F9 v1.1';
```

```
* sqlite:///my_data1.db  
Done.
```

AVG(PAYLOAD_MASS_KG_)
2928.4

First Successful Ground Landing Date

Find the dates of the first successful landing outcome on ground pad. We use «MIN» keyword to get the first successful landing outcome from «SPACEXTBL» and choose a «Landing_Outcome» as «Success (ground pad)».

```
%%sql
SELECT MIN(Date)
FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Success (ground pad)';
```

```
* sqlite:///my_data1.db
```

```
Done.
```

MIN(Date)

01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000. We select «Booster_Version» from «SPACEXTBL» and choose a Payload mass between 4000 and 6000 when «Landing_Outcome» equal «Success (drone ship)».

```
%%sql
SELECT Booster_Version FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Success (drone ship)'
      AND 4000 < PAYLOAD_MASS_KG_ < 6000;
```

```
* sqlite:///my_data1.db
Done.
```

Total Number of Successful and Failure Mission Outcomes

Calculate the total number of successful and failure mission outcomes. Select «MISSION_OUTCOME» from «SPACEXTBL». «COUNT» keyword is used for counting total number of successful and failure outcomes. Then we grouping them using «GROUP BY».

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

```
* sqlite:///my_data1.db
Done.
```

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass. We use «MAX» keyword to find the maximum value in the column «PAYLOAD_MASS__KG_»

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

```
* sqlite:///my_data1.db
Done.
```

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015. We use «WHERE» to filter records that are 'Failure (drone ship)' and are from Year 2015.

```
%%sql
SELECT "Landing _Outcome", Booster_Version, Launch_Site FROM SPACEXTBL
WHERE "Landing _Outcome" = 'Failure (drone ship)'
      AND substr(Date, 4, 2)
      AND substr(Date, 7, 4) = "2015"
```

```
* sqlite:///my_data1.db
```

```
Done.
```

Landing _Outcome	Booster_Version	Launch_Site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- 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. We used «GROUP BY» to group result by «Landing_Outcome» and «ORDER BY» is used to put the results in descending order with «DESC» keyword.

```
%%sql
SELECT "Landing_Outcome", COUNT("Landing_Outcome") AS total_number FROM SPACEXTBL
WHERE DATE BETWEEN "04-06-2010" AND "20-03-2017" AND "Landing_Outcome" LIKE "%Success%"
GROUP BY "Landing_Outcome"
ORDER BY total_number DESC
```

