IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

A. Miskelis
09/02/23

# Outline

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

# Executive Summary

- Summary of methodologies
- Summary of all results

# Introduction

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

Section 1

# Methodology

# Methodology

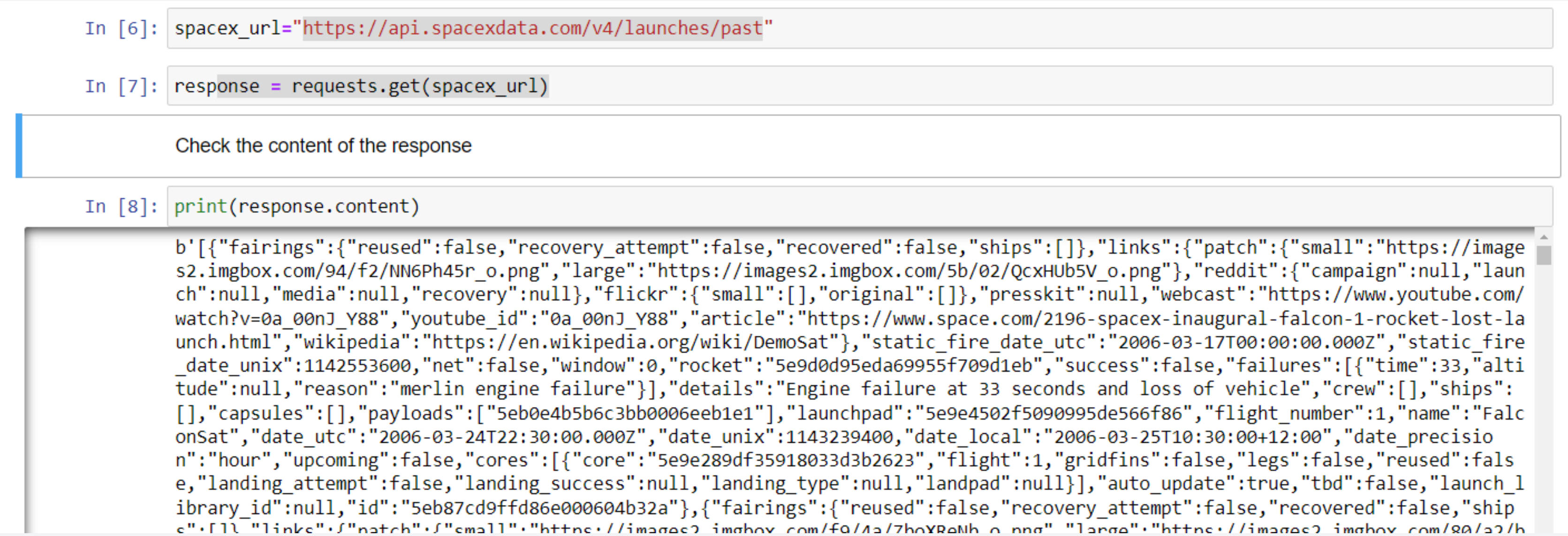
## Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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

To collect data I have used SpaceX API with the following URL: [https://api.spacexdata.com/v4/launches/past](https://api.spacexdata.com/v4/launches/past)

To request data library requests was used



```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)
```

Check the content of the response

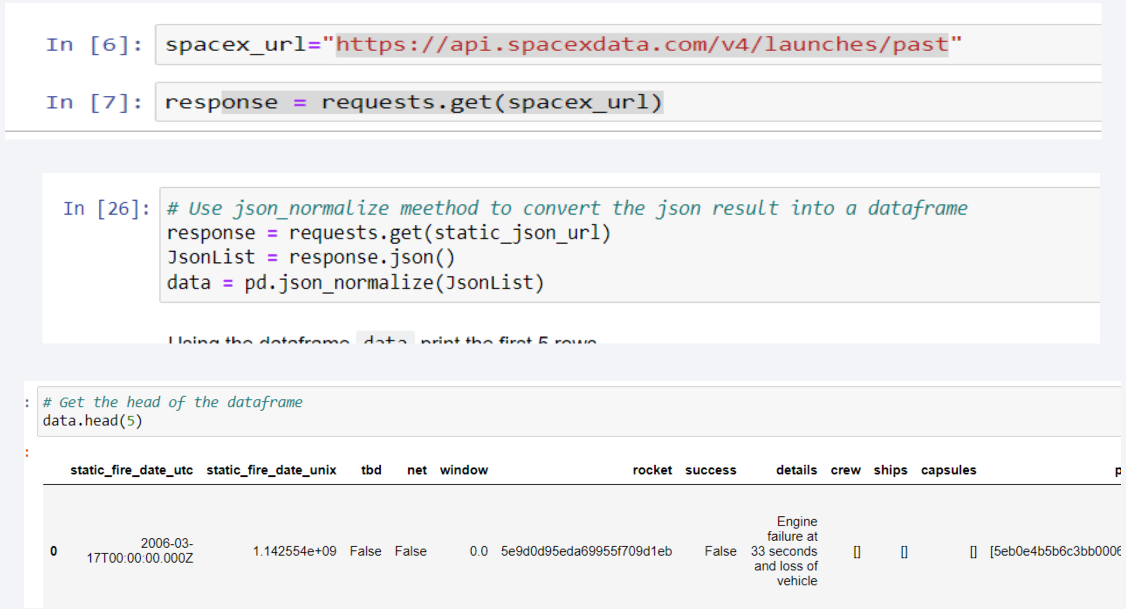
```
In [8]: print(response.content)

b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://image
s2.imgbox.com/94/f2/NN6Ph45r_o.png","large":"https://images2.imgbox.com/5b/02/QcxHUb5V_o.png"},"reddit":{"campaign":null,"laun
ch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/
watch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.space.com/2196-spacex-inaugural-falcon-1-rocket-lost-la
unch.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":"2006-03-17T00:00:00.000Z","static_fire
_date_unix":1142553600,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":33,"alti
tude":null,"reason":"merlin engine failure"}],"details":"Engine failure at 33 seconds and loss of vehicle","crew":[],"ships":
[],"capsules":[],"payloads":["5eb0e4b5b6c3bb0006eeb1e1"],"launchpad":"5e9e4502f5090995de566f86","flight_number":1,"name":"Falc
onSat","date_utc":"2006-03-24T22:30:00.000Z","date_unix":1143239400,"date_local":"2006-03-25T10:30:00+12:00","date_precisio
n":"hour","upcoming":false,"cores":[{"core":"5e9e289df35918033d3b2623","flight":1,"gridfins":false,"legs":false,"reused":fals
e,"landing_attempt":false,"landing_success":null,"landing_type":null,"landpad":null}],"auto_update":true,"tbd":false,"launch_l
ibrary_id":null,"id":"5eb87cd9ffd86e000604b32a"},{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ship
s":[]},"links":{"patch":{"small":"https://images2.imgbox.com/f9/4a/ZboXReNb_o.png","large":"https://images2.imgbox.com/80/a2/b
```

# Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

https://github.com/armanius87/course/blob/main/1%20course/jupyter-labs-spacex-data-collection-api%20(4).ipynb



```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)
```

```
In [26]: # Use json_normalize meethod to convert the json result into a dataframe
         response = requests.get(static_json_url)
         JsonList = response.json()
         data = pd.json_normalize(JsonList)
```

