IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Luana Herllain da Silva Lopes

2021-09-30

Outline

Executive Summary

Introduction

Methodology

Results

Conclusion

Executive Summary

Summary of methodologies

Data Collection via API, SQL and Web Scraping

Data Wrangling

EDA with matplotlib and seaborn

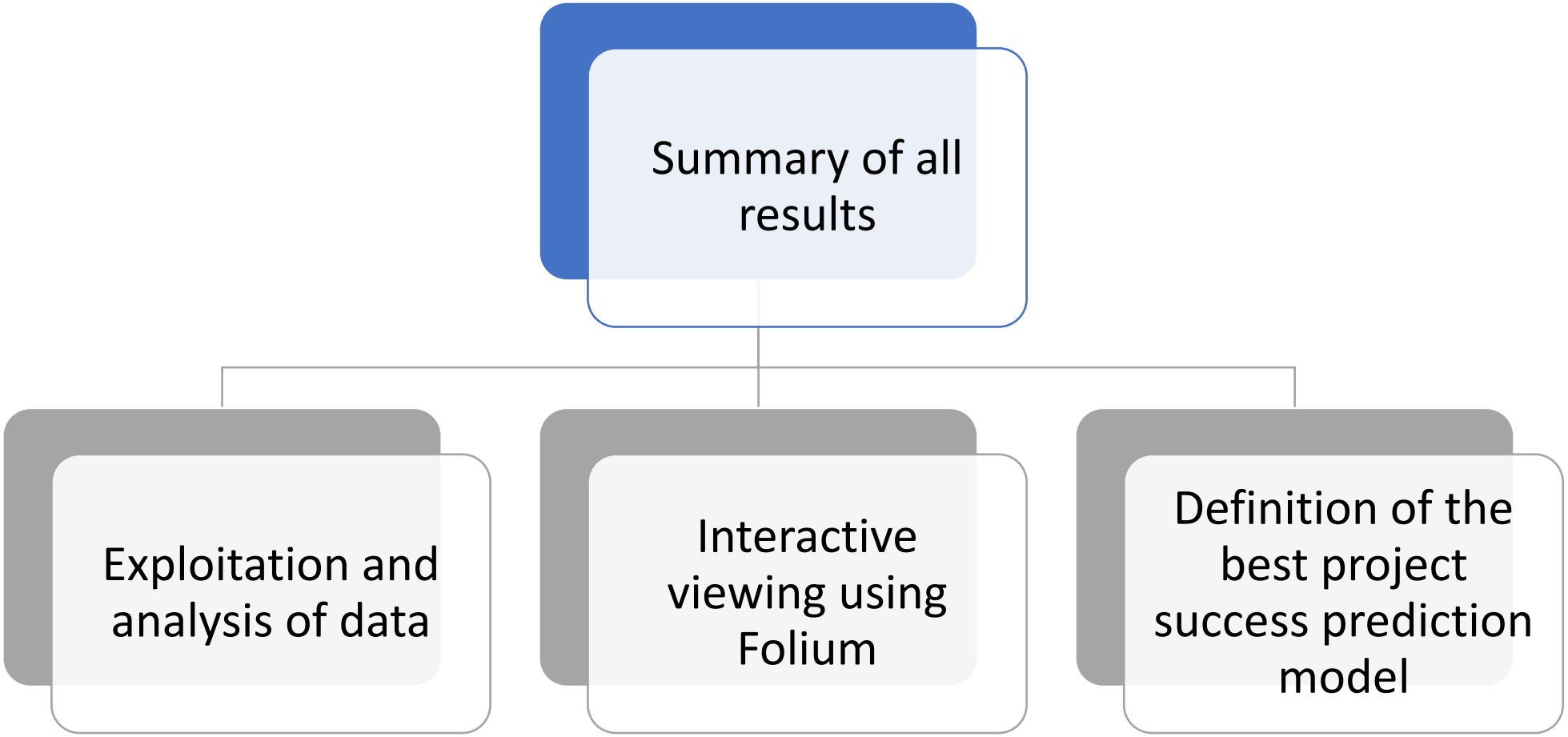
EDA with SQL

Building an interactive map with Folium

Building a dashboard with Plotly Dash

Predictive analysis – ML for classification

Executive Summary



Introduction

- **Project background and context:** Spacex has a cost associated with the launch of a rocket, as an example we can cite the Falcon 9 rocket costing 62 million dollars, others cost more than 165 million dollars each. The greatest potential for savings in launching each rocket is due to Spacex's reuse of the first stage of the project. Therefore, if it is possible to determine whether the first stage will land successfully or not, we can minimize and determine the cost of this project. This information is important in the case of a bidding process in which a rival company can present a lower cost and assume the project claimed by Specex.

- **Problems you want to find answers:**
  - How to predict whether or not a mission will be successful and consequently we will be able to take advantage of the first stage saving on future missions?
  - Which of the launch locations used has the highest percentage of success indicating that there may be some variable in it that has the greatest impact on the successful landing?

Section 1

Methodology

Methodology

Executive Summary

Data collection methodology:

- Sacex Rest API
- Web Scrapping from Wikipedia

Perform data wrangling

- One Hot Encoding data fields for Machine Learning and dropping irrelevant columns

Perform exploratory data analysis (EDA) using visualization and SQL

- Plotting : Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data

Perform interactive visual analytics using Folium and Plotly Dash

Perform predictive analysis using classification models

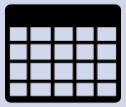
- How to build, tune, evaluate and classification models

Data Collection

- Data collection is a process of gathering relevant information for answering the questions of interest. Usually, it is the first step for data analysis. Data can be collected in many ways according to the location and method they're stored. For this project, we collect the data by REST APIs and web scraping from Wikipedia page.

1. Get data from API and Web page

2. Make data tabulated



3. Filter the data

4. Export to csv files

Data Collection – SpaceX API

Link GitHub

Get response from API

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

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

Convert response to .json files

```
# Use json_normalize meethod to convert the json result into a dataframe
df_response = response.json()

data = json_normalize(df_response)
```

Apply customized functions to clean data

```
# Call getBoosterVersion
getBoosterVersion(data)
```

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

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

Data Collection – SpaceX API

Create dataframe based on dict structure

```
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}
```

Filtering and export to flat files

```
# Create a data from launch_dict
df = pd.DataFrame.from_dict(launch_dict)
```

```
# Hint data['BoosterVersion']!='Falcon 1'
mask = (df['BoosterVersion']!='Falcon 1')
data_falcon9 = df.loc[mask]
```

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
```

```
# Calculate the mean value of PayloadMass column
mean_plm = data_falcon9['PayloadMass'].mean()
mean_plm
# Replace the np.nan values with its mean value
data_falcon9.replace(to_replace=np.nan, value=mean_plm, inplace=True)
```

```
data_falcon9
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad
4	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	6123.547647
5	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	6123.547647
6	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	6123.547647
7	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	6123.547647
8	5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	6123.547647
...	...	...	...	...	...	...	...	...	...	...	...	...

Link GitHub

Data Collection - Scraping

Get response from HTML → Create Beautiful Soup Object → Find tables → Get column names → Create dict-list structure to store data → Convert dict to dataframe → Export to flat files (.csv)

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

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

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`

html_tables = soup.find_all('table')
```

```
column_names = []

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

#col = first_launch_table.find_all('th')

for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

Check the extracted column names

```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant 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']=[]
```

```
df=pd.DataFrame(launch_dict)
```

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

df

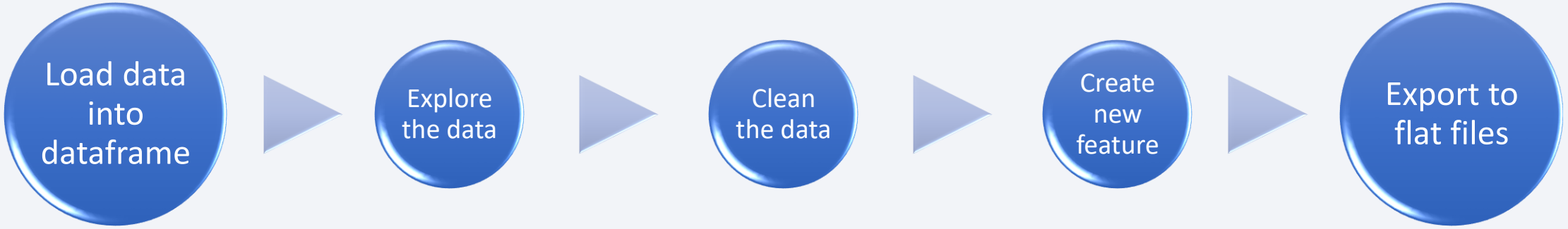
	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date
0	[4 June 2010,, 18:45]	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010
1	[8 December 2010,, 15:43]	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010
2	[22 May 2012,, 07:44]	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012
3	[8 October 2012,, 00:35]	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012
4	[1 March 2013,, 15:10]	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013
...	...	...	...	...	...	...	...	...	...	...