```
* sqlite:///my_data1.db
Done.
```

Landing_Outcome	total_number
Success	20
Success (drone ship)	8
Success (ground pad)	6

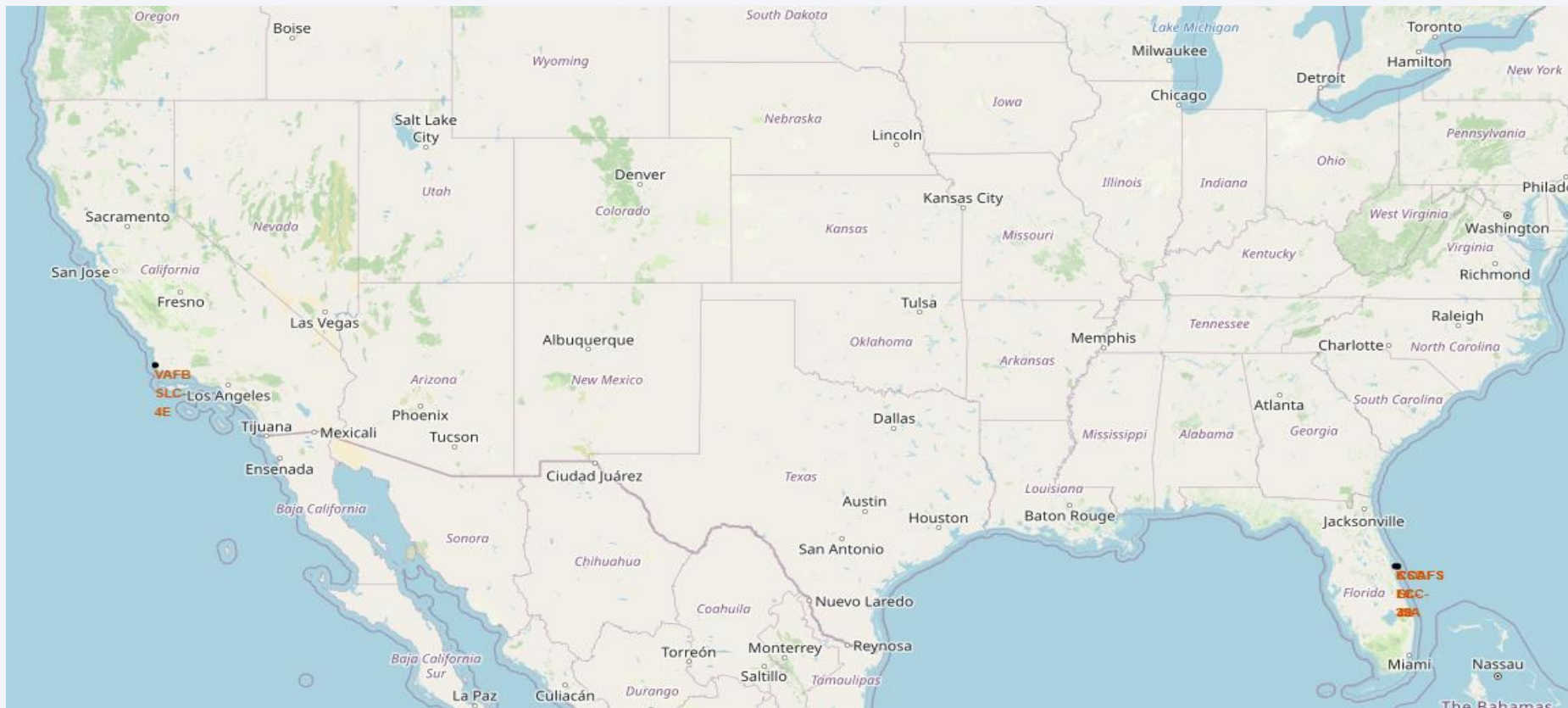
A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

Section 3

Launch Sites Proximities Analysis

All Launch Sites Map Markers

We can see that all launch sites of SpaceX are in the USA, in Florida and California.



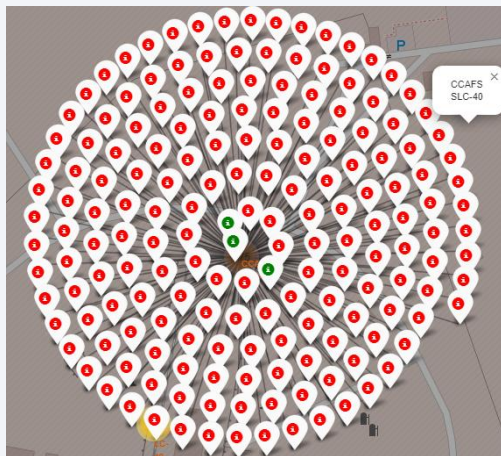
Color-labeled launch outcomes

The number of all launches is marked with circles:

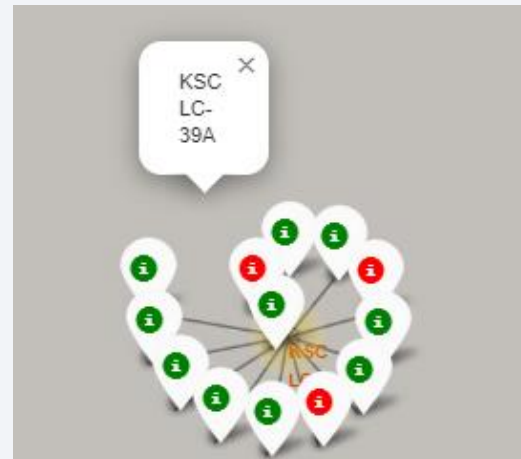
Successful launches labeled with **Green** marker, unsuccessful with **Red** marker. We can see that KSC LC-39A has the most successful launches.