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head(5)
```

| | static_fire_date_utc | static_fire_date_unix | tbd | net | window | rocket | success | details | crew | ships | capsules | p |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2006-03-17T00:00:00.000Z | 1.142554e+09 | False | False | 0.0 | 5e9d0d95eda69955f709d1eb | False | Engine failure at 33 seconds and loss of vehicle | [] | [] | [] | [5eb0e4b5b6c3bb0006 |

# Data Collection - Scraping

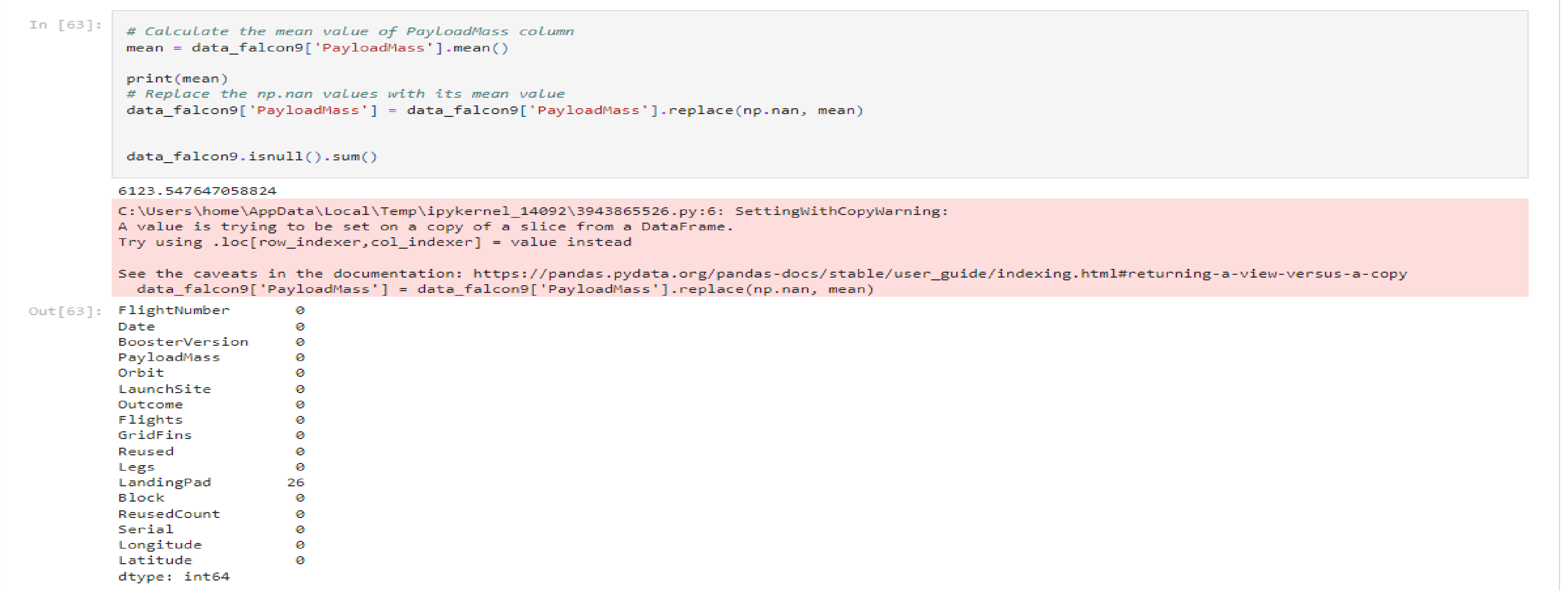
- Present your web scraping process using key phrases and flowcharts

- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose

Place your flowchart of web scraping here

# Data Wrangling

https://github.com/armanius87/course/blob/main/1%20course/jupyter-labs-spacex-data-collection-api%20(4).ipynb



In [63]:

```
# Calculate the mean value of PayloadMass column
mean = data_falcon9['PayloadMass'].mean()

print(mean)
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, mean)


data_falcon9.isnull().sum()
```

```
6123.547647058824
C:\Users\home\AppData\Local\Temp\ipykernel_14092\3943865526.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, mean)
```

Out[63]:

```
FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        0
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

# EDA with Data Visualization

Relationship between Flight Number and Launch Site

Relationship between Payload and Launch Site

Relationship between success rate of each orbit type

Relationship between FlightNumber and Orbit type

Launch success yearly trend

https://github.com/armanius87/course/blob/main/1%20course/jupyter-labs-eda-dataviz.ipynb

# EDA with SQL

Using bullet point format, summarize the SQL queries you performed

Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose

# Build an Interactive Map with Folium

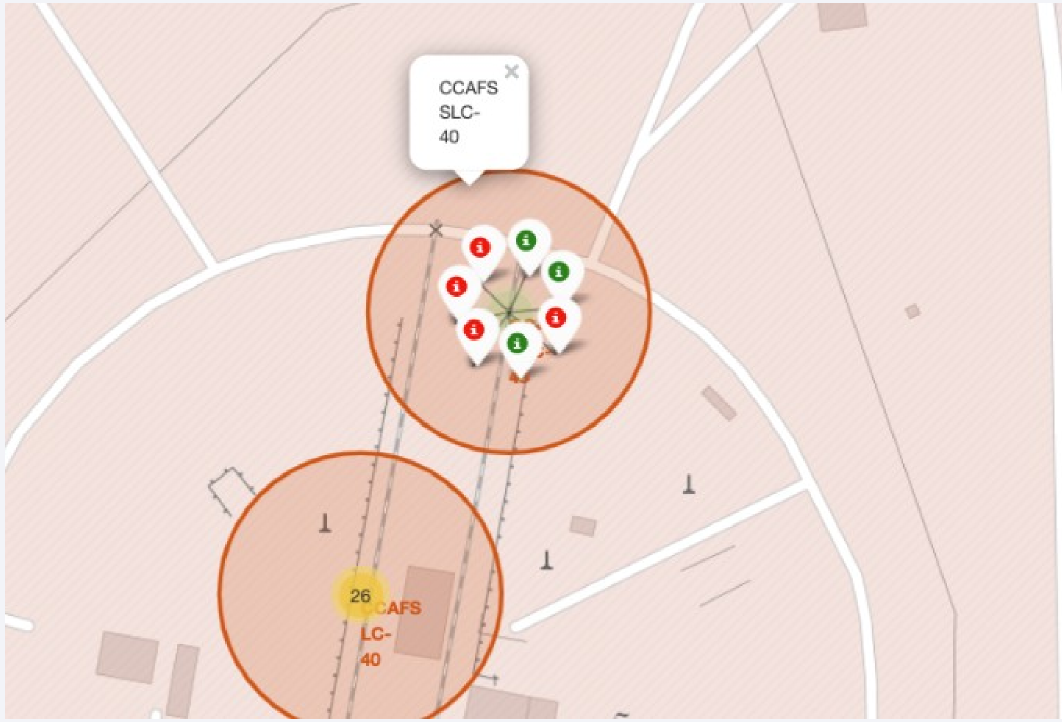
Markers

Circles

Marker_cluster

https://github.com/armanius87/course/blob/main/1%20course/lab_jupyter_launch_site_location.ipynb

# Build a Dashboard with Plotly Dash



https://github.com/armanius87/course/blob/main/1%20course/lab_jupyter_launch_site_location.ipynb

# Predictive Analysis (Classification)

Logistic Regression

Support vector machine

Decision tree classifier

K nearest neighbors

https://github.com/armanius87/course/blob/main/1%20course/SpaceX_Machine%20Learning%20Prediction_Part_5%20(2).ipynb

