IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Oleksii Shulzhenko
26 Dec 2021

# 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

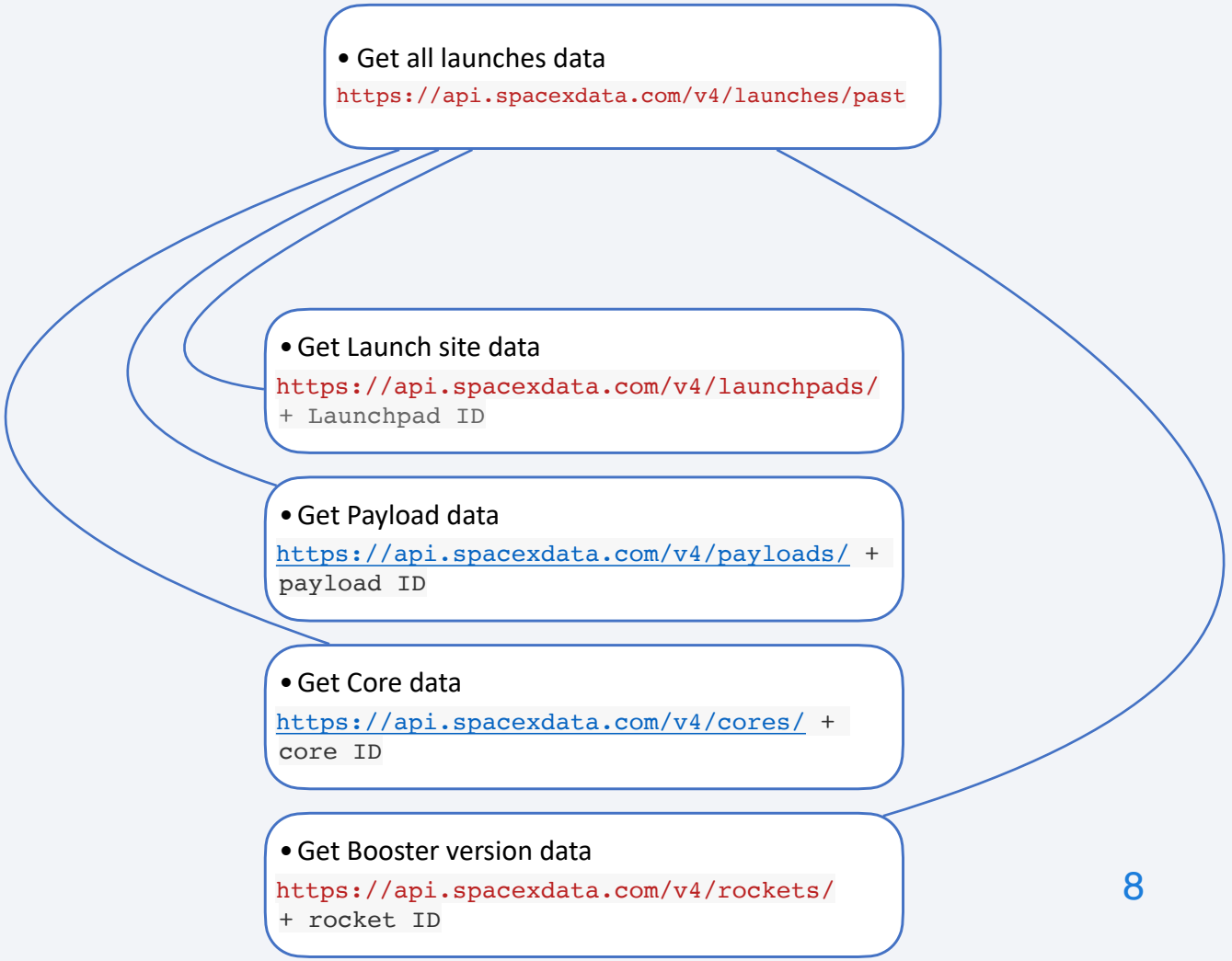
- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

# Data Collection

- Describe how data sets were collected.
- You need to present your data collection process use key phrases and flowcharts

# Data Collection – SpaceX API

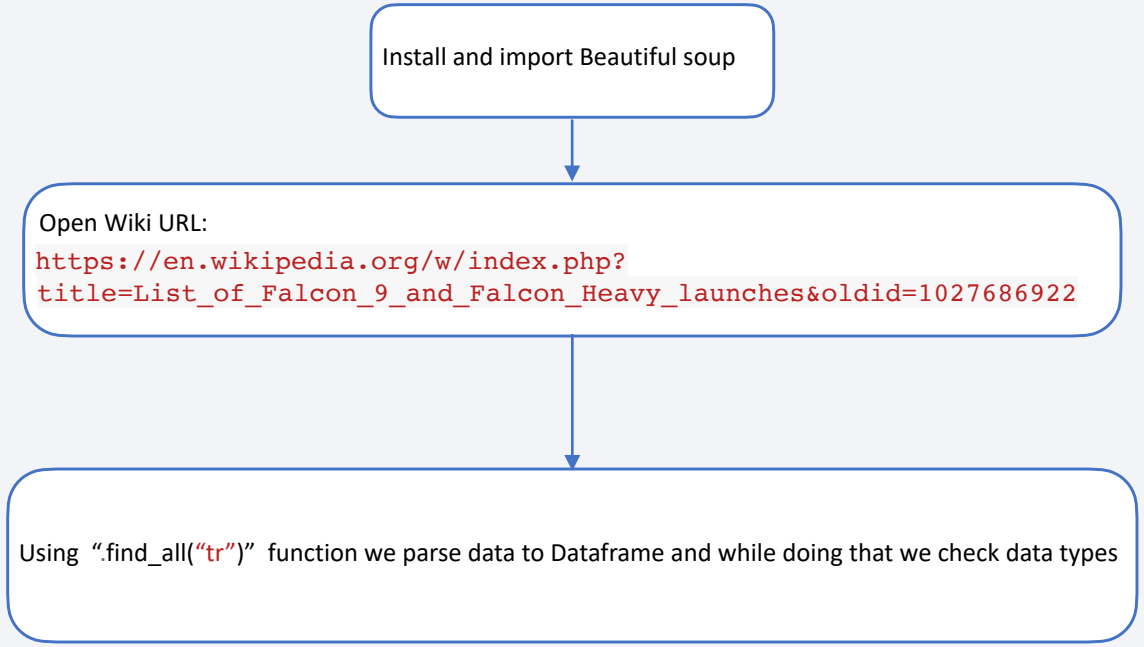
- First we get main launch data and then using subqueries in for-loop we populate data with additional info. After exporting all data we do basic transformation on data types and daters
- GitHub Repo: https://github.com/alexeyshulzhenko/Coursera-IBM-Python-For-Data-Science/blob/master/spacex-data-collection-api.ipynb



# Data Collection - Scraping

- Web Scraping process mainly consists of getting all tables from Wiki page and selecting the second one.