Link GitHub

Data Wrangling

Link GitHub

The ultimate goal of this project is to apply the dataset to a supervised model to decide if the booster can lan successfully or not. However, we can determine this process by many factors. To do this, let's explore and clean up some features(data exchange) to make them easier to input into the machine learning algorithm. Data organization is the process of cleaning up and unifying confusing and complex data sets for easy access and analysis.



Data Wrangling

Calculate the number of launches on each site:

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

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

VAFB SLC 4E - Cape Canaveral Space Launch Complex 40

(SLC-4E) - Vandenberg Air Force Base Space Launch Complex 4E

KSC LC 39ª - Kennedy Space Center Launch Complex 39A

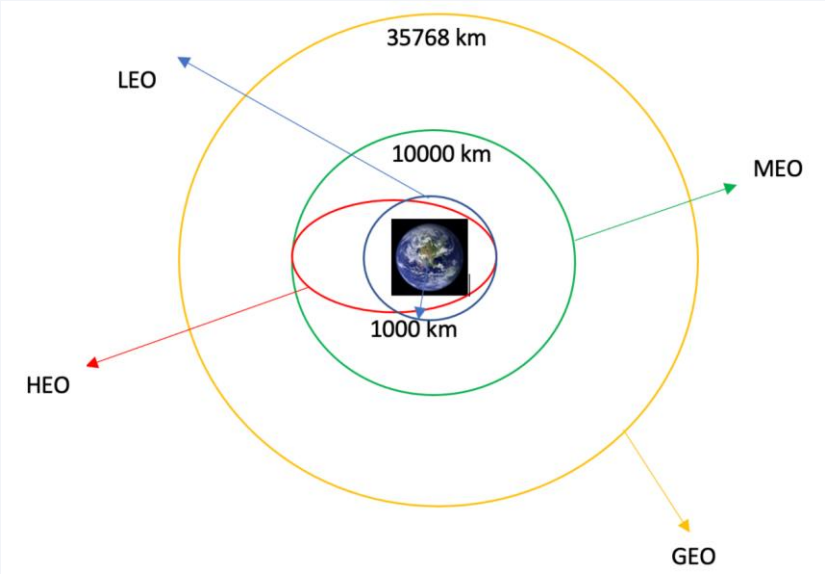
Features of interest:

- ✓ LaunchSite
- ➢ Orbit
- ➢ Outcome
- ➢ Derived Outcome label

Link GitHub

Data Wrangling

Calculate number and occurrence of each orbit:




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

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
HEO       1
GEO       1
ES-L1     1
SO        1
Name: Orbit, dtype: int64
```

Features of interest:

- ➢ LaunchSite
- ✓ Orbit
- ➢ Outcome
- ➢ Derived Outcome label

Link GitHub

Data Wrangling

Calculate number & occurences of mission outcome per orbit type:

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

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
None ASDS       2
False Ocean     2
False RTLS      1
Name: Outcome, dtype: int64
```

True Ocean - mission outcome was successfully landed to a specific region of the ocean.
False Ocean - mission outcome was unsuccessfully landed to a specific region of the ocean.
True RTLS - mission outcome was successfully landed to a ground pad.
False RTLS - mission outcome was unsuccessfully landed to a ground pad.
True ASDS - mission outcome was successfully landed to a drone ship.
False ASDS - mission outcome was unsuccessfully landed to a drone ship. None ASDS / None None - represent a failure to land.

Features of interest:

- ➢ LaunchSite
- ➢ Orbit
- ✓ Outcome
- ➢ Derived Outcome label

Link GitHub

Data Wrangling

Create a landing outcome label from Outcome column:

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class = []

for i in df.Outcome:
    if i.find('True') == -1:
        landing_class.append(0)
    else:
        landing_class.append(1)