CCAFS SLC-40



KSC LC-39A



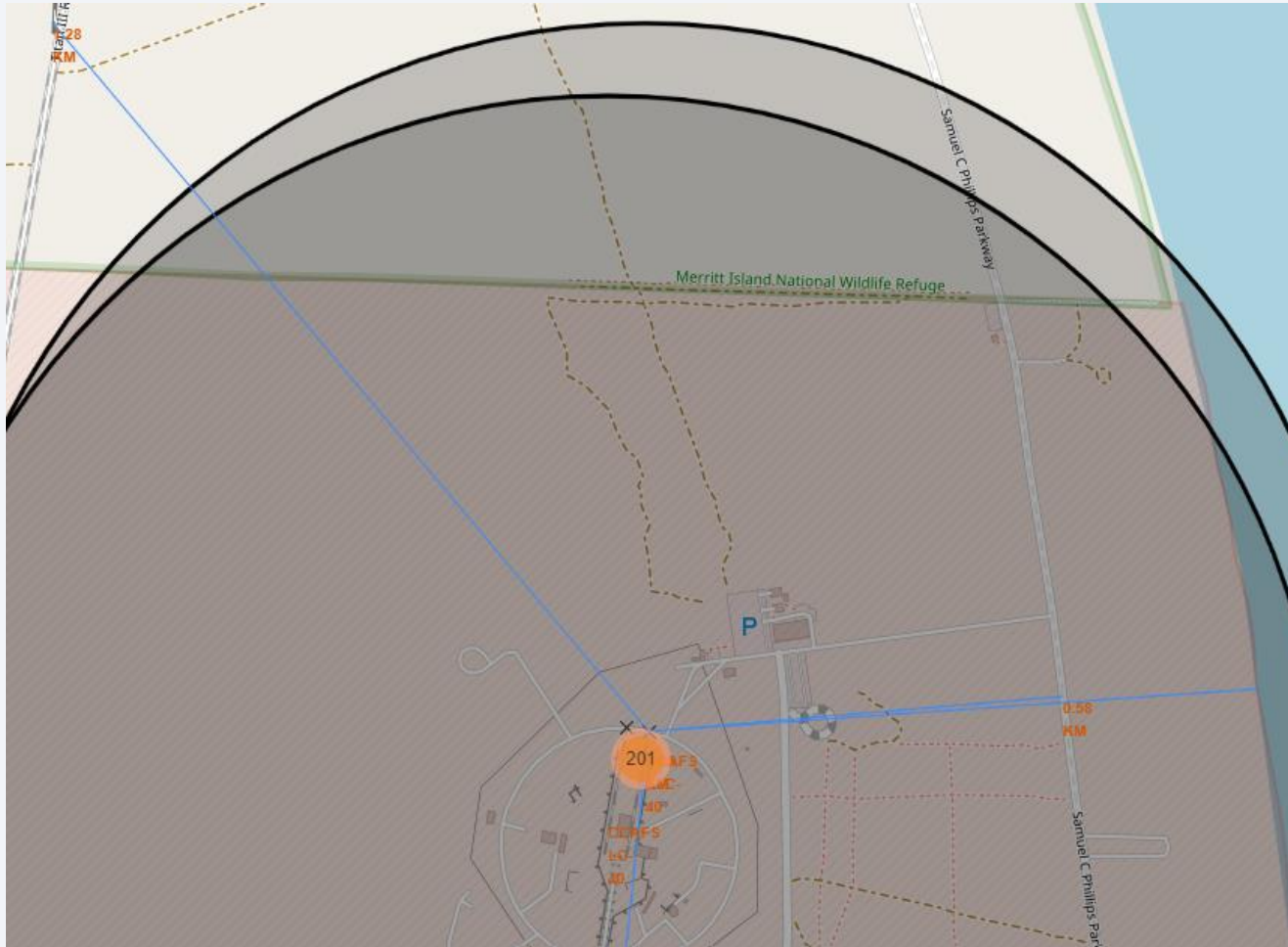
CCAFS LC-40



VAFB SLC-4E



Launch Site to Proximities (Railway, Highway, Coastline, City)



1. Are launch sites in close proximity to railways? Yes (Distance is less than 2 km)
2. Are launch sites in close proximity to highways? Yes (Distance is less than 2 km)
3. Are launch sites in close proximity to coastline? Yes (Distance is less than 2 km)
4. Do launch sites keep certain distance away from cities? Yes (Distance is more than 50 km)



Section 4

Build a Dashboard with Plotly Dash

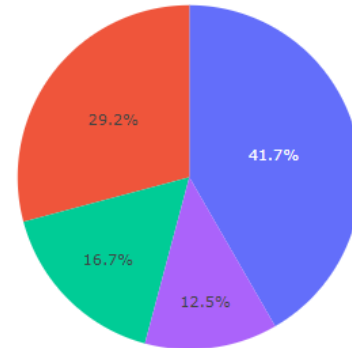
Launch Success Count for All Sites