- GitHub Repo: https://github.com/alexeyshulzhenko/Coursera-IBM-Python-For-Data-Science/blob/master/webscraping.ipynb

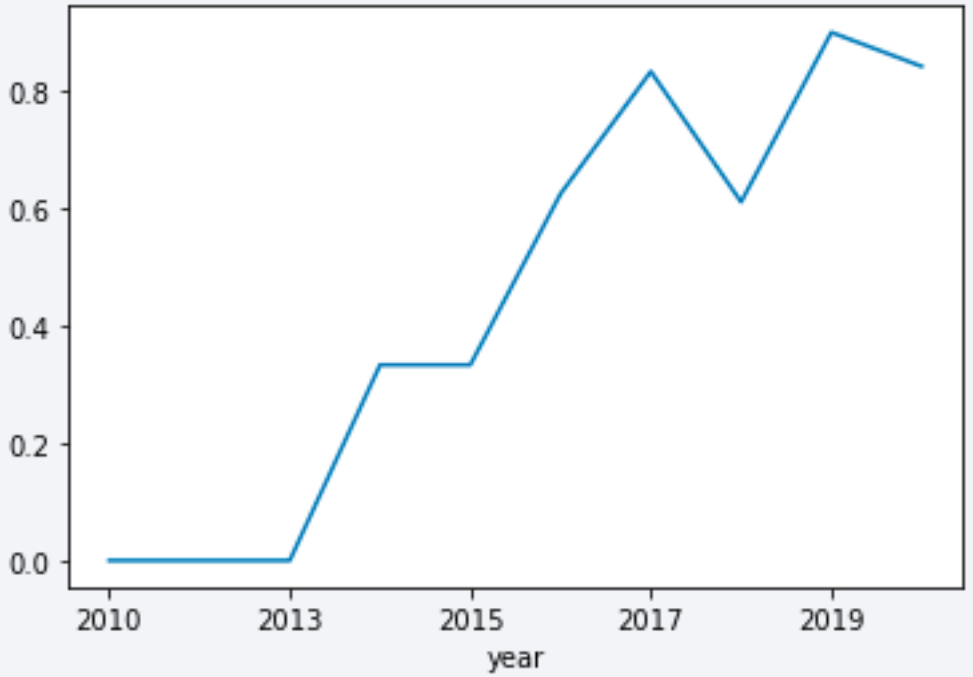


# Data Wrangling

- Data contains:
  - 90 rows
  - 17 columns
- Steps:
  1. Check data types and share of missing values
  2. Replace missing values with column average.
  3. Add Outcomes columns with Boolean value
  4. Check that changes did not corrupted the data
  5. Save
- GitHub URL:
  https://github.com/alexeyshulzhenko/Coursera-IBM-Python-For-Data-Science/blob/master/spacex-Data%20wrangling.ipynb

# EDA with Data Visualization

- The most widely used charts are scatter to see FlightNumber (indicating the continuous launch attempts.) and Payload variables would affect the launch outcome.
- Also I used line chart to see success rate (y Axis) by year (x Axis) (line chart on the left)
- And Bar chart to check success rate by orbit.
- GitHub URL: https://github.com/alexeyshulzhenko/Coursera-IBM-Python-For-Data-Science/blob/master/eda-dataviz.ipynb



# EDA with SQL

- I used DB2 as DB engine and sqlalchemy for performing queries.

- GitHub URL: https://github.com/alexeyshulzhenko/Coursera-IBM-Python-For-Data-Science/blob/master/eda-sql-coursera.ipynb

# Build an Interactive Map with Folium

- Folium is an amazing tool to visualise objects on map. I used this tool to visualise Lunch Sites as circles on map, labels to add info about those circles and Lines to measure distance between Such Sites and distance from launch sites to the ocean and nearest roads.
- GitHub URL: https://github.com/alexeyshulzhenko/Coursera-IBM-Python-For-Data-Science/blob/master/Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

# Build a Dashboard with Plotly Dash

- Plotly Dash appeared to be useful for visualisation metrics with dynamic filters by Year and Lunch Site.
- The most used visualisation types were Line, Bar and Pie charts.
- GitHub URL: https://github.com/alexeyshulzhenko/Coursera-IBM-Python-For-Data-Science/blob/master/spacex_dash_app.py

# Predictive Analysis (Classification)

- I used sklearn library
- Before modelling I had to convert text values to numbers by creating a column for the class
- Then I used StandardScaler() to standardise data.
- Next step was splitting data to test and training data.
- The last step was to try several models (LogisticRegression, support vector machine object, DecisionTreeClassifier and KNeighborsClassifier()) and GridSearchCV to choose optimal parameters. Each model was scored with method .score(X_test, Y_test) and in the end chooses the optimal one

- GitHub URL: https://github.com/alexeyshulzhenko/Coursera-IBM-Python-For-Data-Science/blob/master/Machine%20Learning%20Prediction.ipynb

# Results

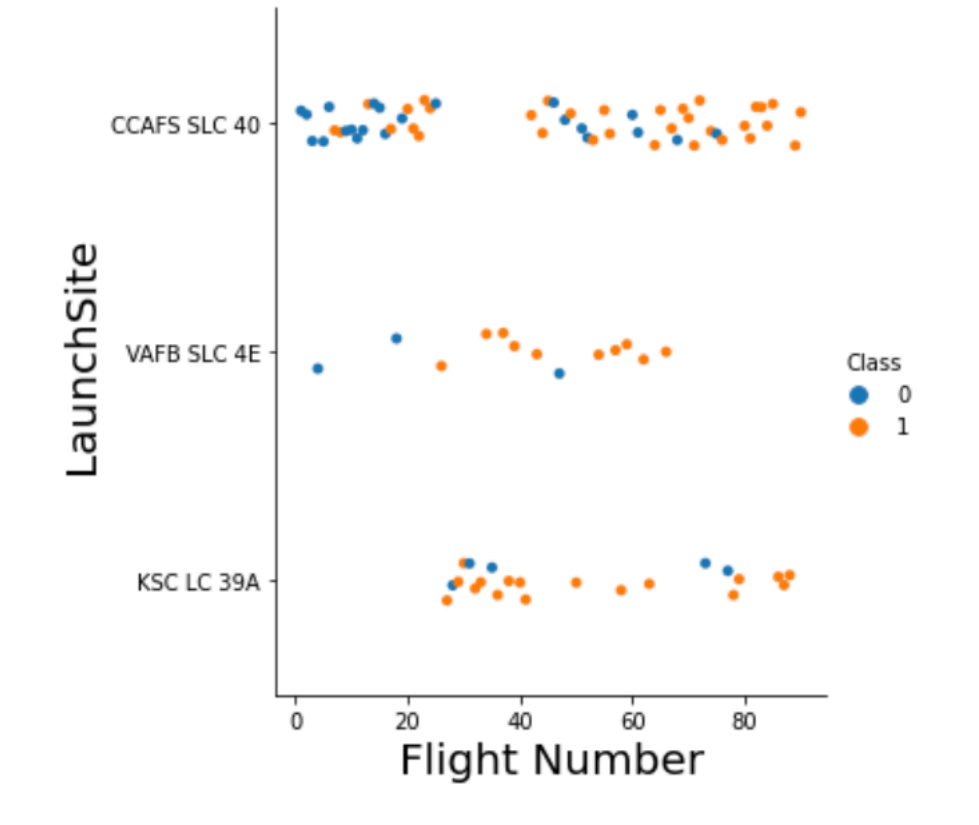
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

# Insights drawn from EDA

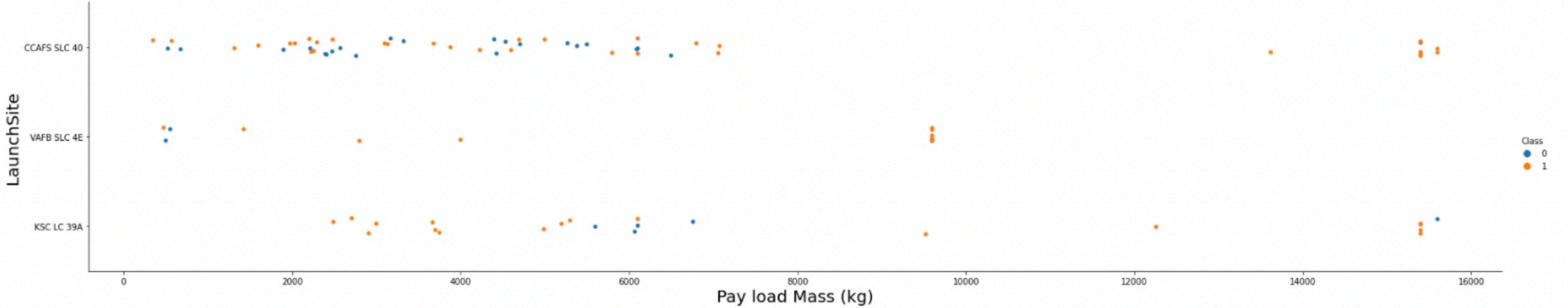
# Flight Number vs. Launch Site

- On this chart we can see, that the most used launch site is CCAFS SLC 40.
- CCAFS SLC 40 has also the largest number of failed launches (class 0).
- KSC LS 39A is the newest Launch Site.