print(landing_class)
```

```
[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

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

```
0.6666666666666666
```

```
df['Class']=landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

Features of interest:

- ➢ LaunchSite
- ➢ Orbit
- ➢ Outcome
- ✓ Derived Outcome label

Link GitHub

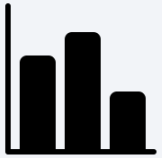
EDA with Data Visualization

Scatter Plots:

Scatter plots can visualize dependencies of attributes with each other. This is usually called correlation. Once a pattern is determined, it will help us to predict which factor would lead to the success of landing outcome.

- Flight Number and Payload
- Flight Number and Launch Site
- Payload and Launch Site
- Flight Number and Orbit type
- Payload and Orbit type

Bar Graphs:



Bar graph is a great way to visualize categorical data type. Each bar represents a member under that categorical feature, and it is easy to compare which member has the largest amount and which the least.

- Success rate of each orbit type

Line Plot:

Line Plot is similar to scatter plot while the former is often used to visualize continuous numeric data type. From line plot, we can easily tell the trend between two variables thus helping us to make prediction about the future.

- Launch success yearly trend

Link GitHub

EDA with SQL

SQL é uma ferramenta indispensável para o cientista e analista de dados, pois a maioria dos dados do mundo real sãoarmazenados em bancos de dados. SQL com seu nome completo de Structured Query Language, pode facilmente nos ajudar a visitaros dados, bem como analisar e extrair insights úteis deles. Neste projeto, usamos IBM Db2,que é um banco de dados SQL totalmente gerenciado fornecido como um serviço.

The following questions are explored:

- 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 successful landing outcome in ground pad was achieved.
- 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. Use a subquery.
- List the failed landing outcomes in drone ship, their booster versions, and launch site names for 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.

Link GitHub

Build an Interactive Map with Folium

To visualize the Launch Data into an interactive map. We took the Latitude and Longitude Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.