From this piechart we can claim that KSC LC-39A site has the highest number of successful launches among all the sites.

SpaceX Launch Records Dashboard

All Sites ×

Total Success Launches By Site

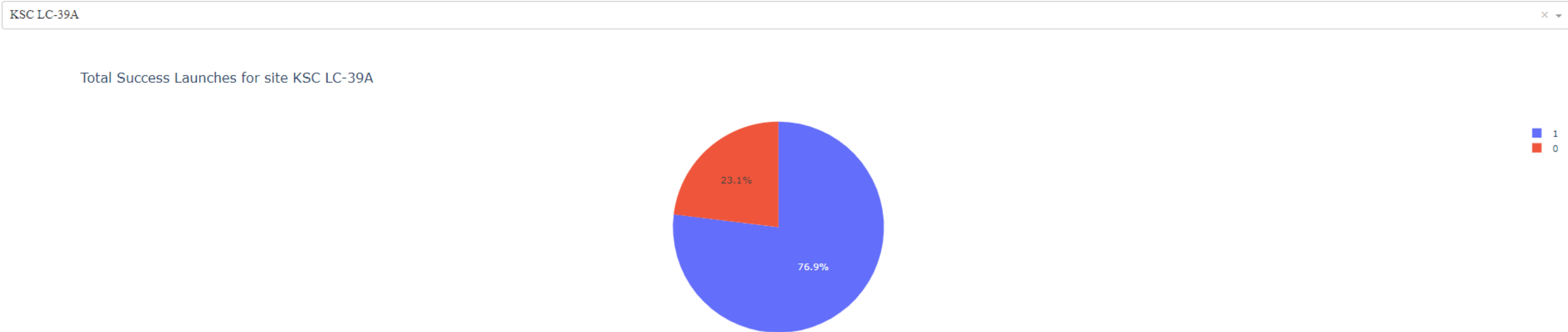


■ KSC LC-39A
■ CCAFS LC-40
■ VAFB SLC-4E
■ CCAFS SLC-40

Launch Site With Highest Launch Success Ratio

The launch site with the highest success ratio is KSC LC-39A with 76,9% of successful launches and 23,1% of unsuccessful.

SpaceX Launch Records Dashboard



Sites With Different Payload

We can see that success rate for low-weighted payloads is higher than for high-weighted payloads. The most successful booster version is FT. The most unsuccessful is v1.1