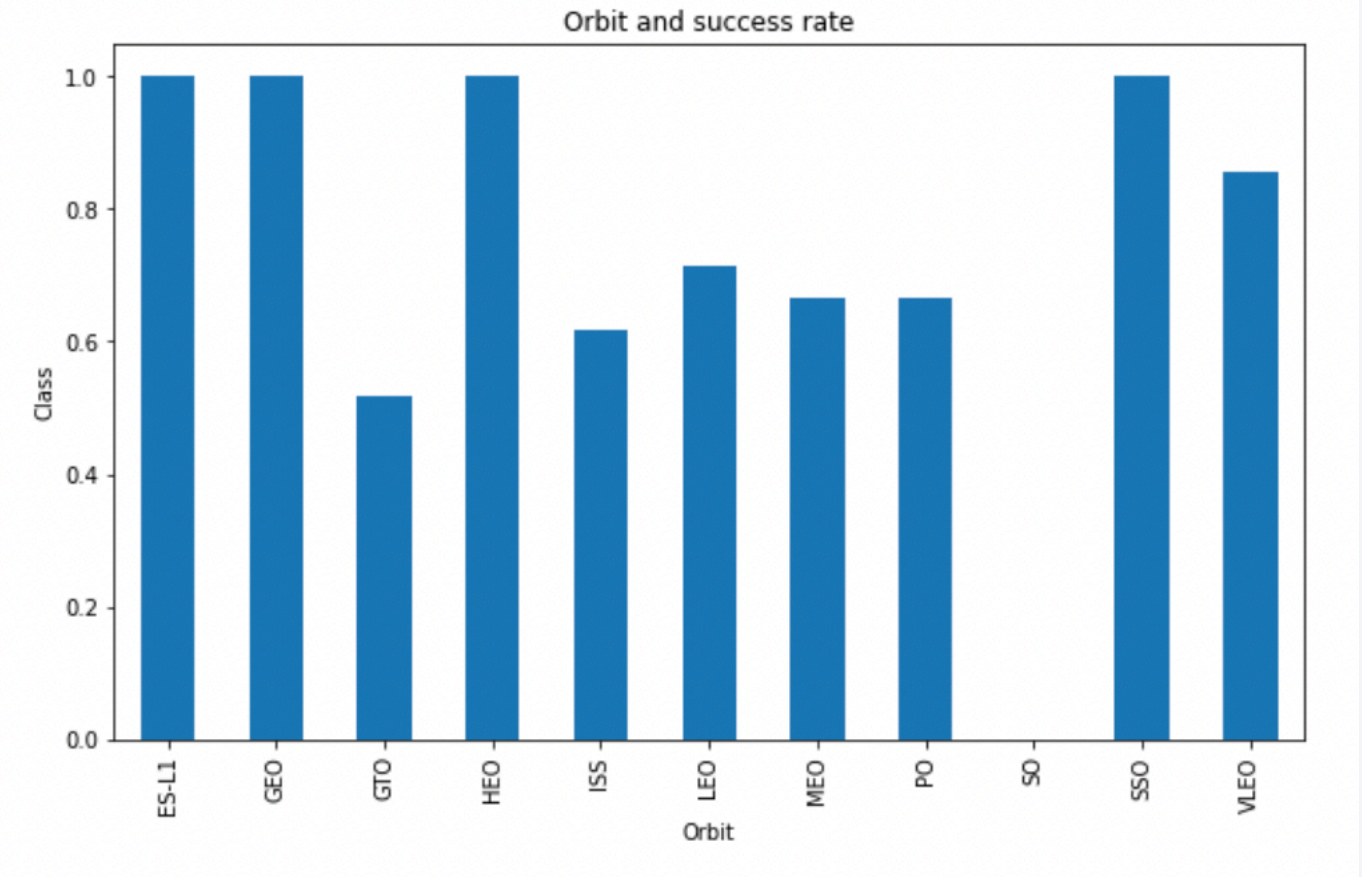
# Payload vs. Launch Site

- Here we can see, that all 3 Launch Sites are used for both light and heavy payloads.
- Also we can see that KSC LS 39A was never used for payload below 2000



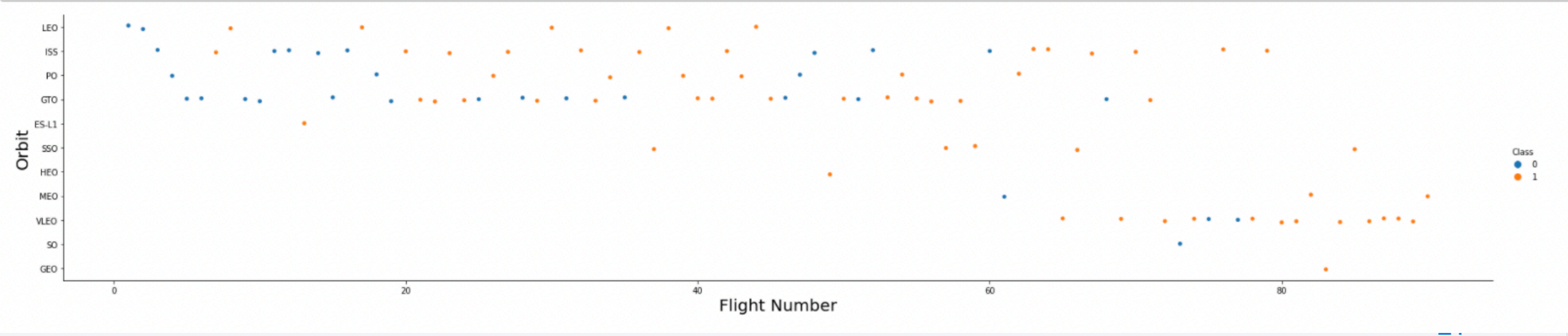
# Success Rate vs. Orbit Type

- ES-L1, GEO, HEO and SSO are having the absolute success rates.
- GTO have the lowest Success rate.



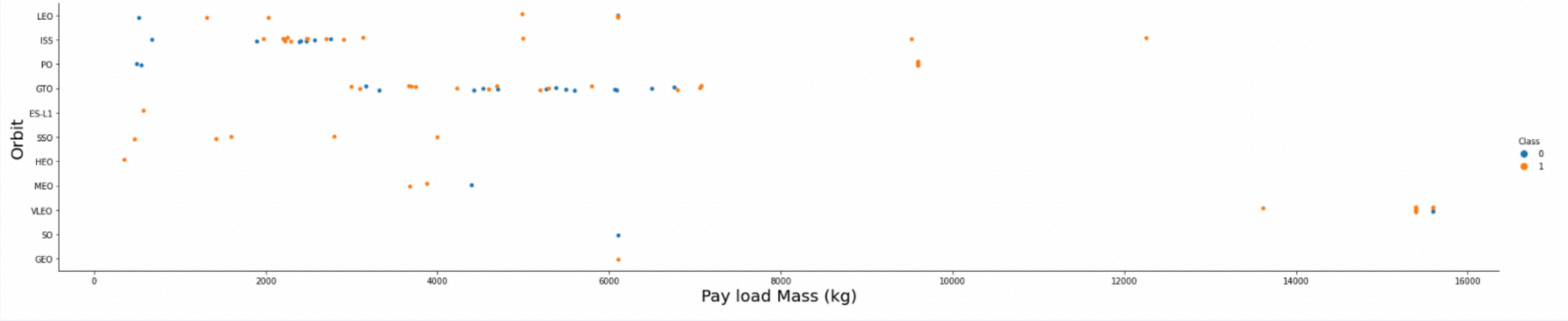
# Flight Number vs. Orbit Type

- Here we can see, that after flight number 60 priority has changed from orbits LEO, ISS, PO, GTO and ES-L1 to orbits GEO, SO, VLEO, MEO and HEO