We assigned the dataframe launch_outcomes(failures, successes) to classes 0 and 1 with Green and Red markers on the map in a MarkerCluster().

Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. Lines are drawn on the map to measure distance to landmarks.

- In this project, we:
  - Place each launch site on the map as a marker.
  - Render successful land outcome as GREEN color and unsuccessful one as RED color.
  - Enable launch sites with the same coordinates to fold/unfold based on zoom level.
  - Calculate and visualize the distance between launch sites and some certain kinds of their proximities.

Link GitHub

Build a Dashboard with Plotly Dash

Pie Chart – shows the successful landing rate for each launch site or overall sites, according to the dropdown menu selected.

Scatter Plot – shows the correlation between two variables. The following has been explored:

- Flight Number and Payload
- Flight Number and Orbit type
- Payload and Orbit type
- Payload and Launch Site
- Flight Number and Launch Site

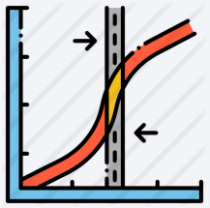
Link GitHub

Objects	Codes	Result
Dash and its components	import dash import dash_html_components as html import dash_core_components as dcc from dash.dependencies import Input, Output	Plotly Dash is simple enough that you can bind a user interface to Python, R, Julia, or F# code in less than 10 minutes. With Dash Open Source, Dash apps can run on a local laptop or server. The Dash Core Component library contains a set of higher-level components such as sliders, graphs, dropdowns, tables and more. Dash provides all of the available HTML tags as user-friendly Python classes
Plotly	import plotly.express as px	Plots graphs with interactive plotly library.
Drpdown	dcc.Dropdownd()	Create a dropdown menu for launch sites.
RangeSlider	dcc.RangSlider()	Create a range slider for PayLoadMass range selection.
Pie Chart	px.pie()	Create pie charts for successful landing rate of each launch site.
Sactter Plot	Px.scatter()	Create scatter plot for correlation among features.

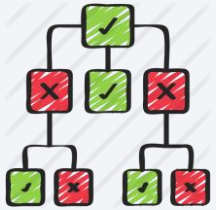
Predictive Analysis (Classification)

Classification Model Candidates:

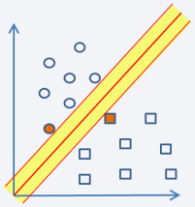
Link GitHub



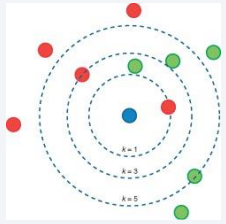
Logistic Regression



Decision Tree



SVM



KNN

Build