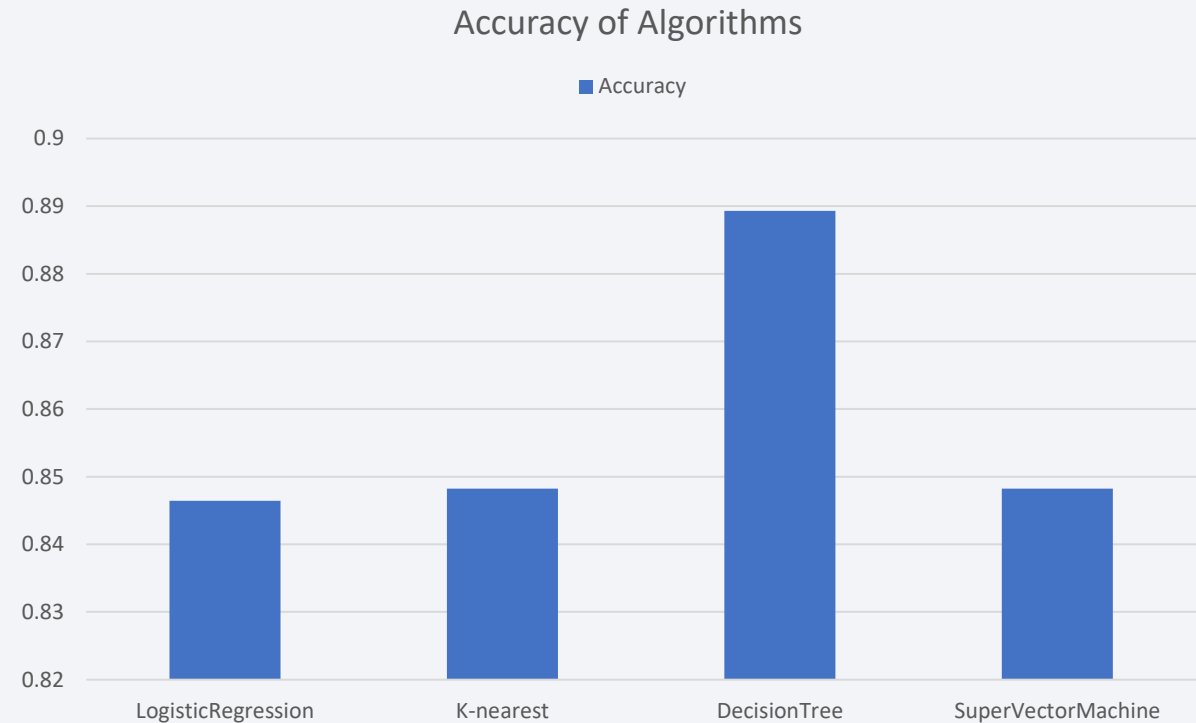


Section 5

Predictive Analysis (Classification)

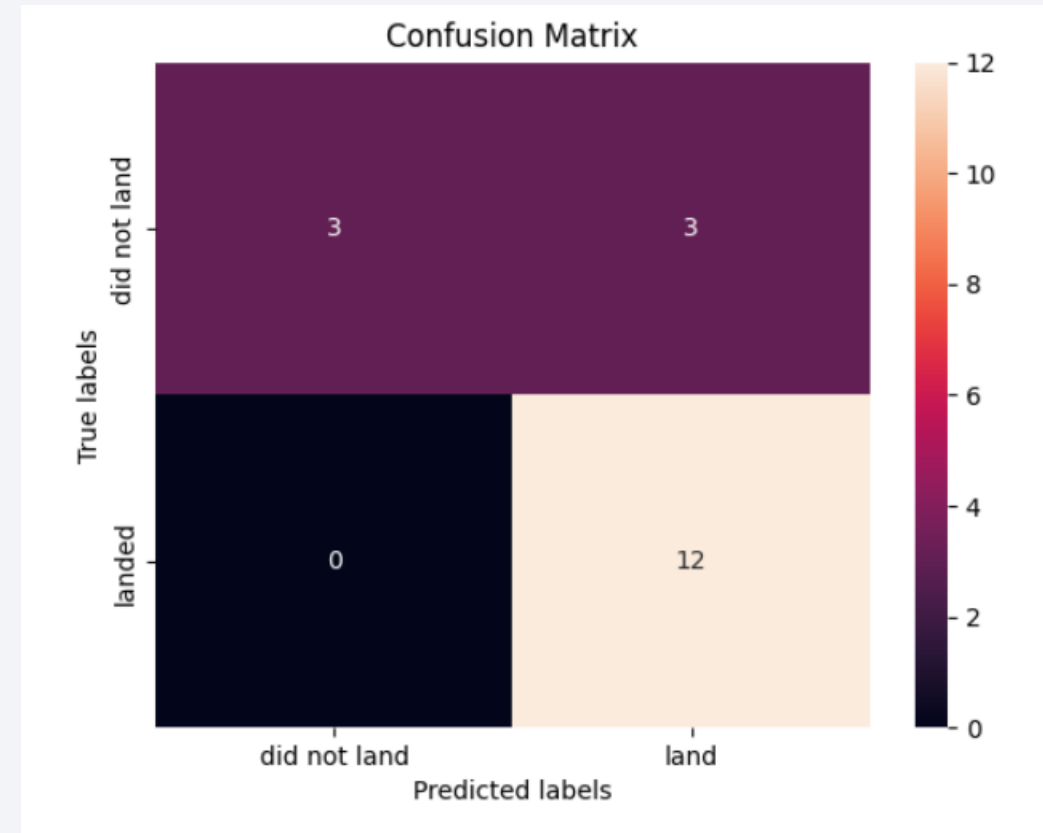
Classification Accuracy

- Decision Tree has performed best with maximum accuracy of 88.9% on a training data
- After selecting best hyperparameters, we get 83,33% accuracy on a testing data.

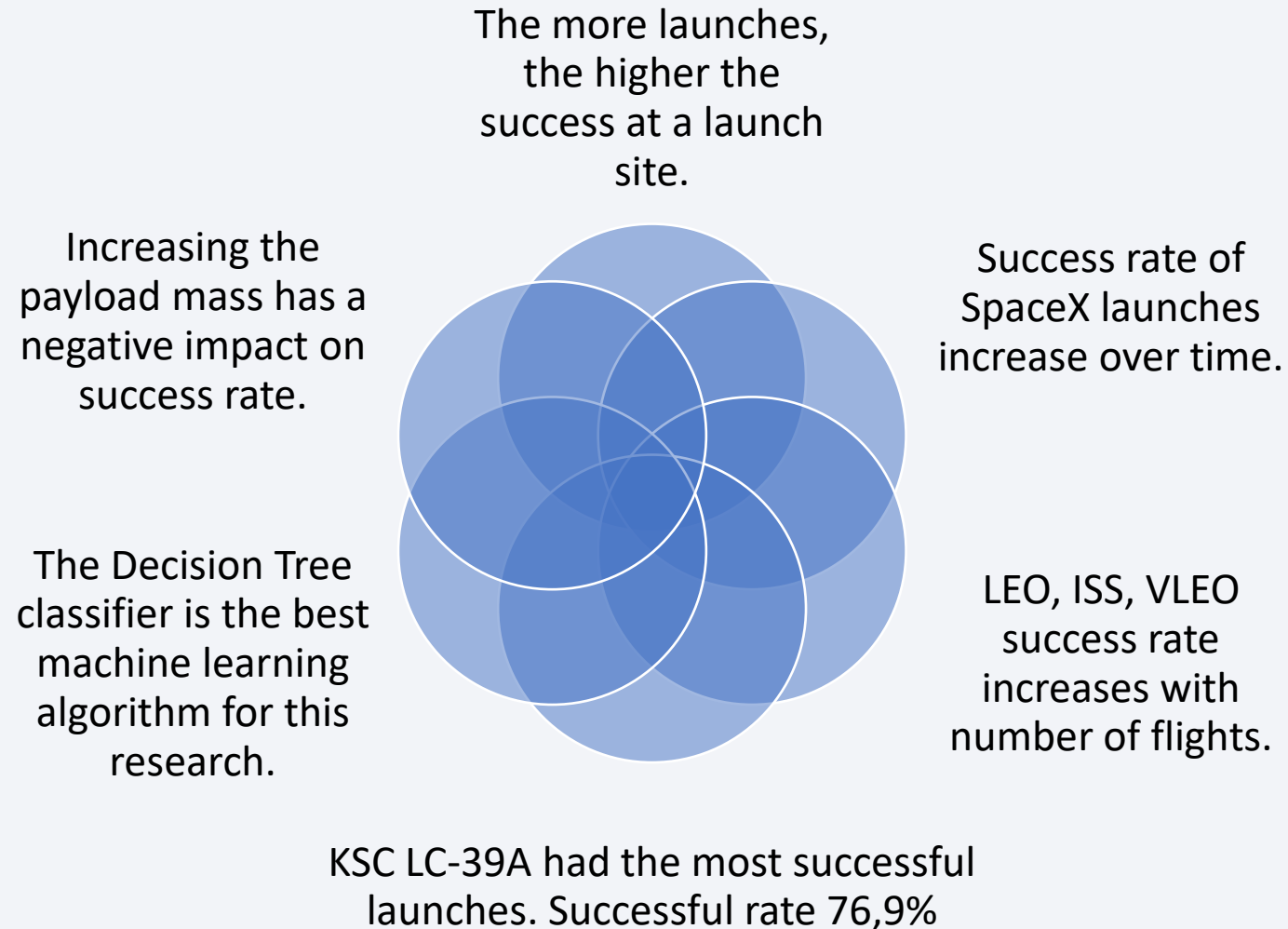


Confusion Matrix

We can see, that the main problem in False Positive combinations because unsuccessful landings were marked as successful landings by the classifier.

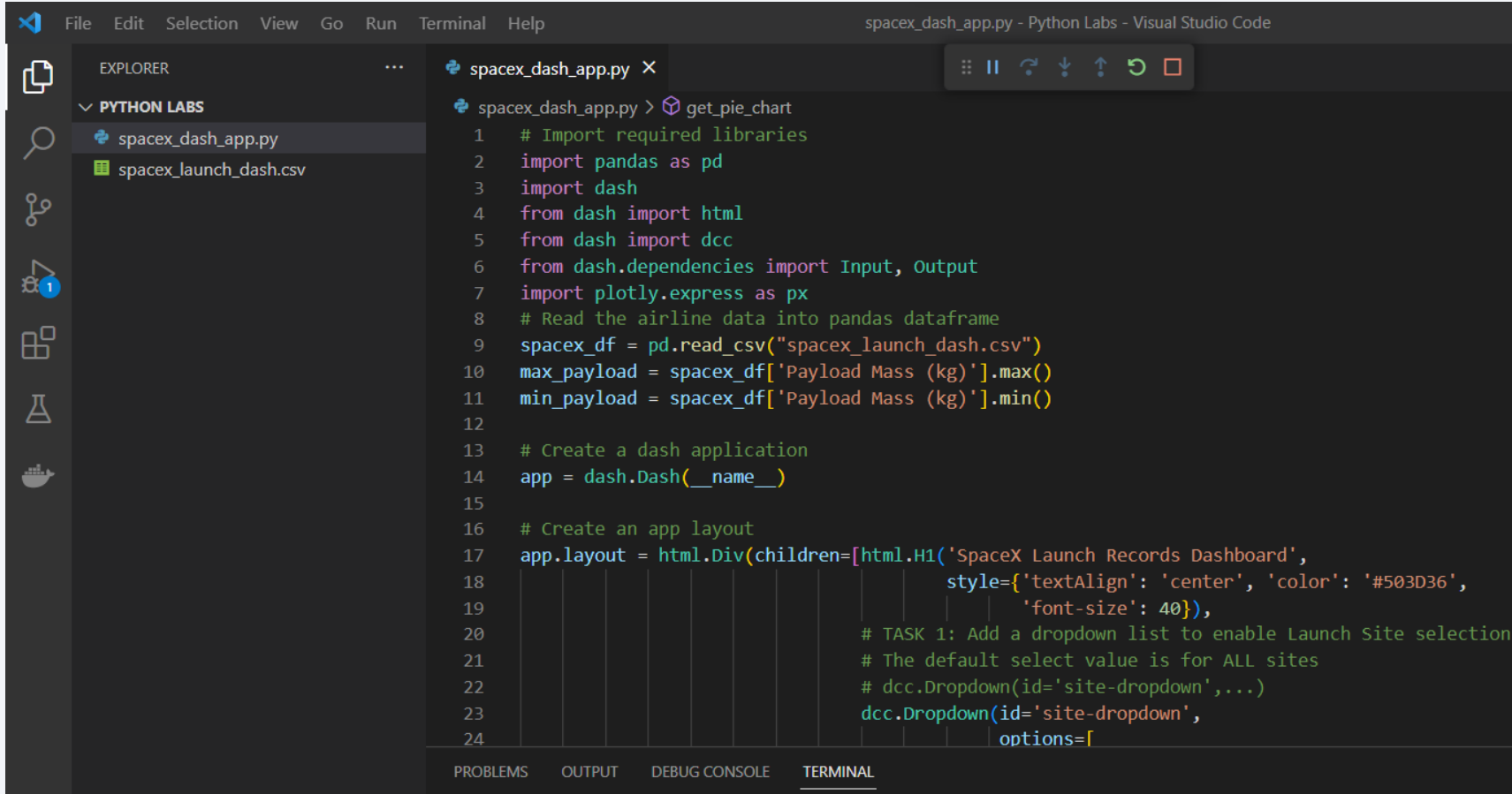


Conclusions



Appendix

Microsoft Visual Studio Code was used with Anaconda virtual environment for creating Dashboard.



```
spacex_dash_app.py - Python Labs - Visual Studio Code

EXPLORER
PYTHON LABS
  spacex_dash_app.py
  spacex_launch_dash.csv

spacex_dash_app.py > get_pie_chart
1  # Import required libraries
2  import pandas as pd
3  import dash
4  from dash import html
5  from dash import dcc
6  from dash.dependencies import Input, Output
7  import plotly.express as px
8  # Read the airline data into pandas dataframe
9  spacex_df = pd.read_csv("spacex_launch_dash.csv")
10 max_payload = spacex_df['Payload Mass (kg)'].max()
11 min_payload = spacex_df['Payload Mass (kg)'].min()
12
13 # Create a dash application
14 app = dash.Dash(__name__)
15
16 # Create an app layout
17 app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
18                                         style={'textAlign': 'center', 'color': '#503D36',
19                                             'font-size': 40}),
20                               # TASK 1: Add a dropdown list to enable Launch Site selection
21                               # The default select value is for ALL sites
22                               # dcc.Dropdown(id='site-dropdown',...)
23                               dcc.Dropdown(id='site-dropdown',
24                                             options=]
```


Thank you!