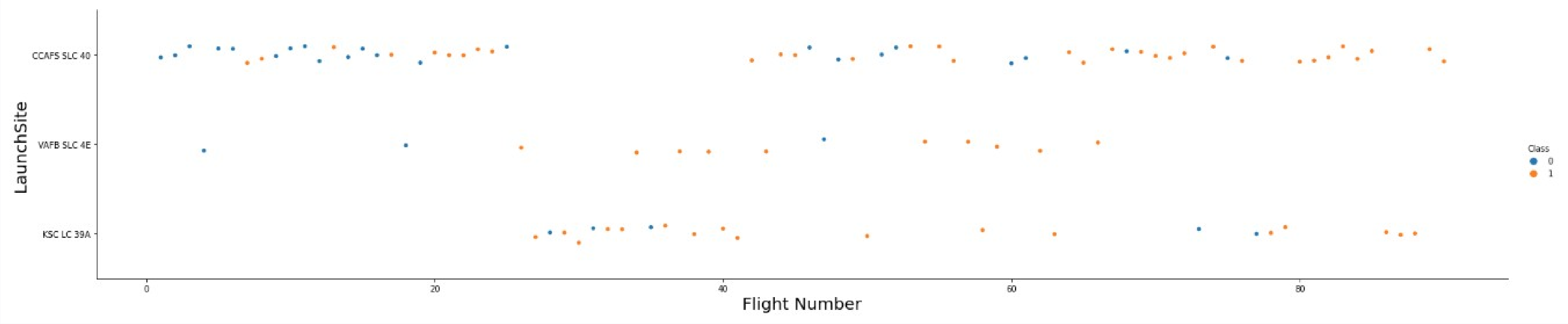
# Results

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

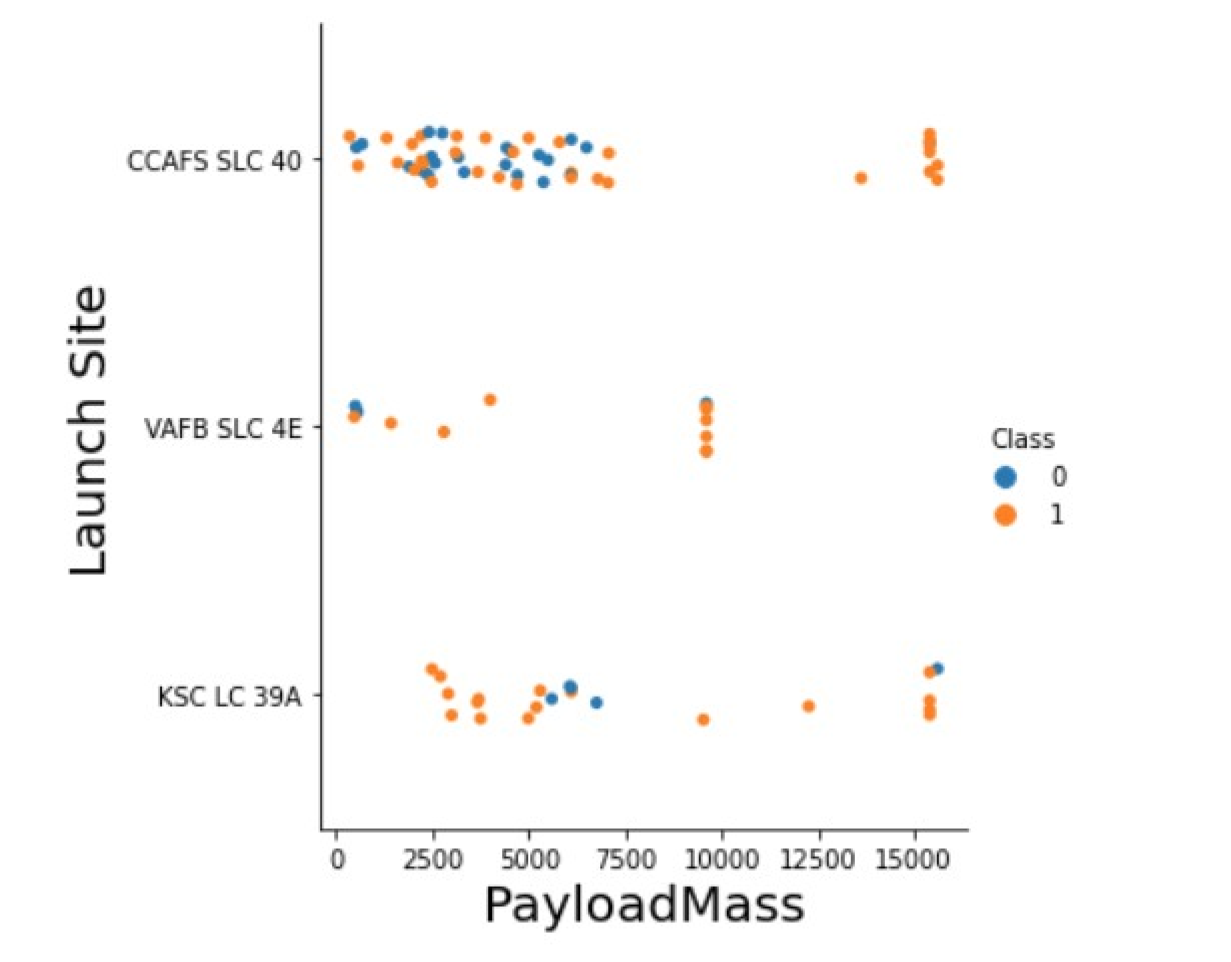
Section 2

# Insights drawn from EDA

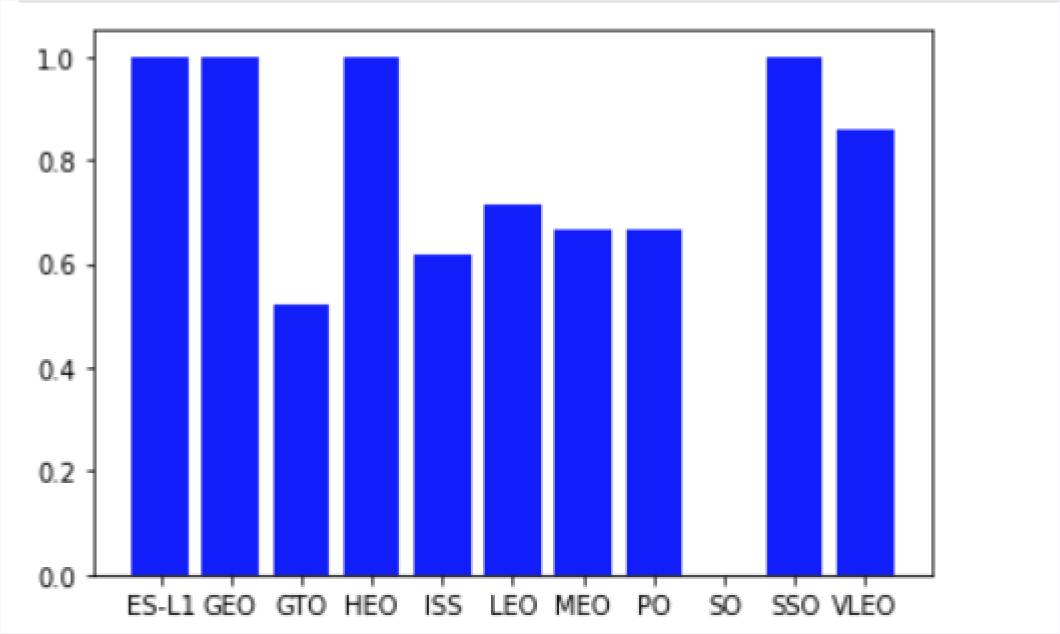
# Flight Number vs. Launch Site



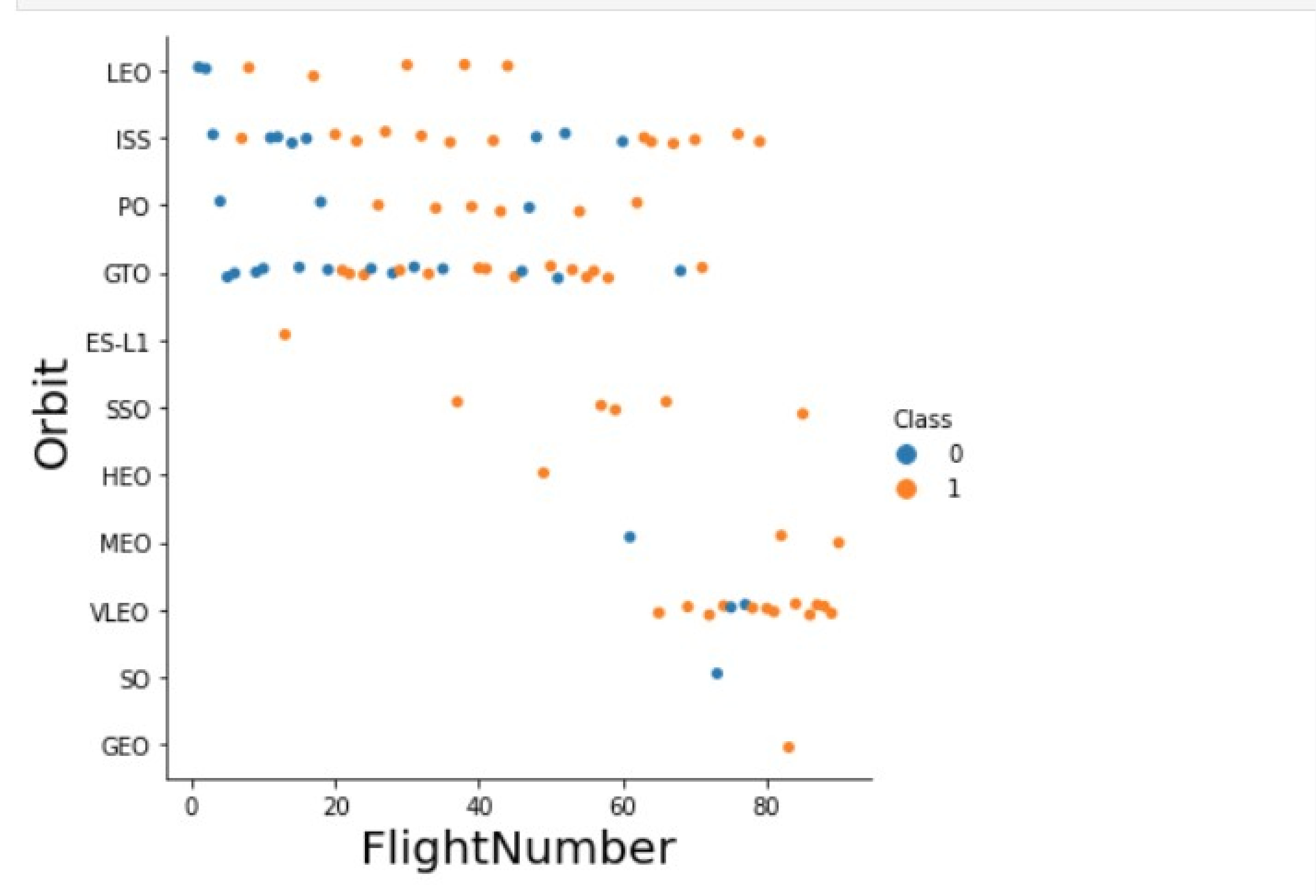
# Payload vs. Launch Site



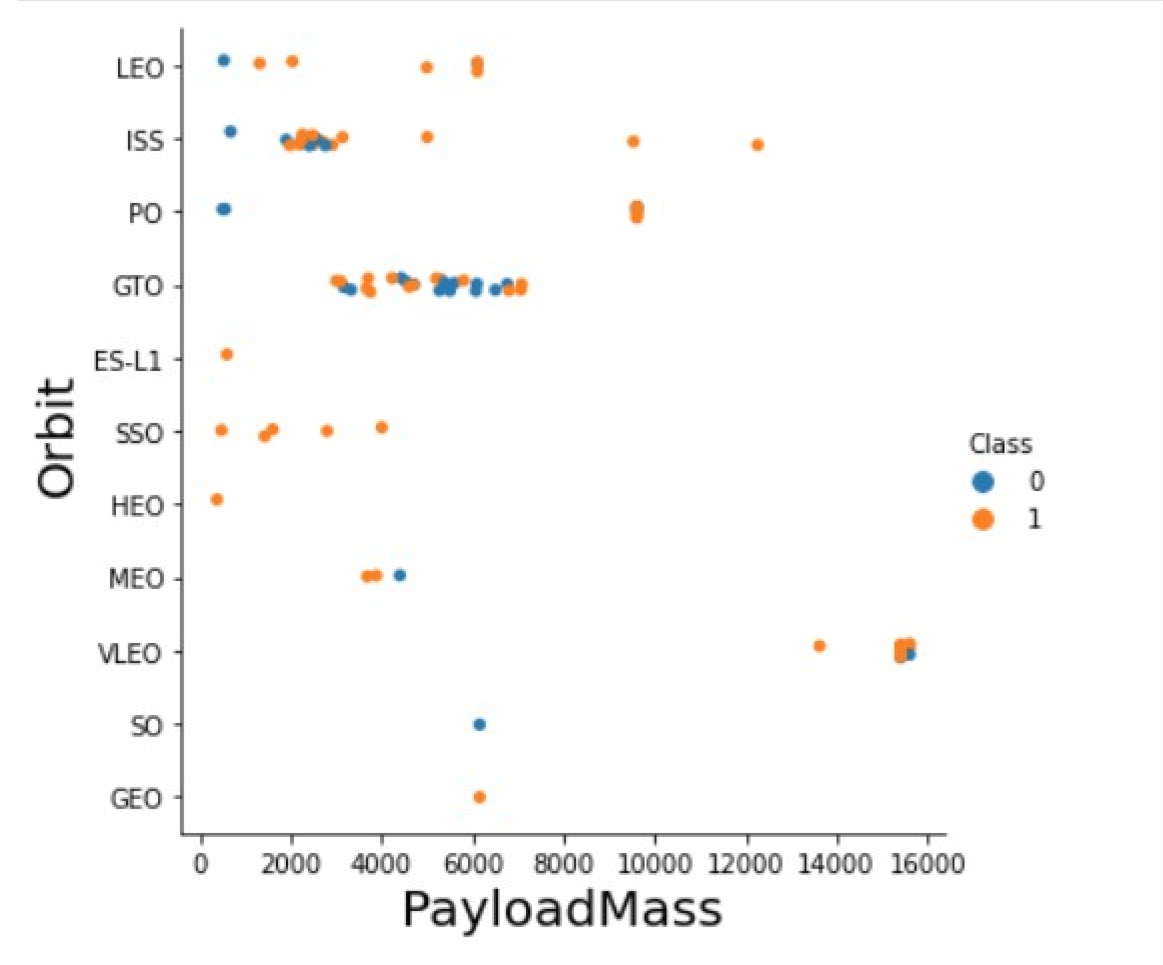
# Success Rate vs. Orbit Type



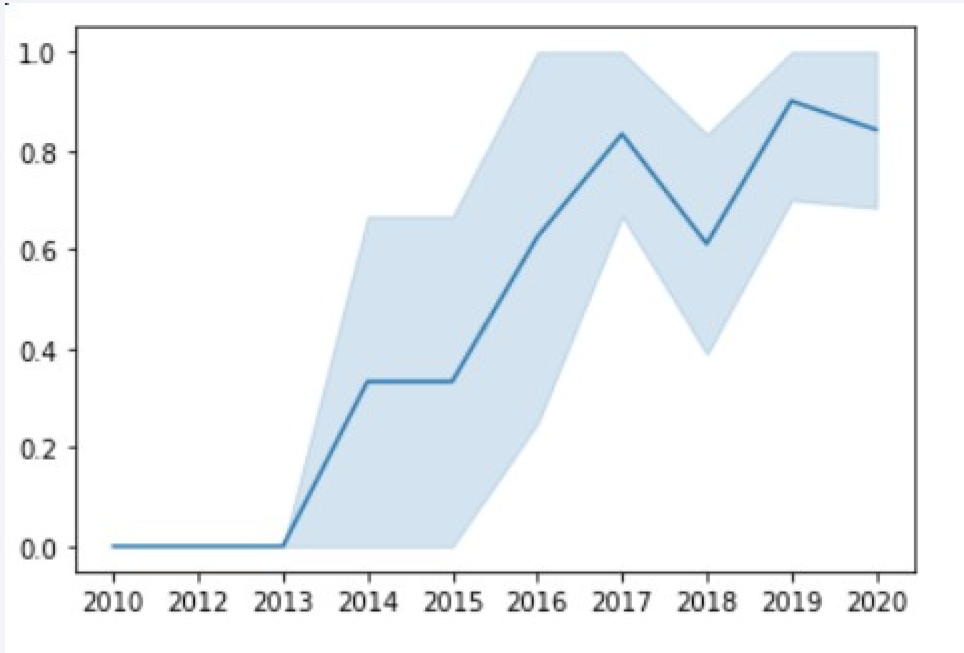
# Flight Number vs. Orbit Type



# Payload vs. Orbit Type



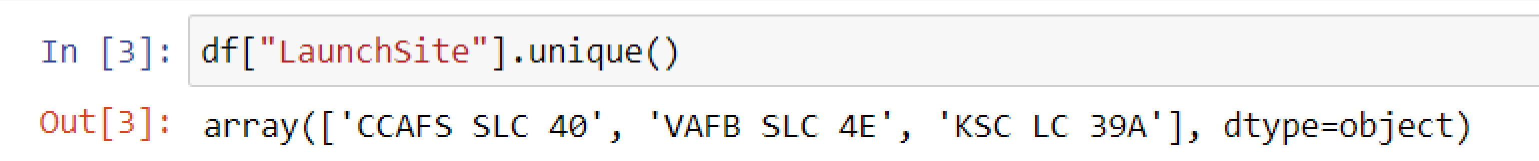
# Launch Success Yearly Trend



# All Launch Site Names

Find the names of the unique launch sites