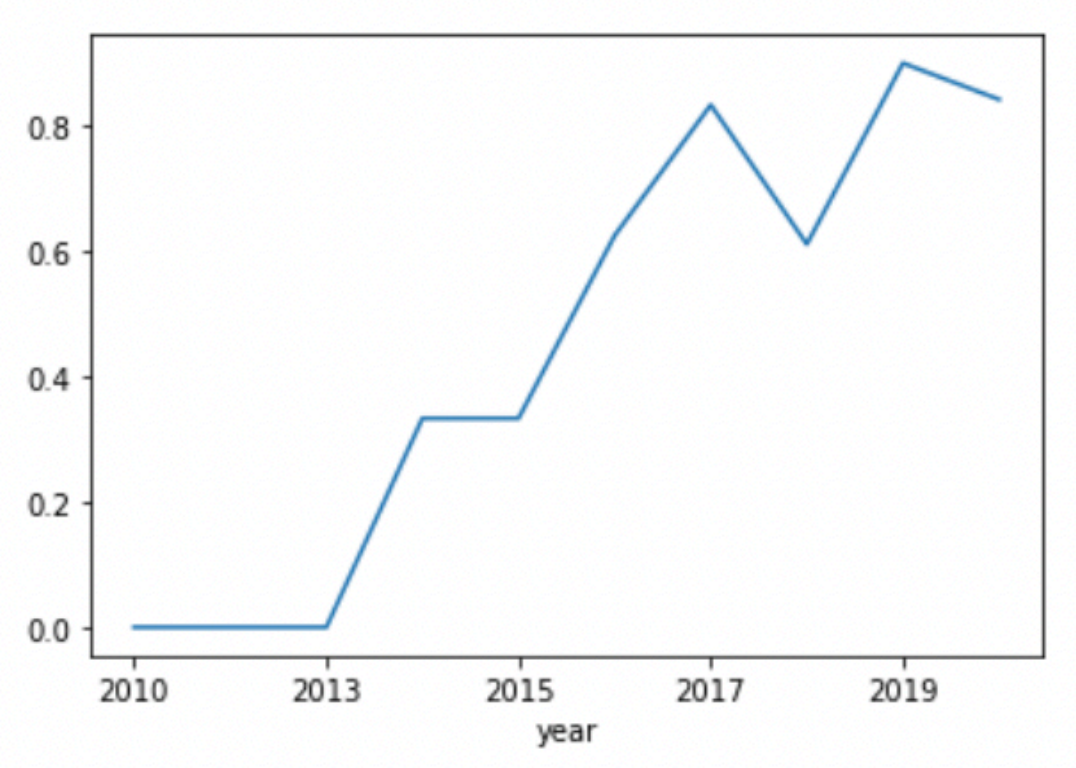
# Payload vs. Orbit Type

- The heaviest payload traveled to VLEO orbit.
- ISS orbit mostly have payload weight between 2000 and 4000
- GTO orbit ranges from 2500 to 6500



# Launch Success Yearly Trend

- The longer rocket exists (in years) - the better success rate it has.
- Y-axis - Avg. success rate
- X-axis - Year



# All Launch Site Names

- Just a bit SQL Magic and simple query

```
%sql SELECT DISTINCT launch_site FROM SPACEXTBL
```

```
 * ibm_db_sa://jsh96667:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:3228
6/bludb
Done.
```

| launch_site |
| --- |
| CCAFS LC-40 |
| CCAFS SLC-40 |
| KSC LC-39A |
| VAFB SLC-4E |

# Launch Site Names Begin with 'CCA'

```
%sql SELECT * from SPACEXTBL WHERE launch_site LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://jsh96667:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

| 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 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 00:35:00 | 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

- I think I actually should add column name, but it was not required and I am too lazy, sorry.

```
%sql SELECT SUM(payload_mass__kg_) from SPACEXTBL WHERE customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://jsh96667:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:3228
6/bludb
Done.
```

| 1 |
|---|
| 45596 |

# Average Payload Mass by F9 v1.1

- Simple SQL Magic Query as well.

```
%sql SELECT AVG(payload_mass__kg_) from SPACEXTBL WHERE booster_version LIKE 'F9 v1.1%'
```

```
 * ibm_db_sa://jsh96667:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:3228
6/bludb
Done.
```

| 1 |
| --- |
| 2534 |

# First Successful Ground Landing Date

- There are multiple landing outcomes, so it is important to filter only Ground pad and Success landings.

```
%sql SELECT MIN(DATE) FROM SPACEXTBL WHERE landing__outcome = 'Success (ground pad)'
```

```
 * ibm_db_sa://jsh96667:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:3228
6/bludb
Done.
```

| 1 |
|---|
| 2015-12-22 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

- Where clause

```
%sql SELECT booster_version FROM SPACEXTBL WHERE landing__outcome = 'Success (drone ship)' AND payload_mass__kg_ > 400
0 AND payload_mass__kg_ < 6000
```

```
 * ibm_db_sa://jsh96667:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:3228
6/bludb
Done.
```

| booster_version |
|---|
| F9 FT B1022 |
| F9 FT B1026 |
| F9 FT B1021.2 |
| F9 FT B1031.2 |

# Total Number of Successful and Failure Mission Outcomes

- There were only one in flight failure and 1 success mission with unclear payload status.

```
%sql SELECT mission_outcome, count(*) FROM SPACEXTBL GROUP BY mission_outcome
```

```
 * ibm_db_sa://jsh96667:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:3228
6/bludb
Done.
```

| mission_outcome | 2 |
|---|---|
| Failure (in flight) | 1 |
| Success | 99 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload

***List the names of the booster_versions which have carried the maximum payload mass. Use a subquery***

```
%%sql

SELECT booster_version FROM SPACEXTBL WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXTBL)
```

```
 * ibm_db_sa://jsh96667:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:3228
6/bludb
Done.
```

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

- Did not expected that the result was so big.

# 2015 Launch Records

```
%%sql

SELECT landing__outcome, booster_version, launch_site FROM SPACEXTBL WHERE YEAR(DATE) = 2015 AND landing__outcome = 'F
ailure (drone ship)'
```

```
 * ibm_db_sa://jsh96667:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:3228
6/bludb
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

```
%%sql

SELECT landing__outcome, count(*) AS f_num
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY landing__outcome
ORDER BY f_num DESC
```