- Load data into NumPy & Pandas
- Ransform and standardize data
- Split data into training and testing ste
- List down suitable machine learning models
- Wrap hyperparameters into GridSearchCV object
- Fit training data to macine learning models

Evaluate

- Check accuracy for each model
- Get tuned hyperparameters for each type of algorithm
- Plot confusion metrics

Improve

- Feature engeneering
- Algorithm tunning

Select

- The model with the beste accuracy score on testing data wiins the best perfoming classification model

Results

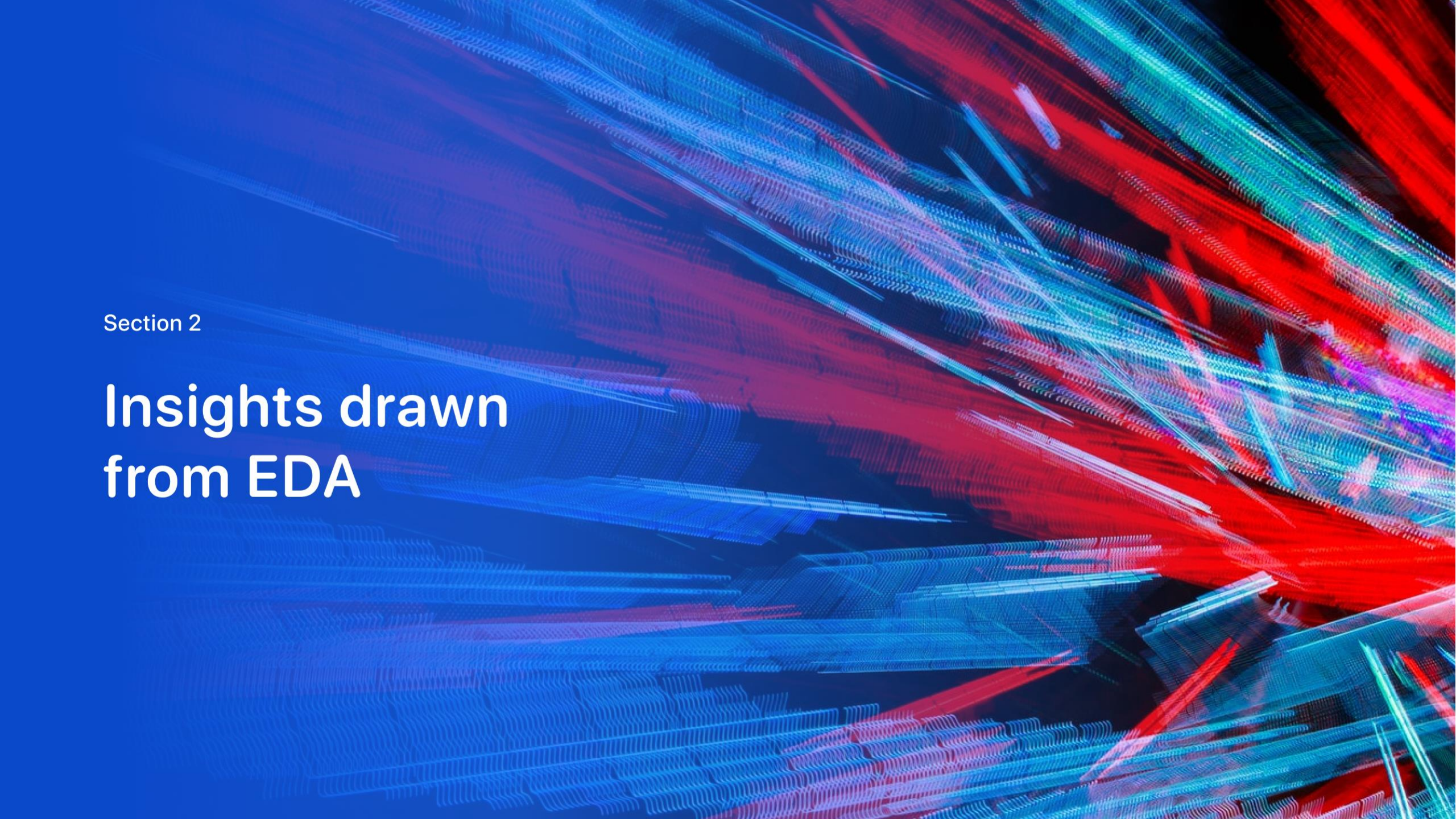
Exploratory data analysis results

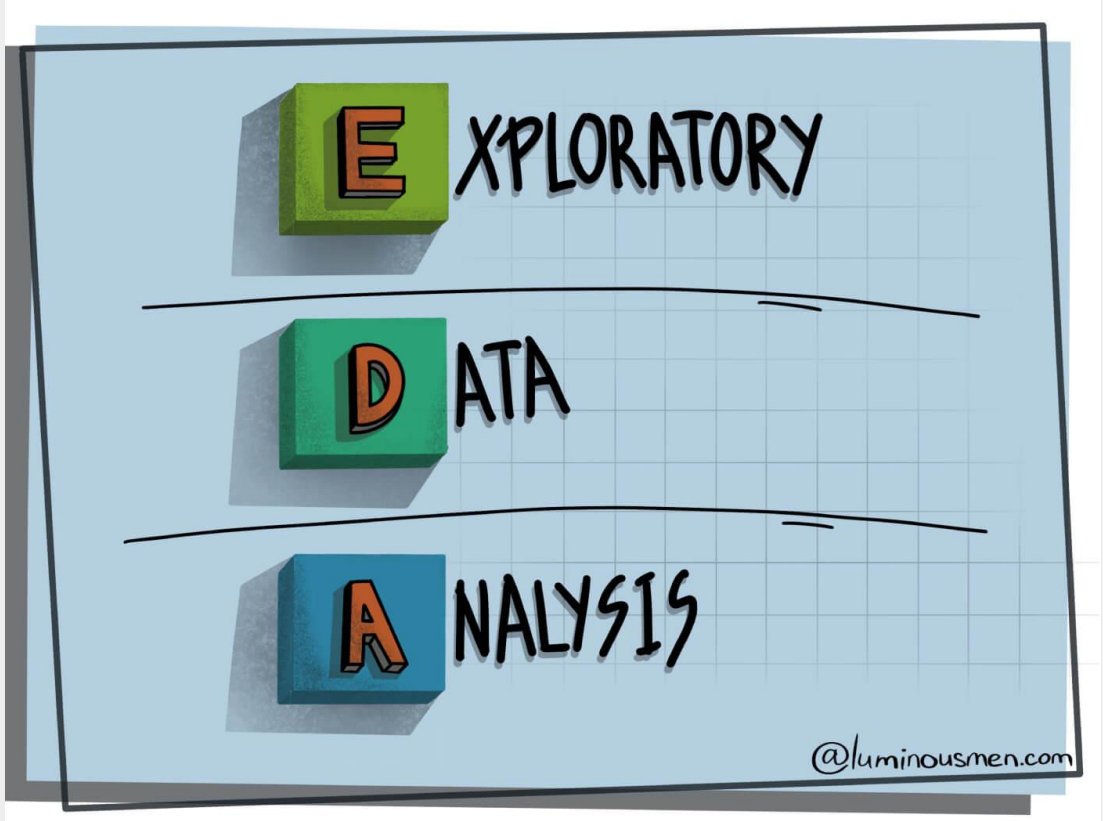
Interactive analytics demo in screenshots

Predictive analysis results

Section 2

Insights drawn from EDA



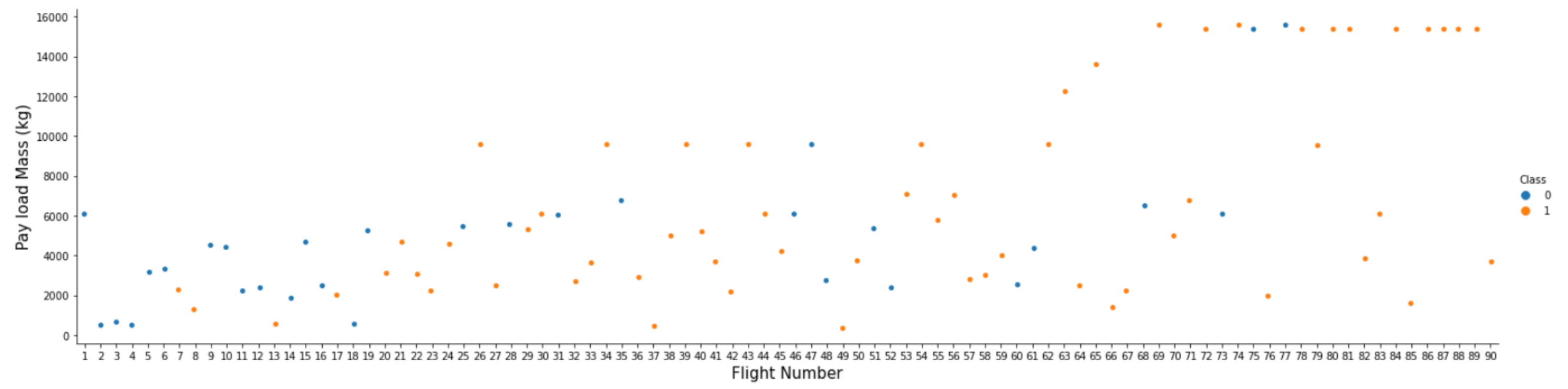


EDA WITH VISUALIZATION

Flight Number vs. Launch Site

Link GitHub

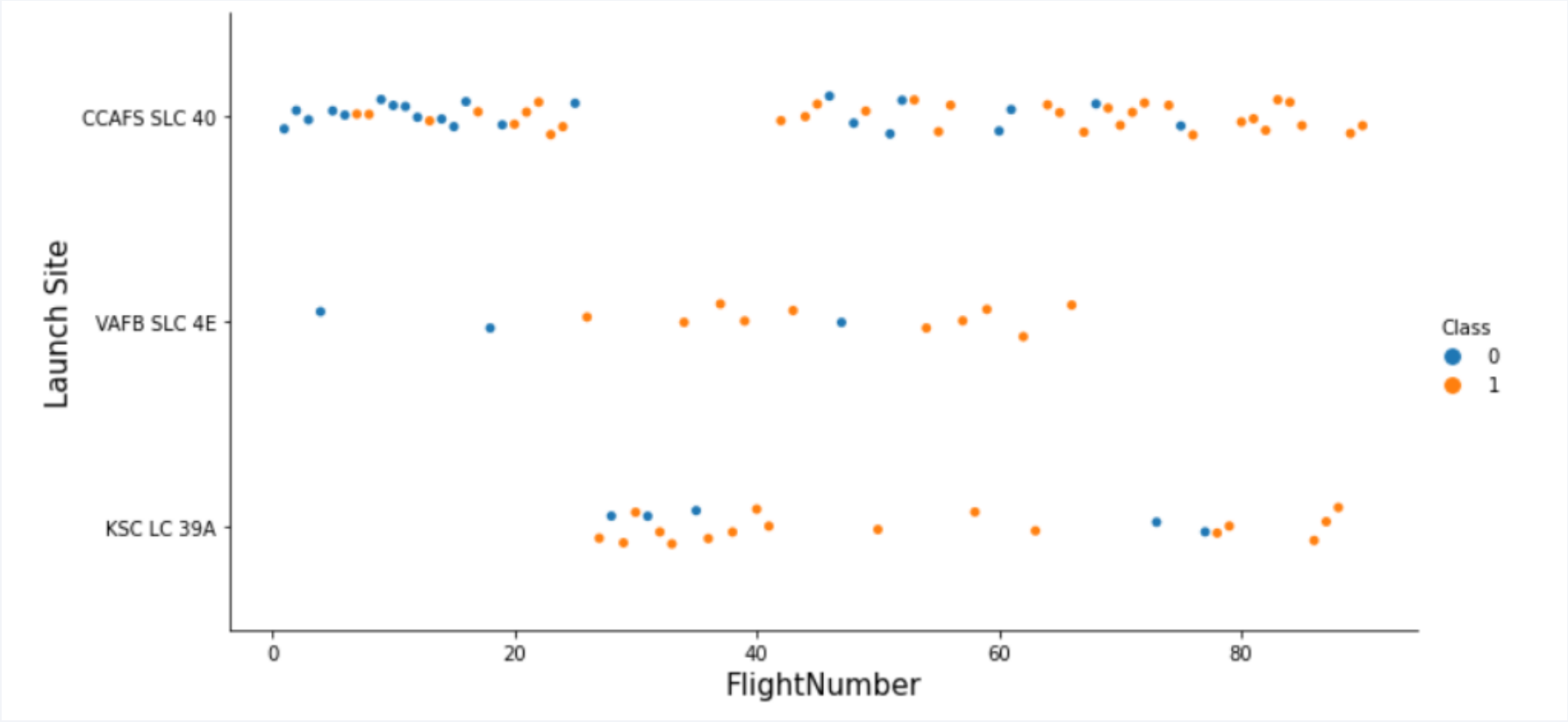
The more flights at a launch site, the higher your launch success rate will be.



Payload vs. Launch Site

Link GitHub

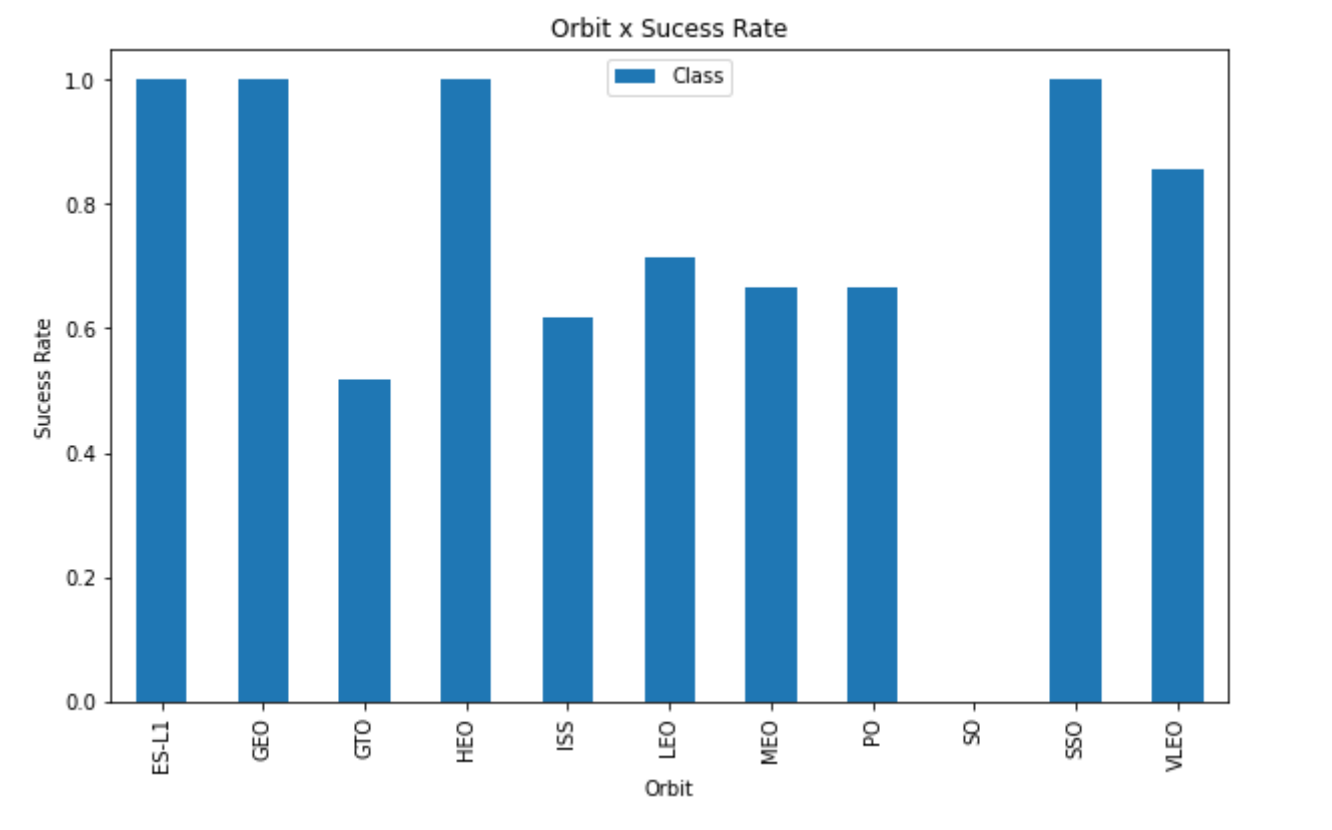
The smaller the payload mass for the CCAFS SLC 40 launch site, the greater the success rate of the rocket. However, there is no clear pattern to be found using this view to make a decision whether the launch location is dependent on Pay Load Mass for a successful launch.



Success Rate vs. Orbit Type

Link GitHub

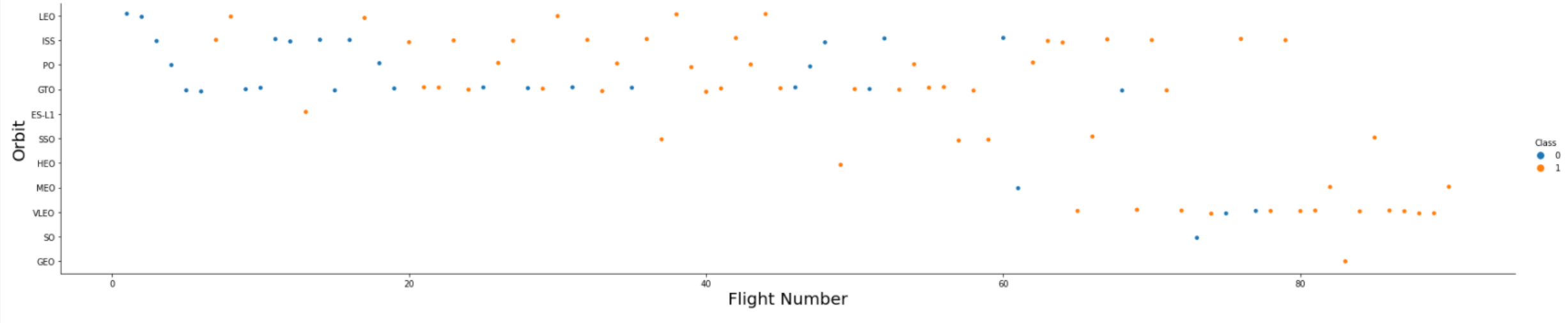
It can be seen that GEO, HEO, SSO, ES-L1 orbits have the best success rates.



Flight Number vs. Orbit Type

Link GitHub

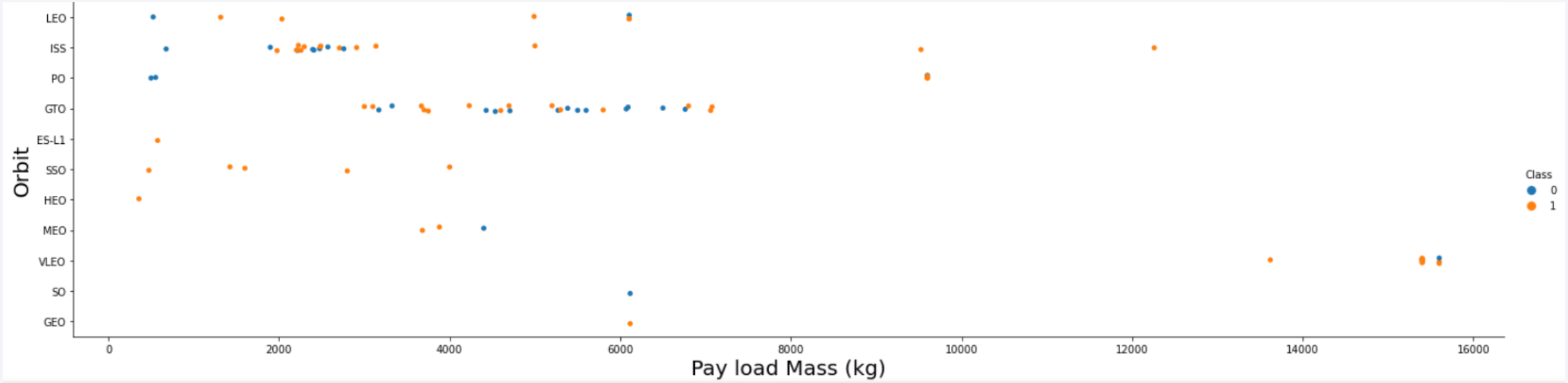
The success of the LEO orbit appears to be related to the number of flights. On the other hand, there seems to be no relationship between the flight number when in orbits such as GTO, ISS, PO, etc.



Payload vs. Orbit Type

Link GitHub

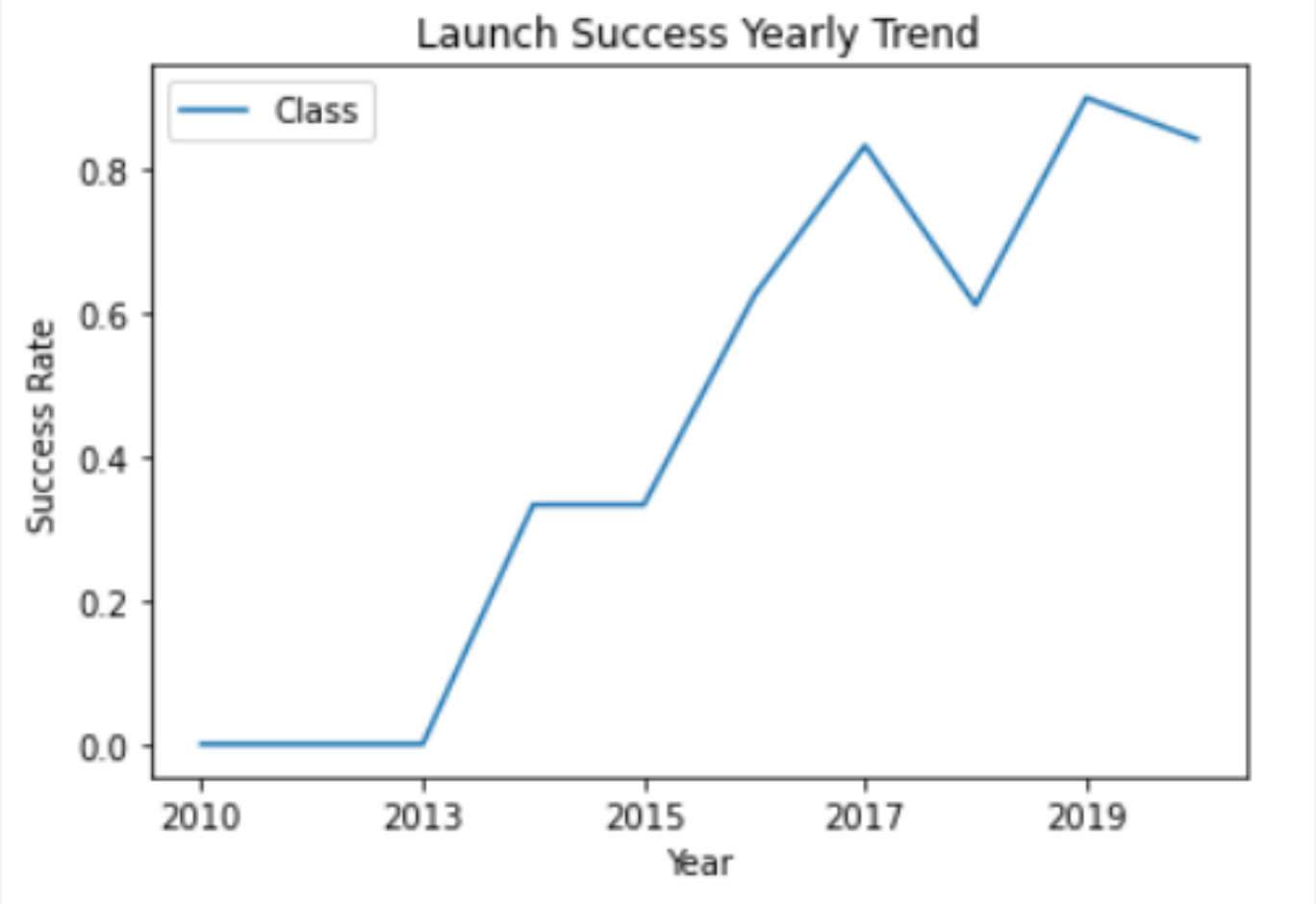
You should note that payloads have a negative influence on GTO orbits and a positive influence on GTO and Polar LEO (ISS) orbits.

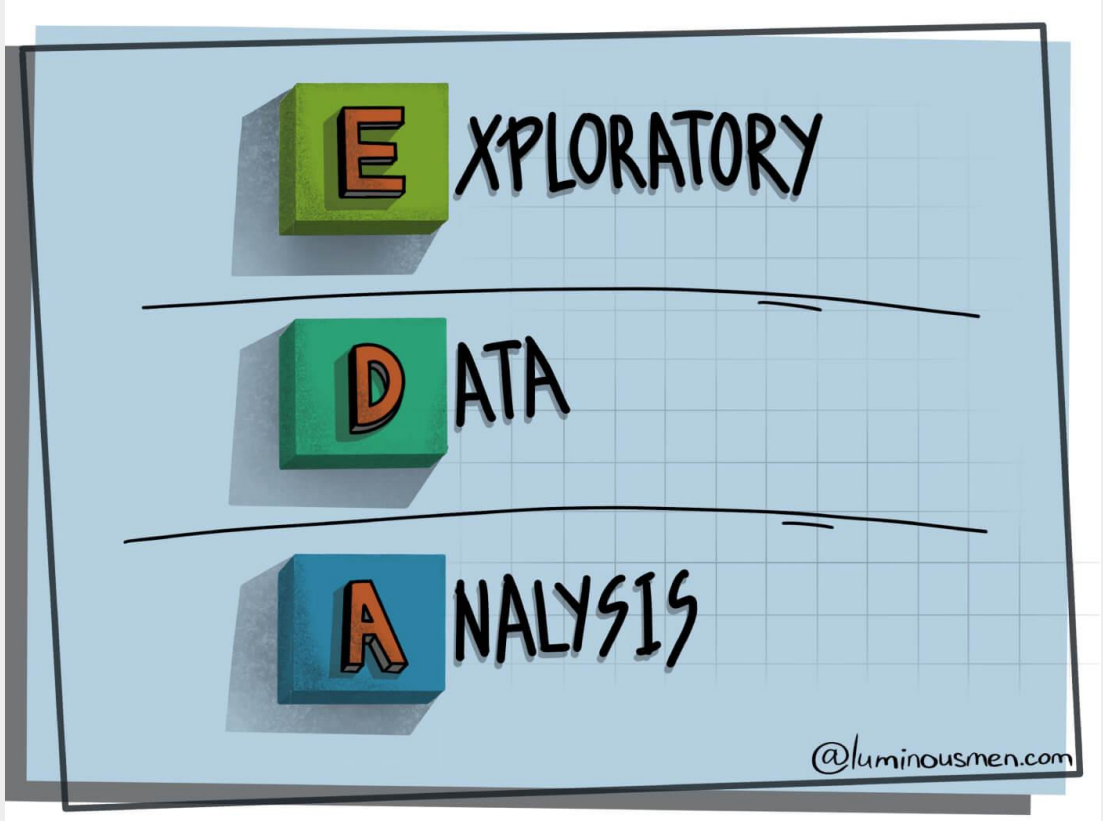


Launch Success Yearly Trend

Link GitHub

It can be seen that the success rate since 2013 has continued to increase, with only a considerable drop in 2018.





EDA WITH SQL

All Launch Site Names

Link GitHub

Task 1

Display the names of the unique launch sites in the space mission