Present your query result with a short explanation here



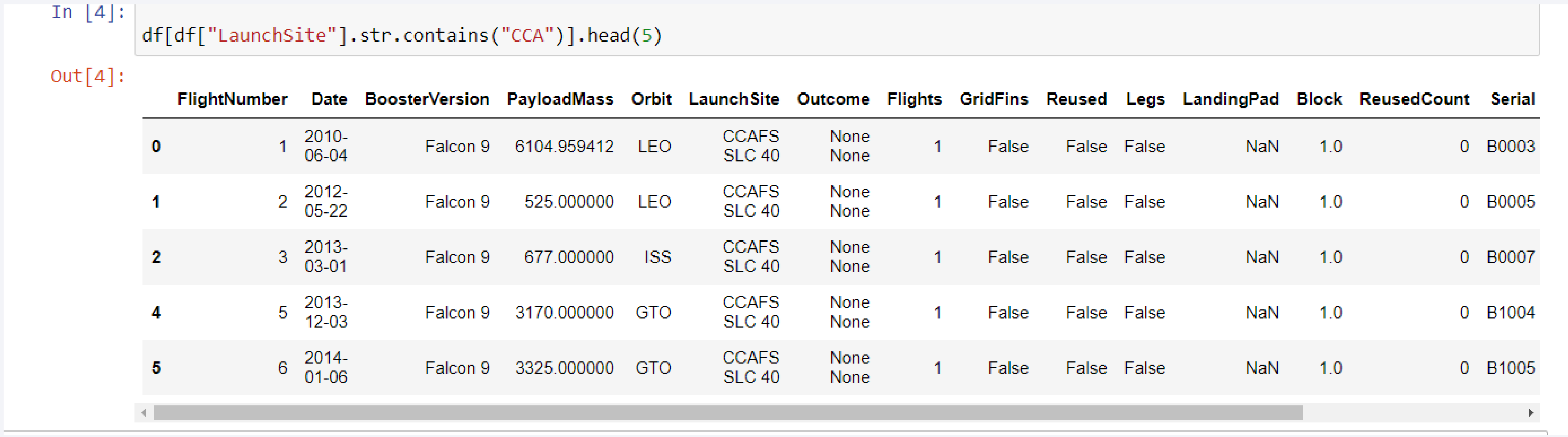
```
In [3]: df["LaunchSite"].unique()

Out[3]: array(['CCAFS SLC 40', 'VAFB SLC 4E', 'KSC LC 39A'], dtype=object)
```

# Launch Site Names Begin with 'CCA'

Find 5 records where launch sites begin with `CCA`

Present your query result with a short explanation here



In [4]:

```
df[df["LaunchSite"].str.contains("CCA")].head(5)
```

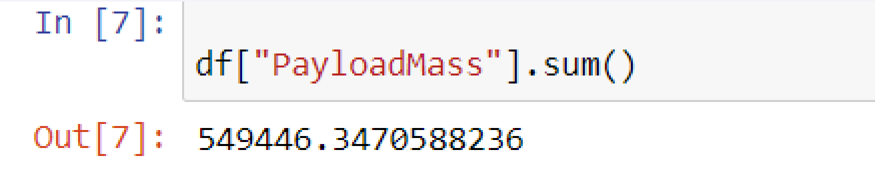
Out[4]:

| | FlightNumber | Date | BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome | Flights | GridFins | Reused | Legs | LandingPad | Block | ReusedCount | Serial |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2010-06-04 | Falcon 9 | 6104.959412 | LEO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN | 1.0 | 0 | B0003 |
| **1** | 2 | 2012-05-22 | Falcon 9 | 525.000000 | LEO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN | 1.0 | 0 | B0005 |
| **2** | 3 | 2013-03-01 | Falcon 9 | 677.000000 | ISS | CCAFS SLC 40 | None None | 1 | False | False | False | NaN | 1.0 | 0 | B0007 |
| **4** | 5 | 2013-12-03 | Falcon 9 | 3170.000000 | GTO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN | 1.0 | 0 | B1004 |
| **5** | 6 | 2014-01-06 | Falcon 9 | 3325.000000 | GTO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN | 1.0 | 0 | B1005 |