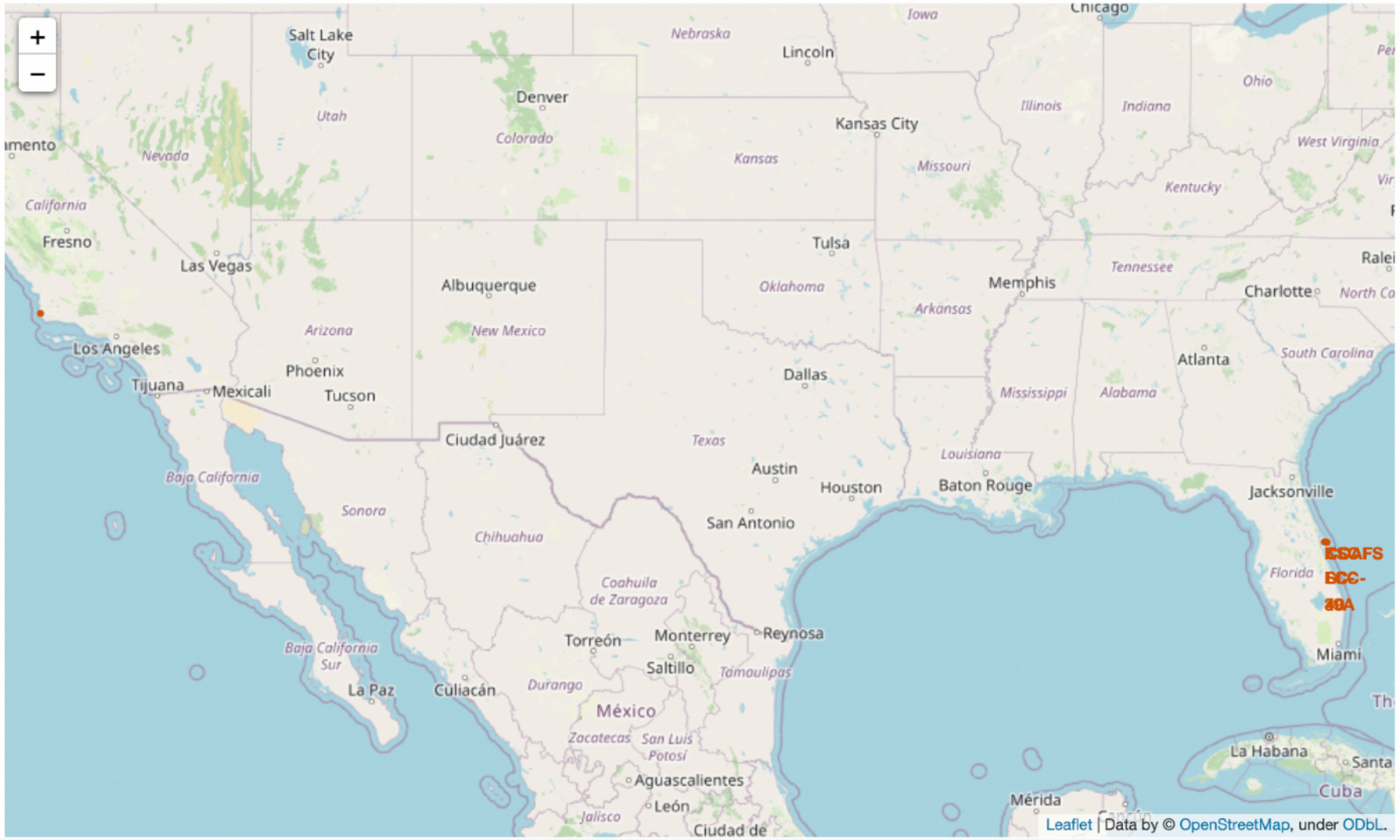
```
 * ibm_db_sa://jsh96667:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:3228
6/bludb
Done.
```

| landing__outcome | f_num |
|---|---|
| No attempt | 10 |
| Failure (drone ship) | 5 |
| Success (drone ship) | 5 |
| Controlled (ocean) | 3 |
| Success (ground pad) | 3 |
| Failure (parachute) | 2 |
| Uncontrolled (ocean) | 2 |
| Precluded (drone ship) | 1 |