```
%sql SELECT UNIQUE LAUNCH_SITE FROM SPACEXDATASET
```

```
 * ibm_db_sa://xbs38332:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

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

Launch Site Names Begin with 'CCA'

Link GitHub

Task 2

Display 5 records where launch sites begin with the string 'CCA'

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

```
 * ibm_db_sa://xbs38332:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

Link GitHub

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS) ¶

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXDATASET WHERE CUSTOMER LIKE '%NASA (CRS)%'
```

```
 * ibm_db_sa://xbs38332:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

1
48213

Average Payload Mass by F9 v1.1

Link GitHub

Task 4

Display average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXDATASET WHERE BOOSTER_VERSION LIKE '%F9 v1.1%'
```

```
 * ibm_db_sa://xbs38332:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

1
2534

First Successful Ground Landing Date

Link GitHub

Task 5

List the date when the first successful landing outcome in ground pad was acheived.

Hint:Use min function

```
%sql SELECT MIN("DATE") FROM SPACEXDATASET WHERE LANDING__OUTCOME LIKE '%Success%'
```

```
 * ibm_db_sa://xbs38332:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Link GitHub

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
%sql SELECT BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM SPACEXDATASET WHERE LANDING__OUTCOME LIKE 'Success (drone ship)' AND PAYLOAD
_MASS__KG_ < 6000 AND PAYLOAD_MASS__KG_ > 4000
```

```
 * ibm_db_sa://xbs38332:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

booster_version	payload_mass__kg_
F9 FT B1022	4696
F9 FT B1026	4600
F9 FT B1021.2	5300
F9 FT B1031.2	5200

Total Number of Successful and Failure Mission Outcomes

Link GitHub

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_MISSION_TYPES FROM SPACEXDATASET GROUP BY MISSION_OUTCOME
```

```
 * ibm_db_sa://xbs38332:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

mission_outcome	total_mission_types
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

Link GitHub

Task 8

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

```
%sql SELECT BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM SPACEXDATASET WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXDATASET) ORDER BY PAYLOAD_MASS__KG_ DESC
```

```
 * ibm_db_sa://xbs38332:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

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

Link GitHub

Task 9

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

```
%sql SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE, "DATE" FROM SPACEXDATASET WHERE "DATE" LIKE '%2015%' AND LANDING__OU
TCOME LIKE 'Failure (drone ship)'
```

```
 * ibm_db_sa://xbs38332:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

landing__outcome	booster_version	launch_site	DATE
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015-01-10
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015-04-14