# Total Payload Mass

Calculate the total payload carried by boosters from NASA

Present your query result with a short explanation here



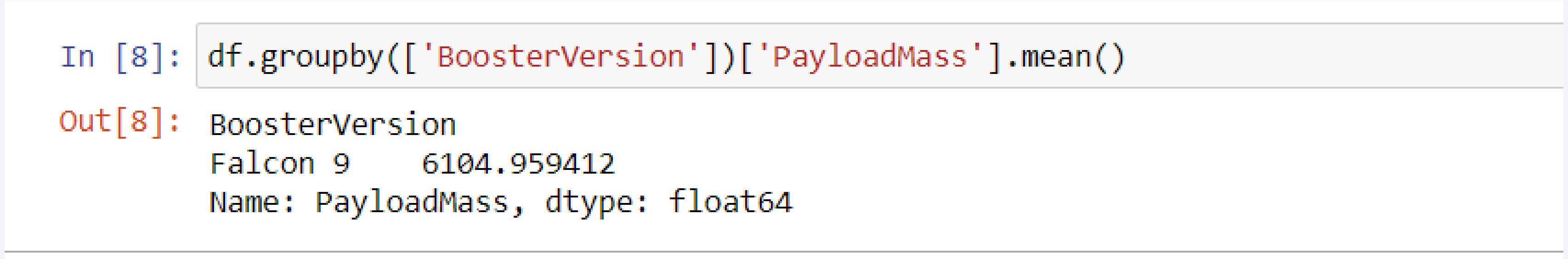
```
In [7]:  df["PayloadMass"].sum()

Out[7]:  549446.3470588236
```

# Average Payload Mass by F9 v1.1

Calculate the average payload mass carried by booster version F9 v1.1

Present your query result with a short explanation here



```
In [8]: df.groupby(['BoosterVersion'])['PayloadMass'].mean()
```

```
Out[8]: BoosterVersion
        Falcon 9    6104.959412
        Name: PayloadMass, dtype: float64
```

# First Successful Ground Landing Date

Find the dates of the first successful landing outcome on ground pad

Present your query result with a short explanation here

# 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

Present your query result with a short explanation here

# Total Number of Successful and Failure Mission Outcomes

Calculate the total number of successful and failure mission outcomes

Present your query result with a short explanation here

# Boosters Carried Maximum Payload

List the names of the booster which have carried the maximum payload mass

Present your query result with a short explanation here

# 2015 Launch Records

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

Present your query result with a short explanation here

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

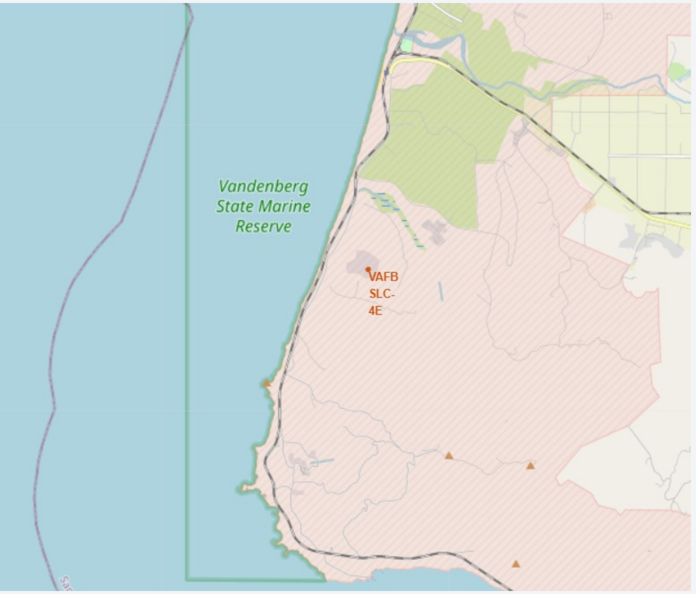
Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

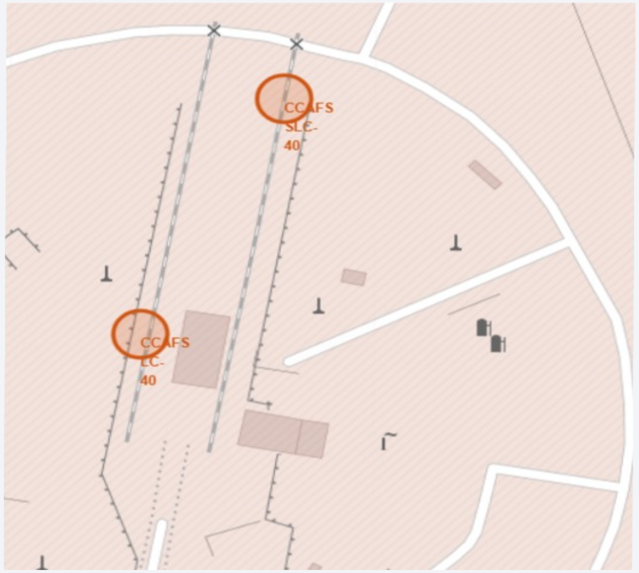
Present your query result with a short explanation here