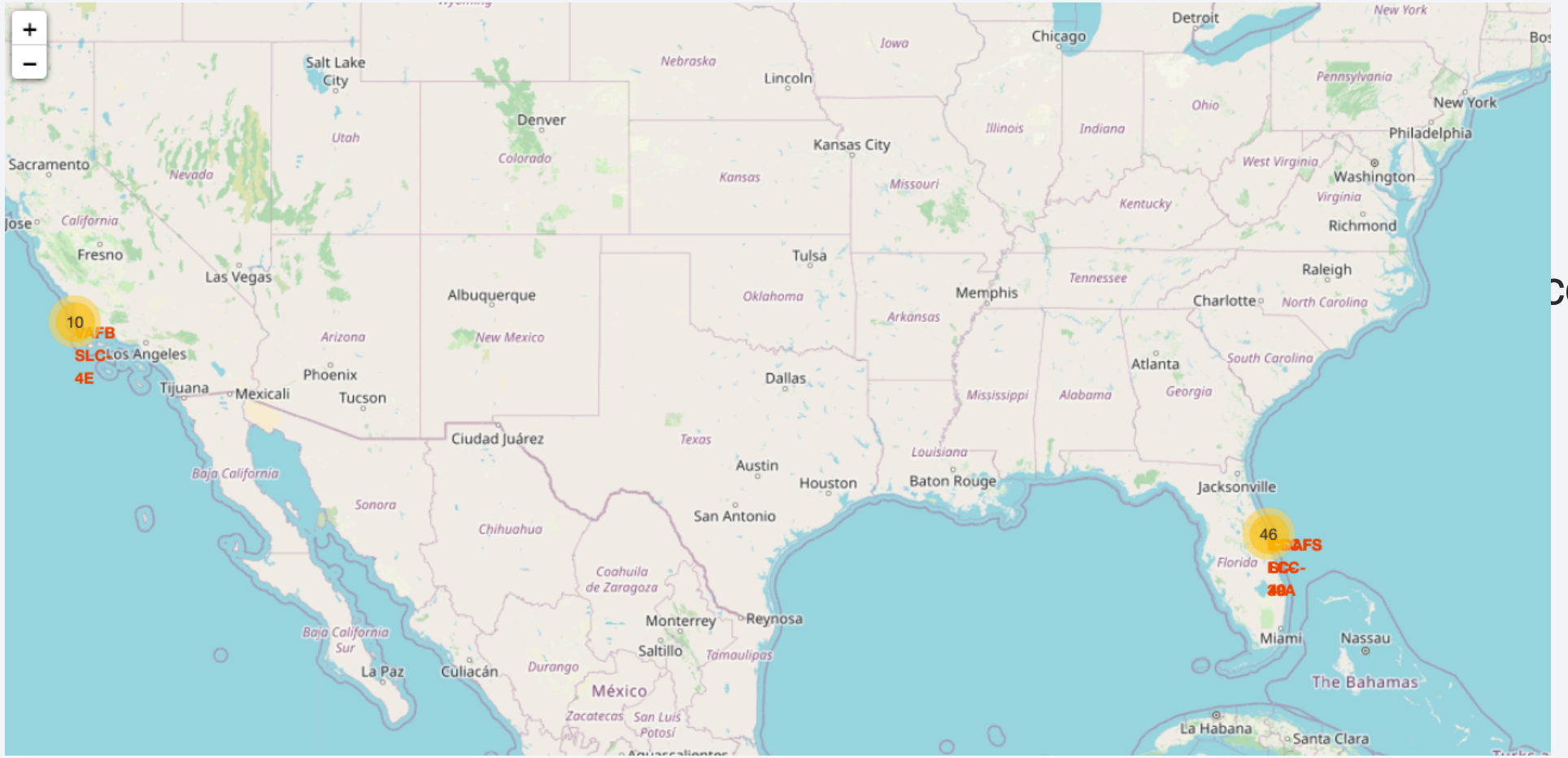
Section 4

# Launch Sites Proximities Analysis

# launch sites on a map

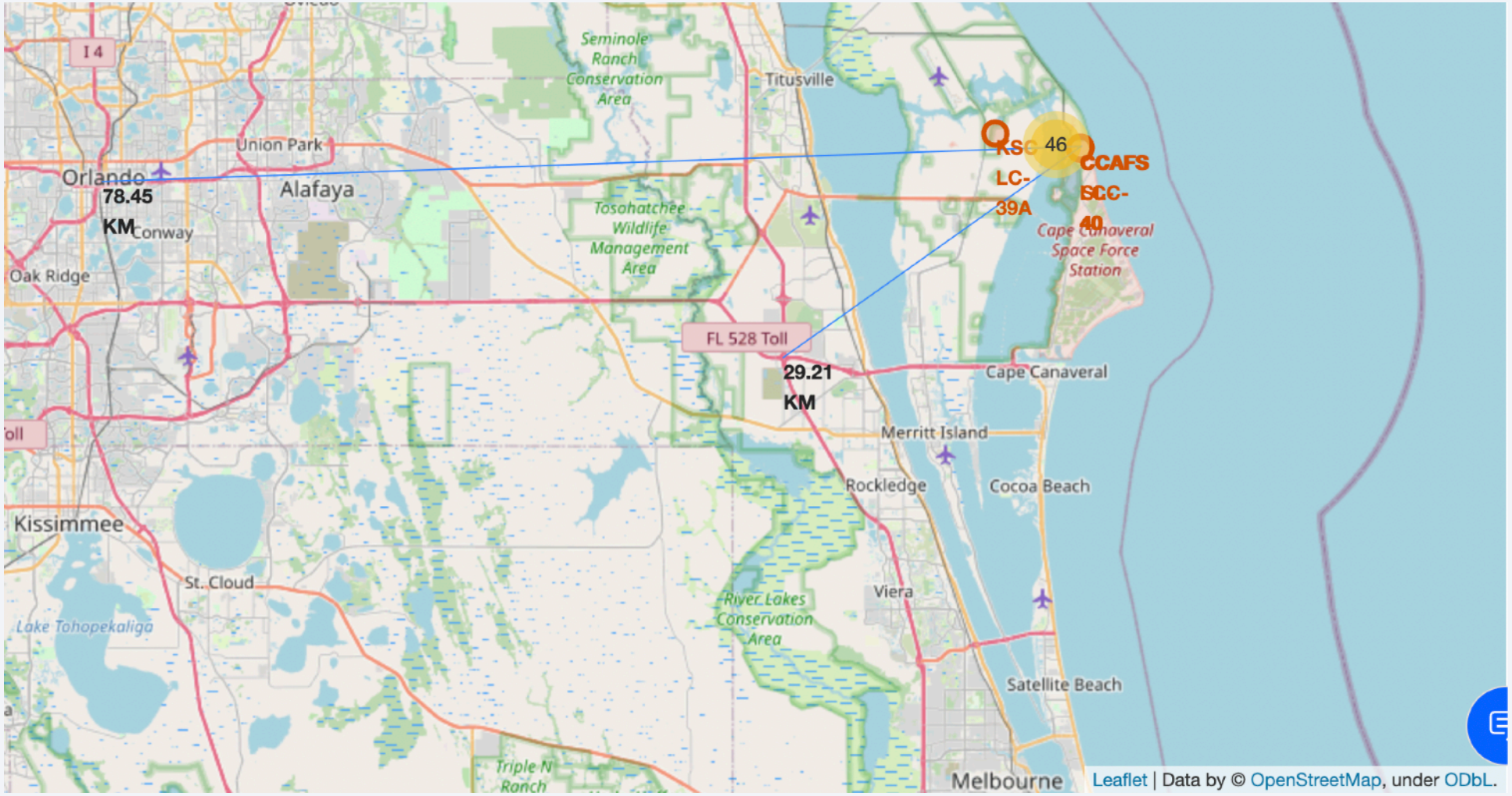


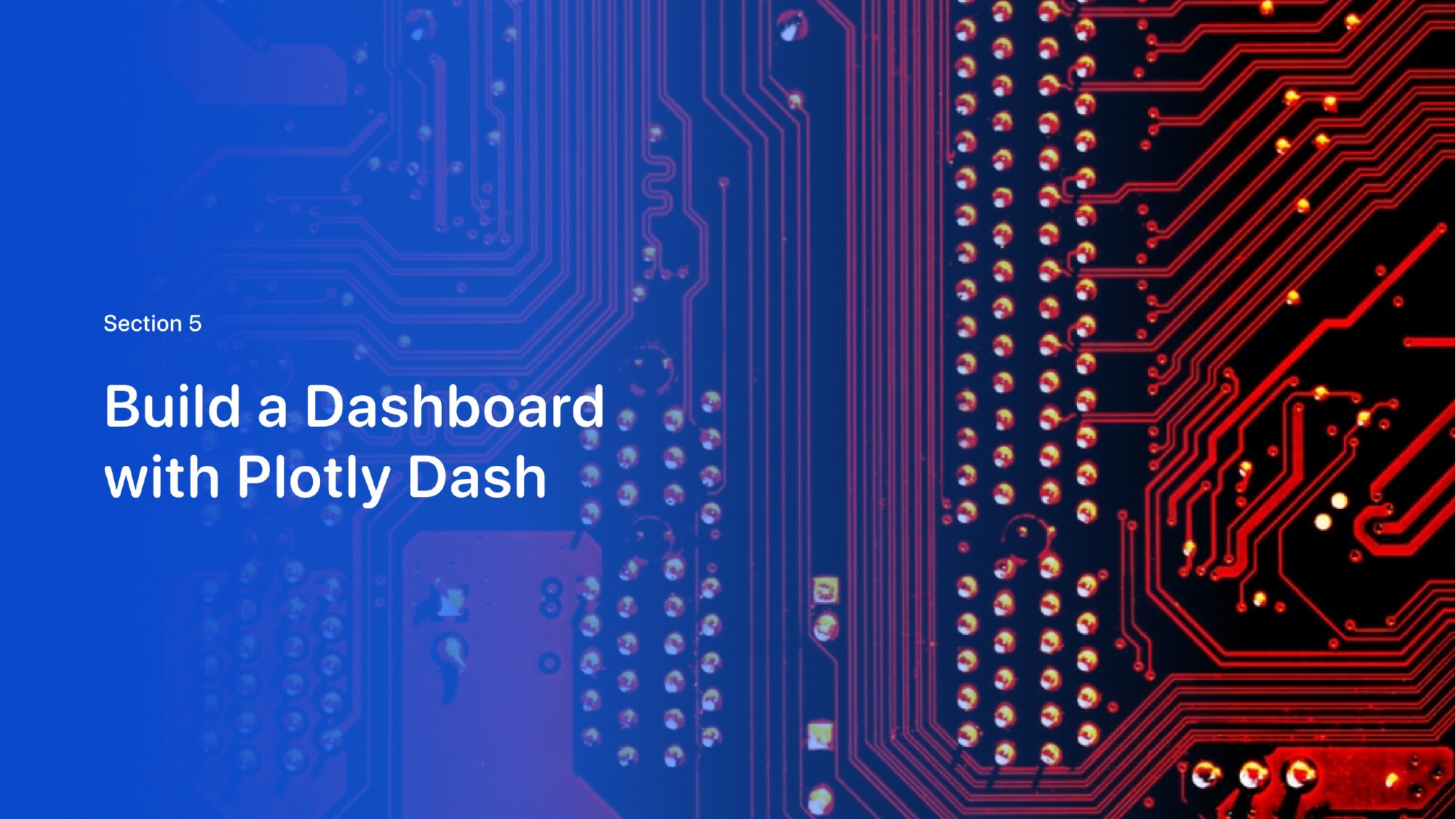
# Success/failed launches for each site on the map