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

Link GitHub

Task 10

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

```
%%sql SELECT
    LANDING__OUTCOME,
    COUNT(LANDING__OUTCOME) AS TOTAL_LANDING__OUTCOME
FROM
    SPACEXDATASET
WHERE
    LANDING__OUTCOME LIKE 'Failure (drone ship)' OR LANDING__OUTCOME LIKE 'Success (ground pad)'
    AND "DATE" > '2010-06-04'
    AND "DATE" < '2017-03-20'
GROUP BY
    LANDING__OUTCOME
ORDER BY
    TOTAL_LANDING__OUTCOME DESC
```

```
 * ibm_db_sa://xbs38332:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

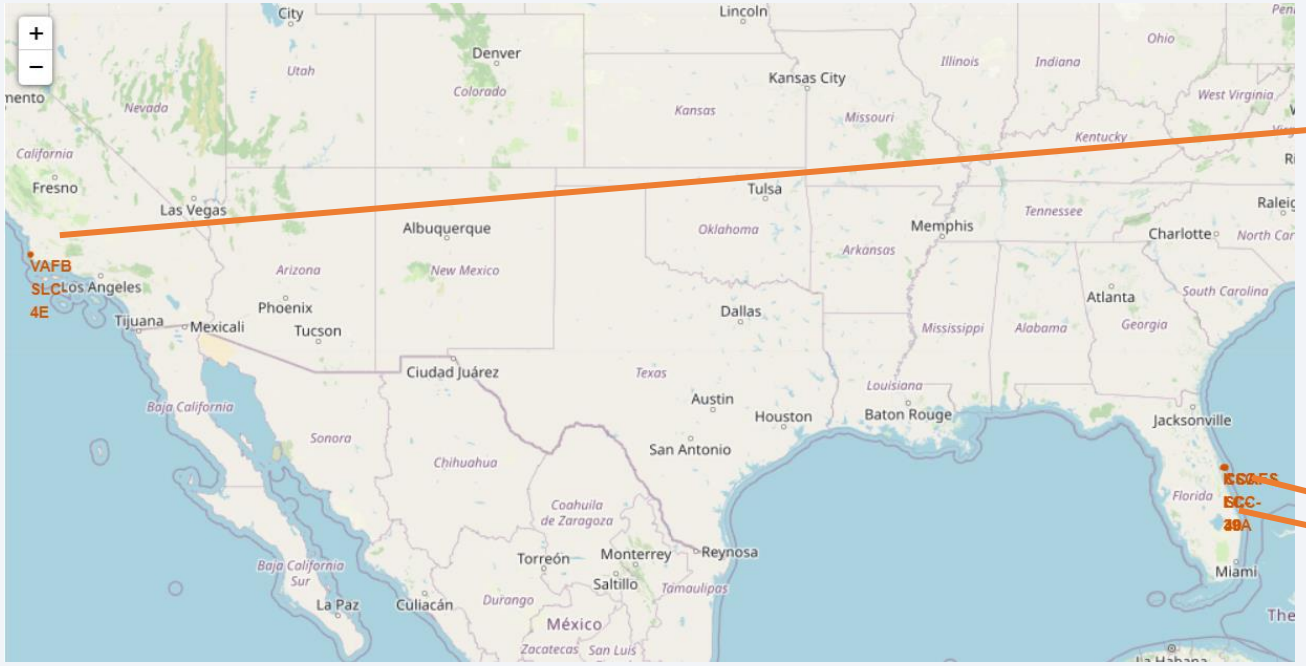
landing__outcome	total_landing__outcome
Failure (drone ship)	5
Success (ground pad)	3

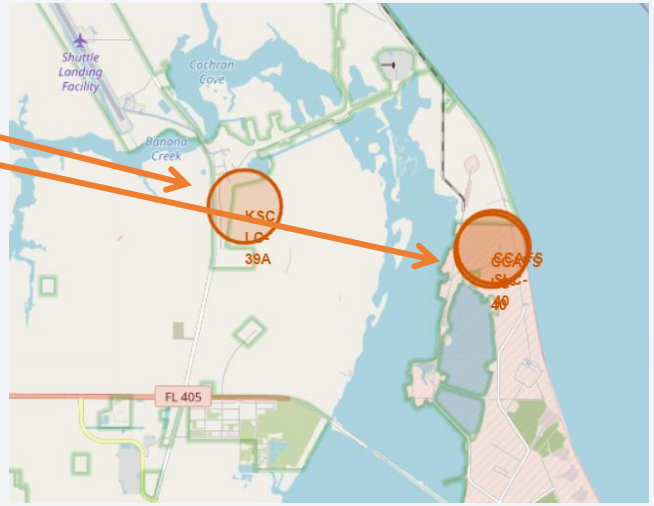
Section 4

Launch Sites Proximities Analysis

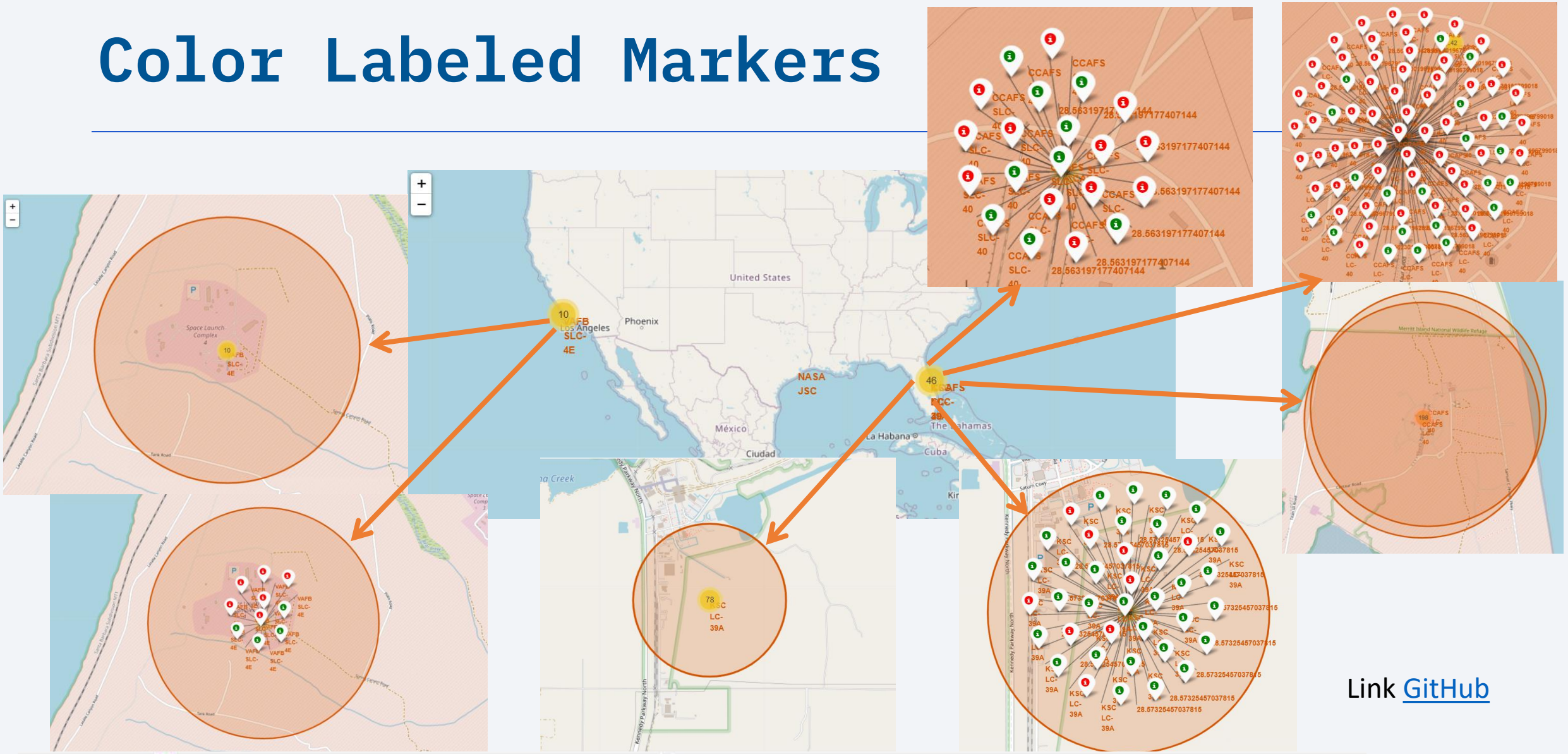
All Launch Sites on the Map

Link GitHub





Color Labeled Markers



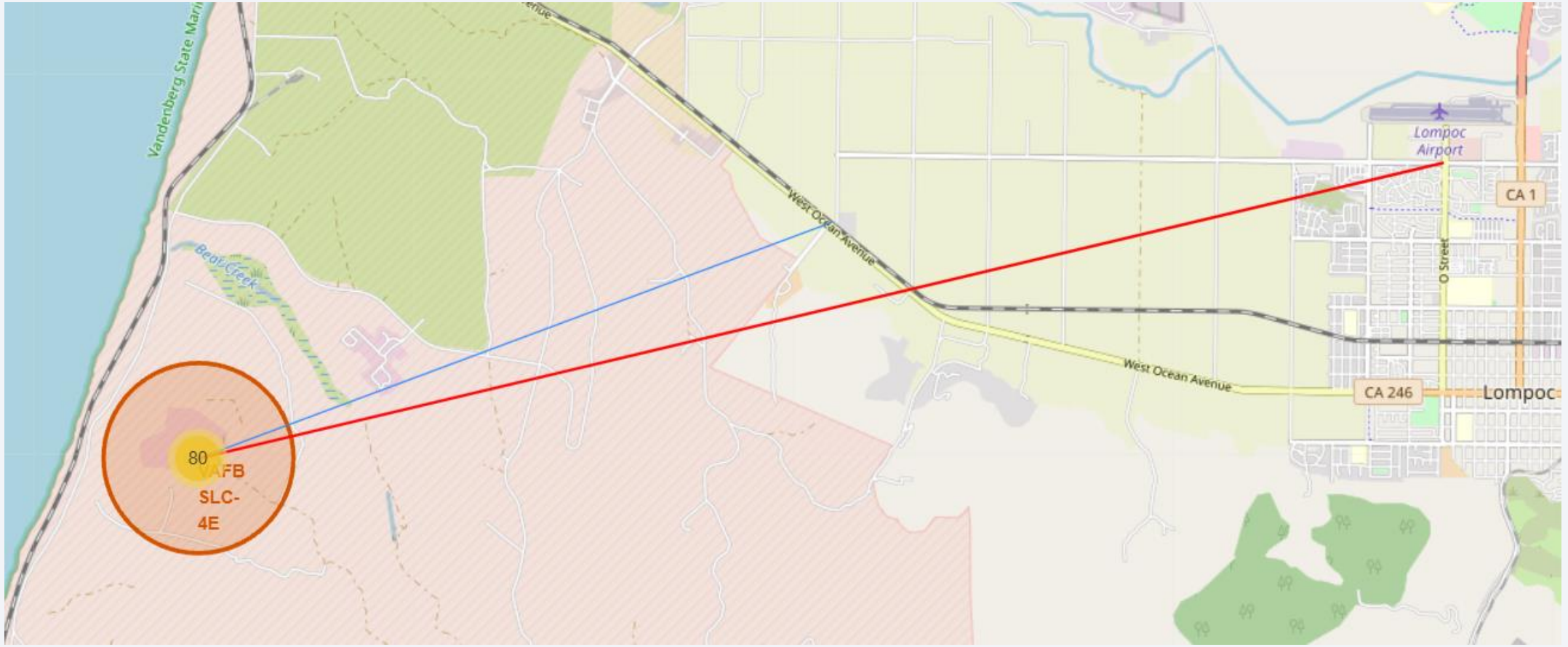
Link [GitHub]

- Green label represent successful landing outcome