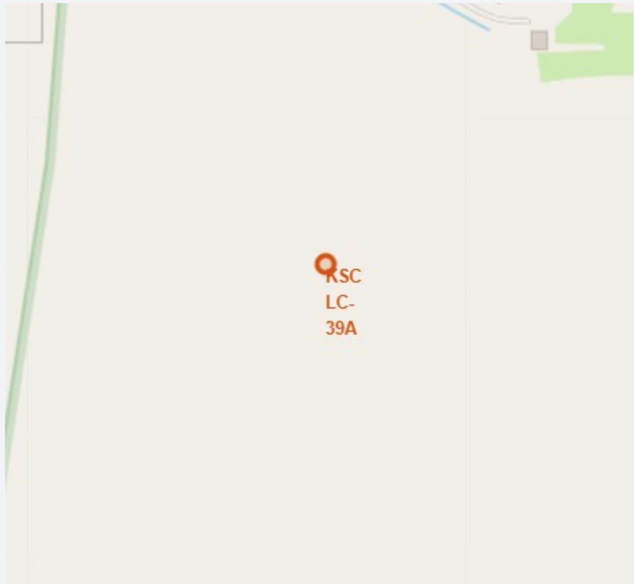
Section 3

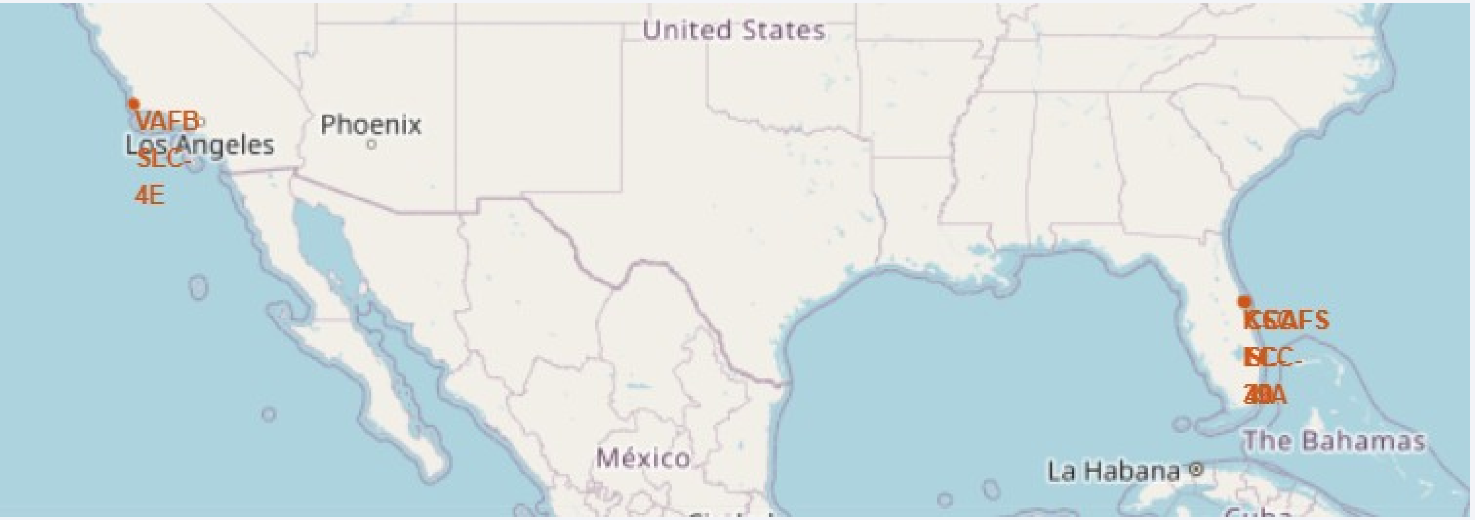
# Launch Sites Proximities Analysis

# Launch Sites

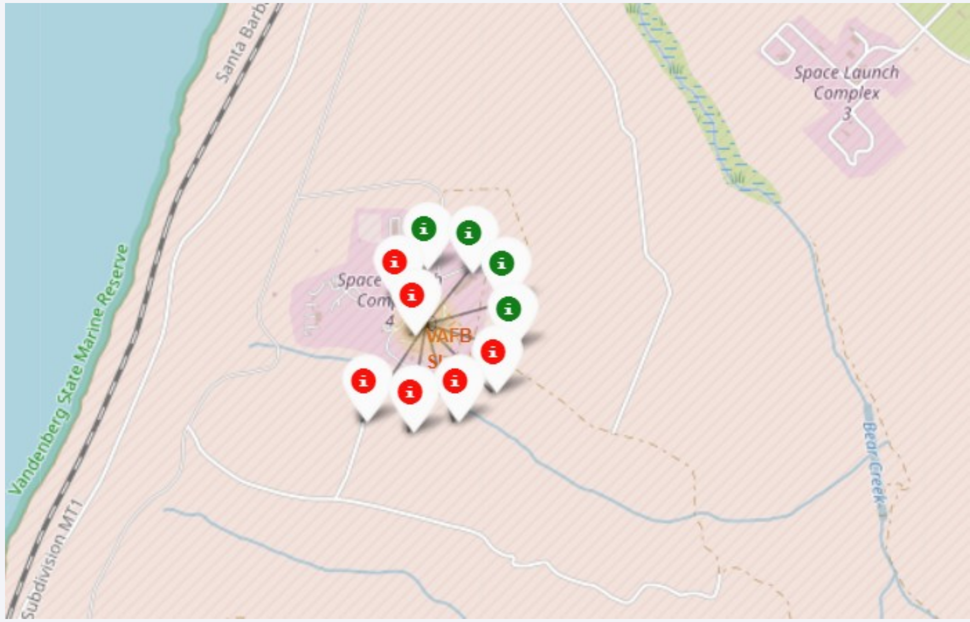


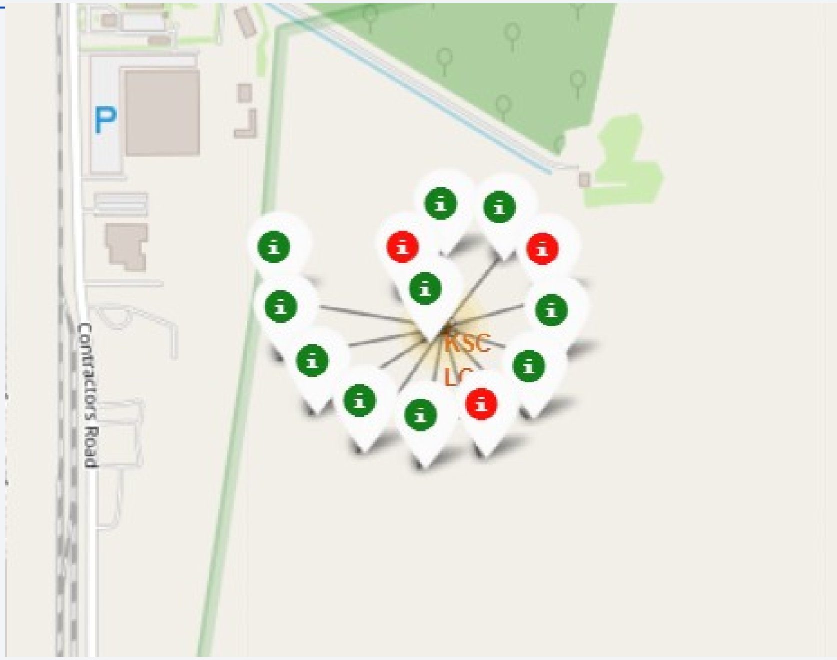


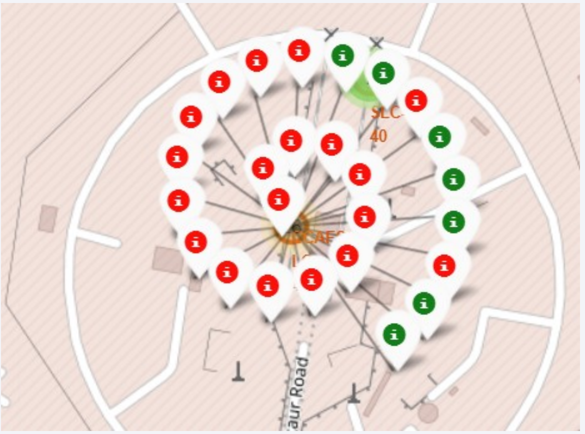




# Launch outcomes

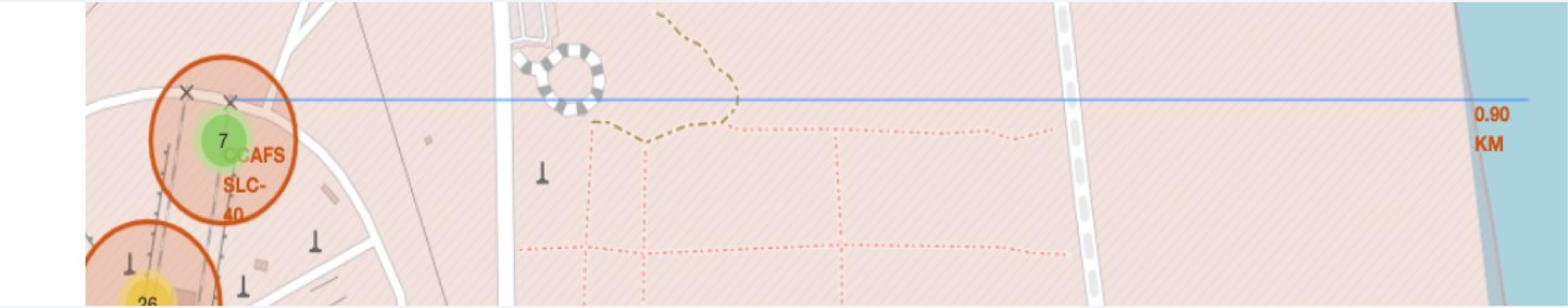






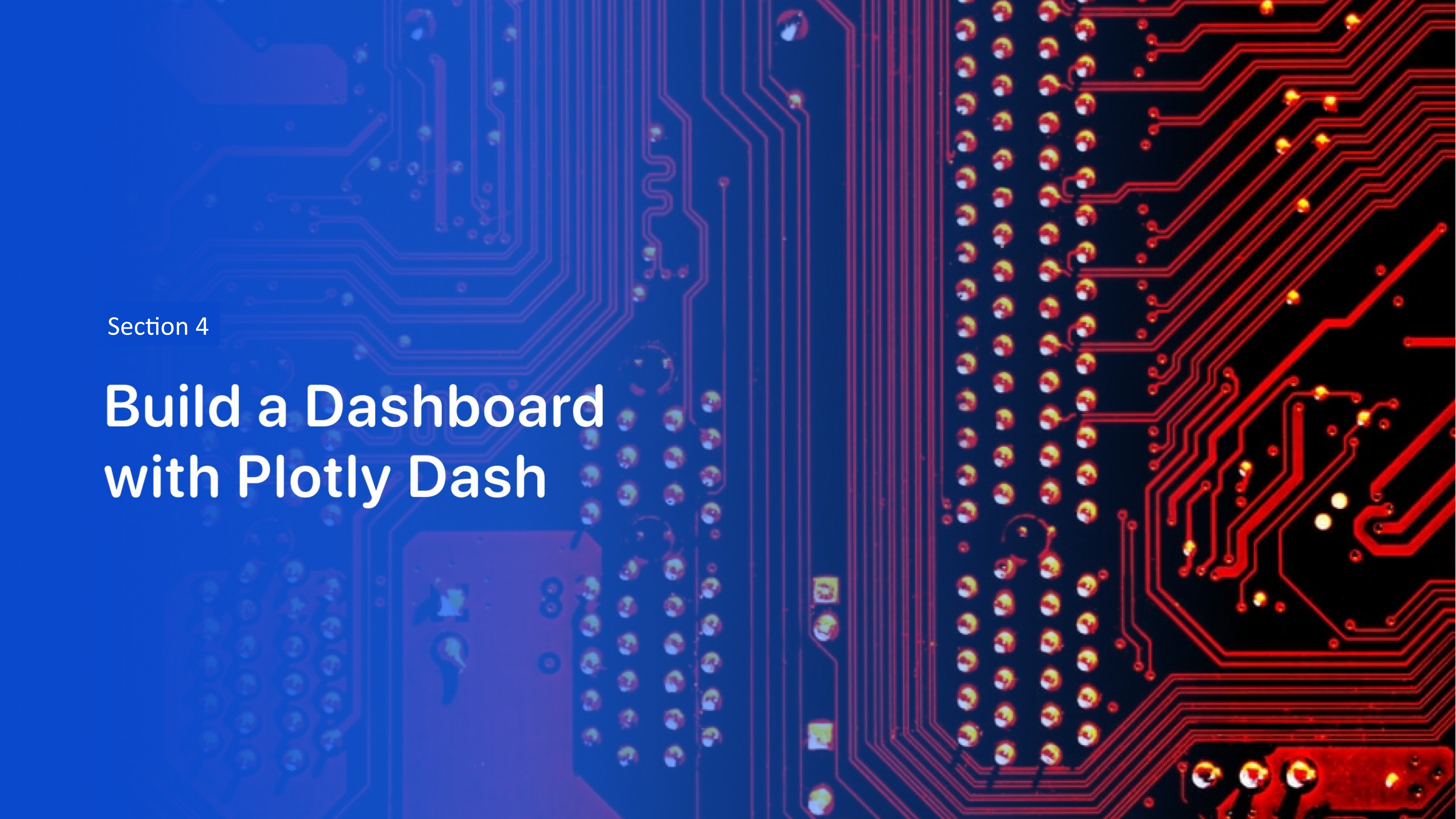
26
CAFS LC-40
ur Road

# Line distance



Section 4

# Build a Dashboard with Plotly Dash



# <Dashboard Screenshot 1>

Replace <Dashboard screenshot 1> title with an appropriate title

Show the screenshot of launch success count for all sites, in a piechart

Explain the important elements and findings on the screenshot

# <Dashboard Screenshot 2>

Replace <Dashboard screenshot 2> title with an appropriate title