color-

# Distances between a launch site to its proximities

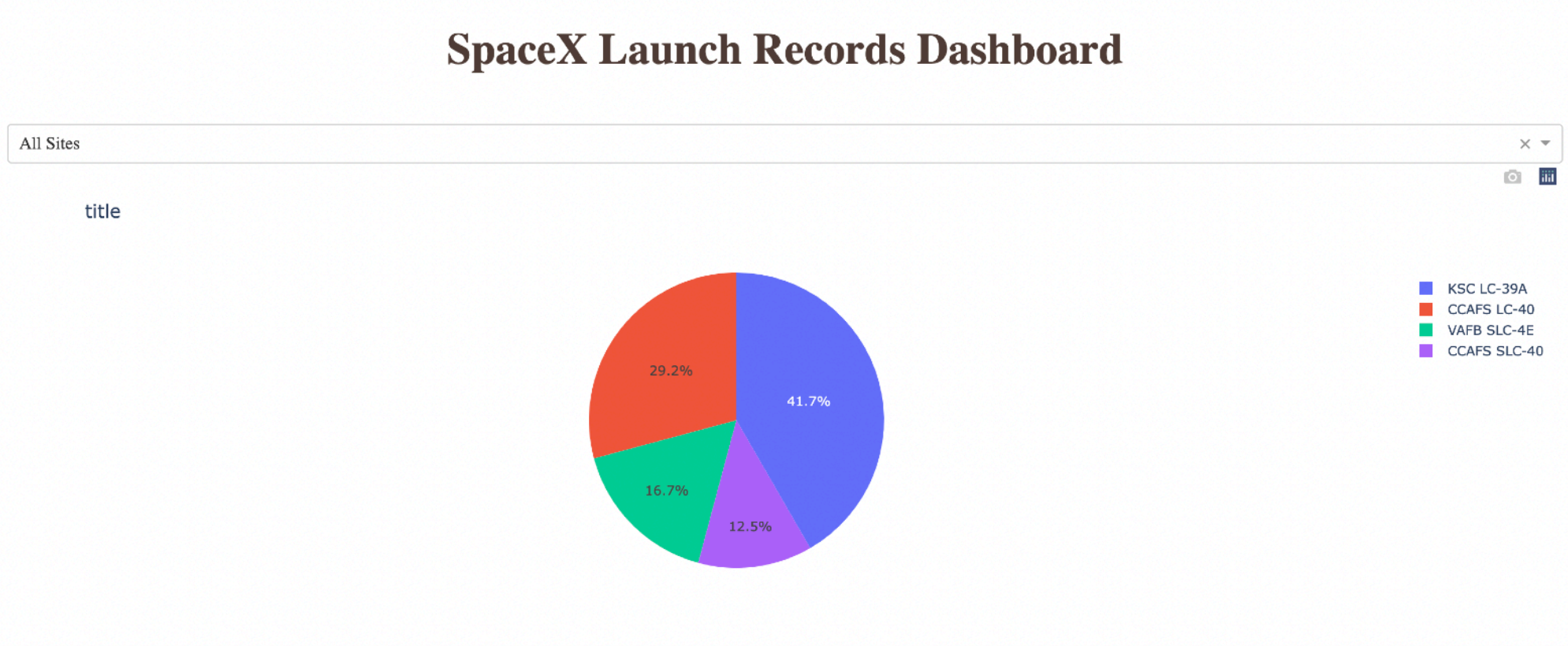




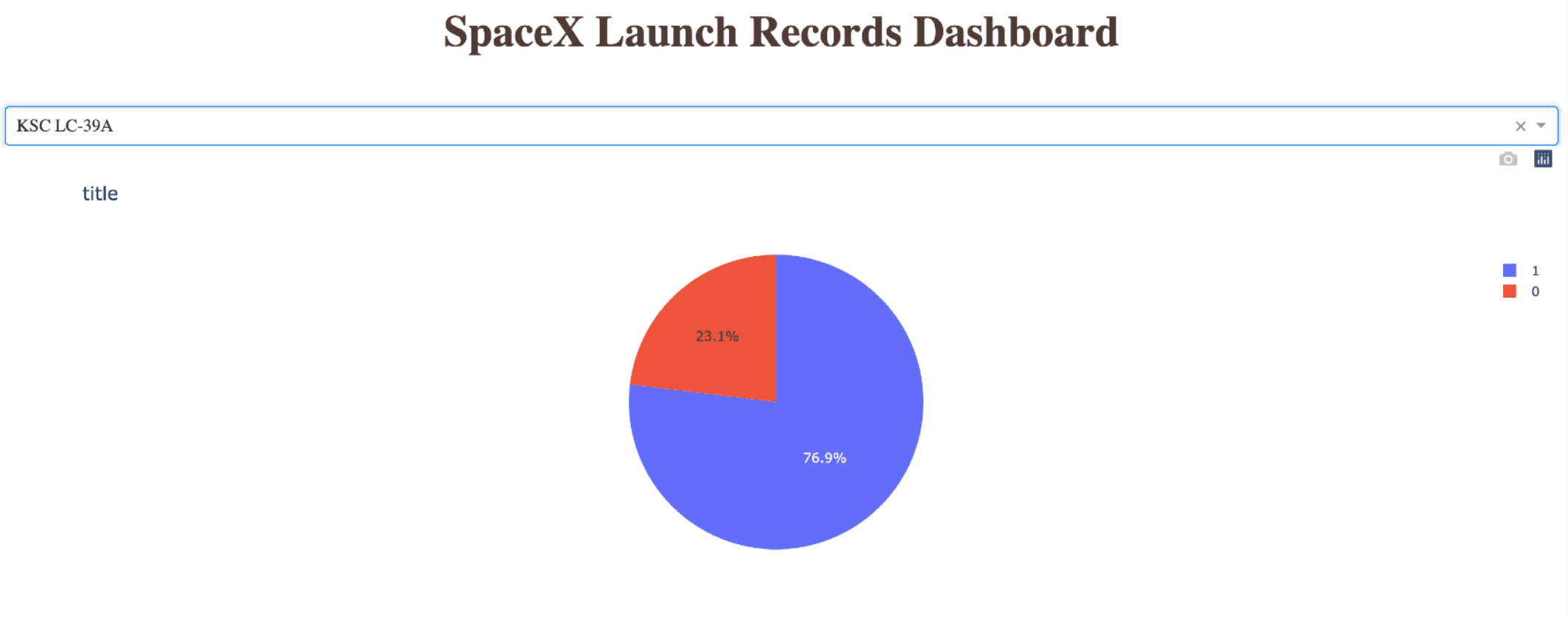
Section 5

# Build a Dashboard with Plotly Dash

# SpaceX Launch success count for all sites

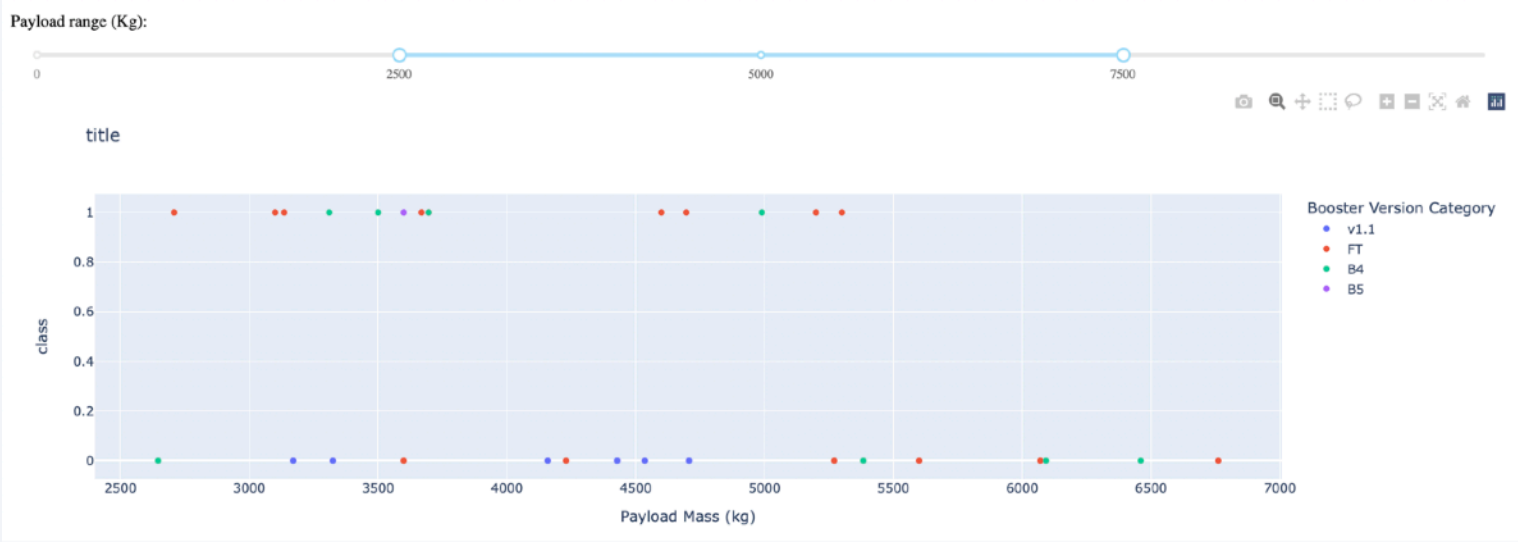


# Launch site with highest launch success ratio



# Payload vs. Launch Outcome scatter plot for all sites

- All sites, Payload from 2500 to 7500





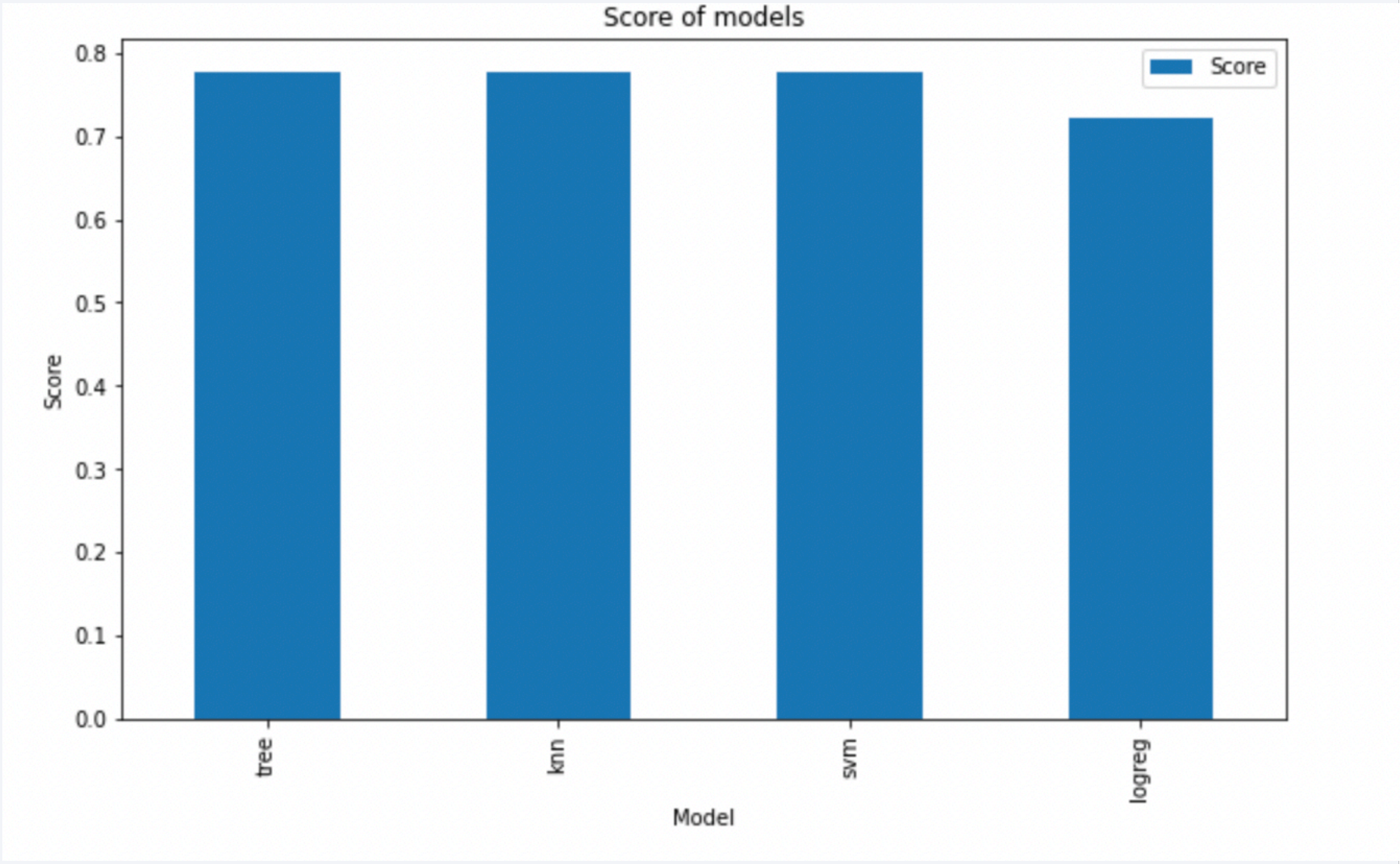
- All Sites all possible payloads

Section 6

# Predictive Analysis (Classification)

# Classification Accuracy

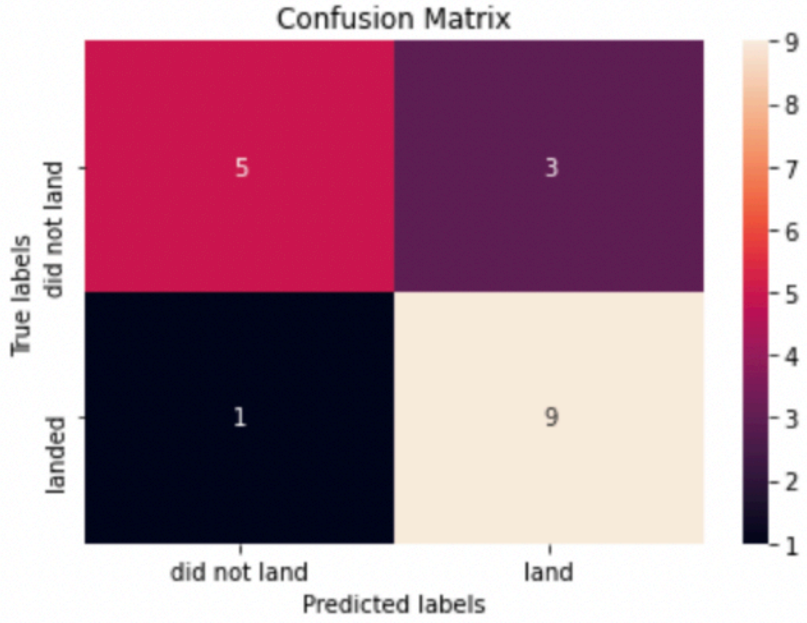
- Tree CV has the highest accuracy



# Confusion Matrix

- TreeCV has the the best performing model that predicted with the most accurate results.

```
In [24]: yhat = tree_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



# Conclusions

- The more rockets fly - the higher success rate
- Most of lunch sites have min 20 km from the nearest road but all of them are close to the water.
- The most popular launch site KSC LC-39A
- The biggest payload was 45 596, lowest was 2534
- ES-L1, GEO, HEO and SSO are having the absolute success rates.
- The heaviest payload traveled to VLEO orbit.
- ISS orbit mostly have payload weight between 2000 and 4000

# 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!