Show the screenshot of the piechart for the launch site with highest launch success ratio

Explain the important elements and findings on the screenshot

# <Dashboard Screenshot 3>

Replace <Dashboard screenshot 3> title with an appropriate title

Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

# Predictive Analysis (Classification)

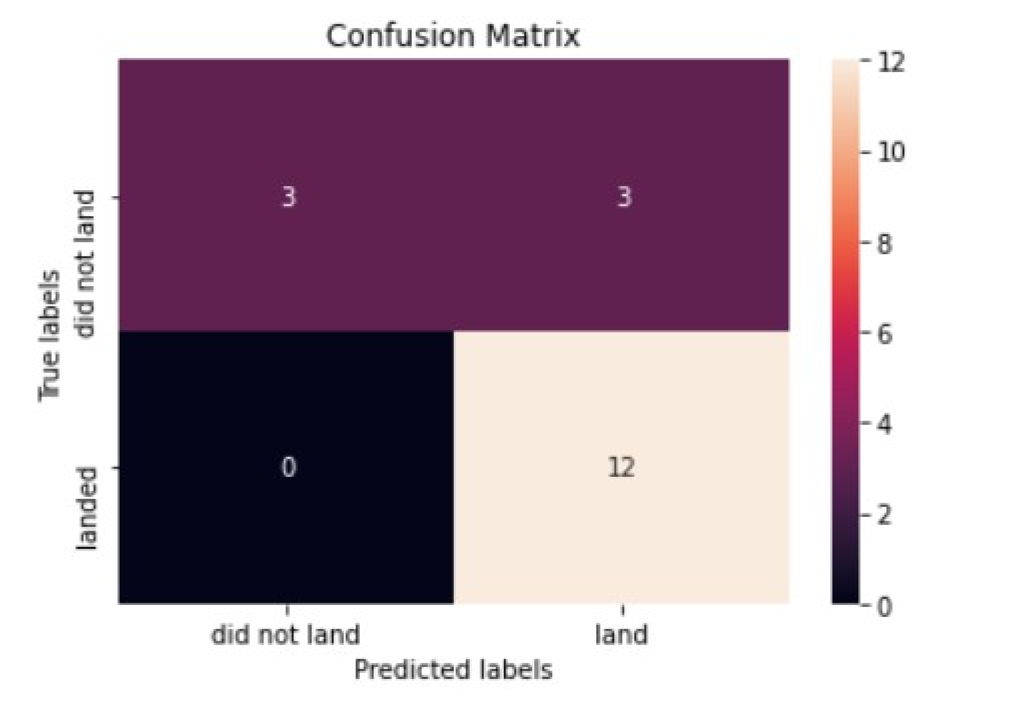
# Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart

- Find which model has the highest classification accuracy

# Confusion Matrix

Decision tree classifier



# Conclusions

Decision tree has provided best accuracy

KsC39 LC39A has most successful launches

KsC39 LC39A shows great sucses at low load as well high loads.

# Appendix

Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!