- Red label represents unsuccessful landing outcome;

- KSC LC-39A has the **largest** successful landing ratio;
- CCAFS SLC-40 has the **largest** unsuccessful landing ratio;

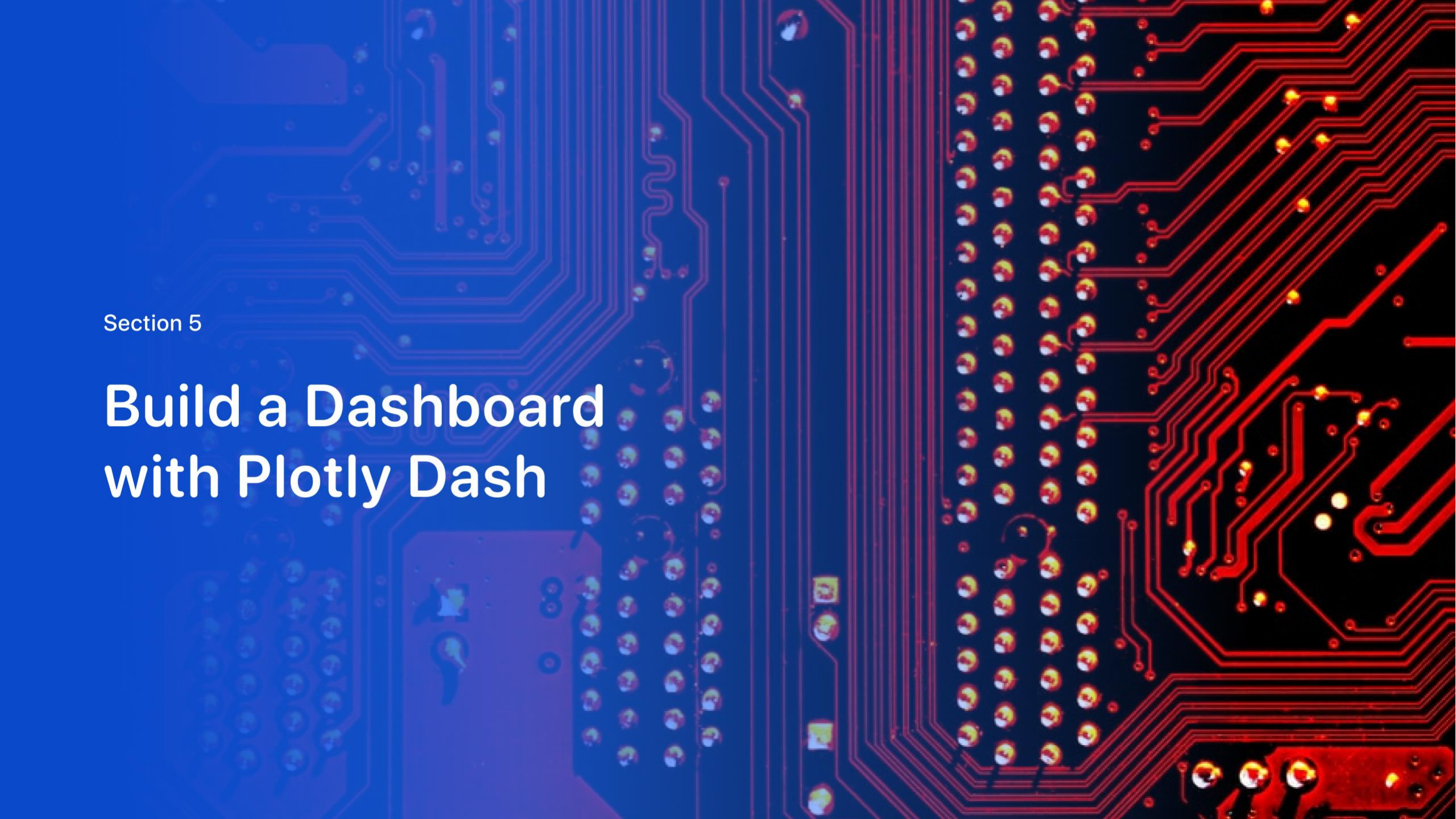
Launch Site Distance Airport & Railways

Link GitHub



Section 5

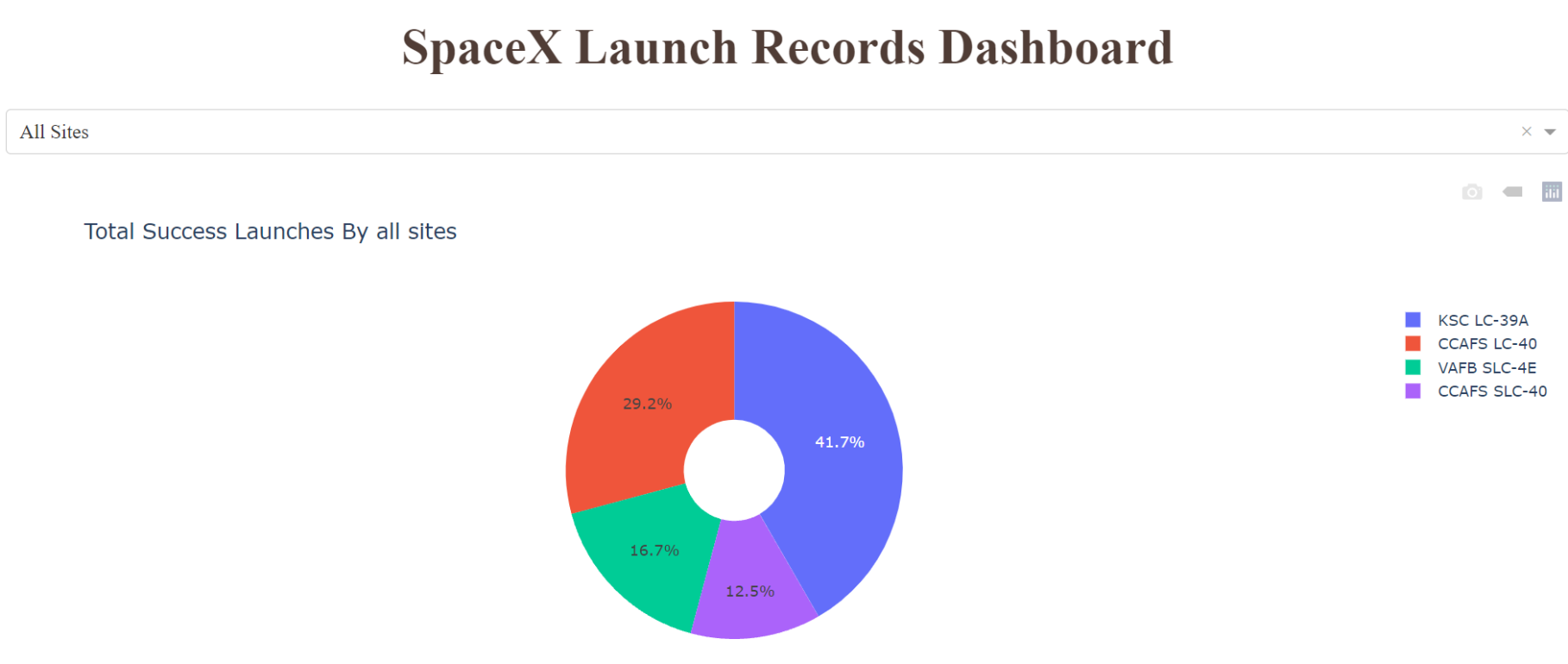
Build a Dashboard with Plotly Dash



Launch Success Ratio for All Successful Launch Records

Link GitHub

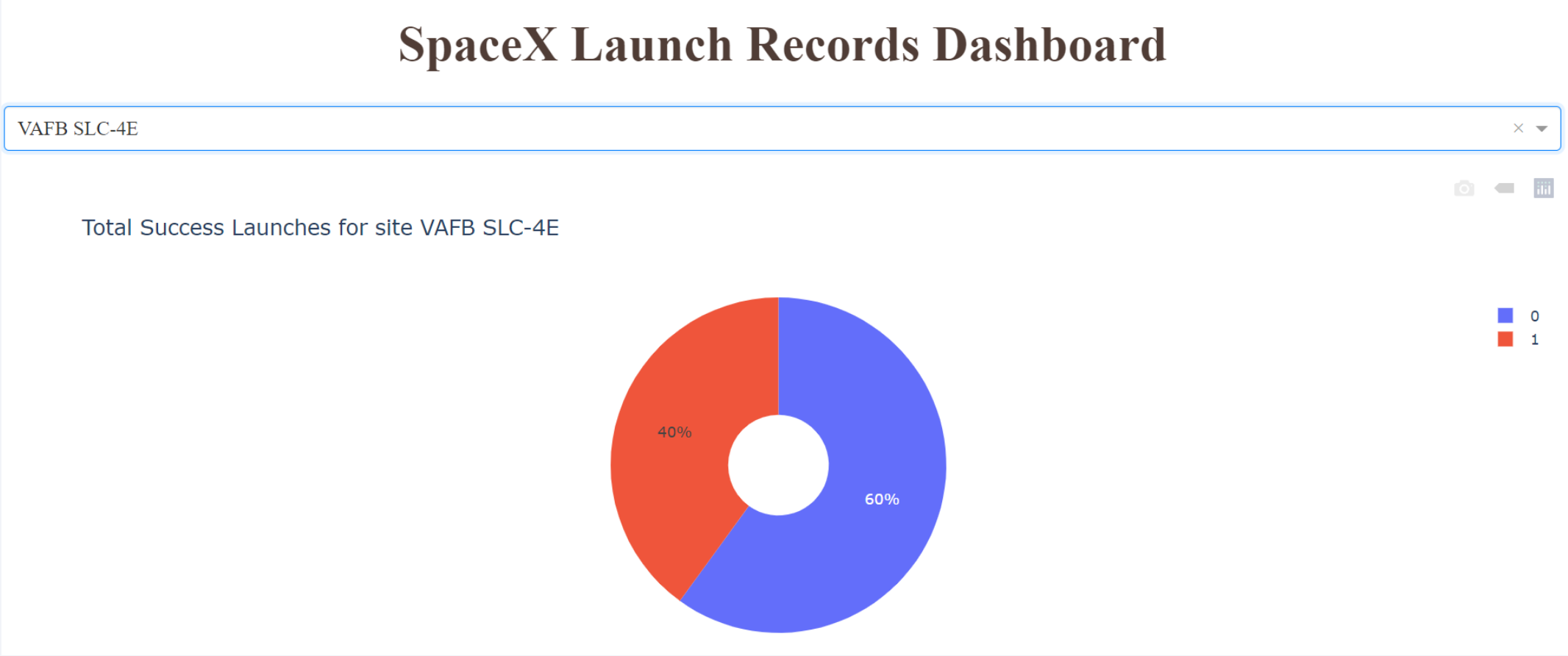
We can see that KSC LC-39A (41.7%) has the MOST successful launches of all successful launches.



Success Rate for the Launch Site with Most Success Launch Count Percentag

Link GitHub

You can see from the pie chart below that VAFB SLC-4E has ONLY 40% success rate.



Effect of Payload Mass on Successful Launches

Link GitHub

We can see that the successful launches' count is likely larger when payload mass is below 4,000 kg

Section 6

Predictive Analysis (Classification)

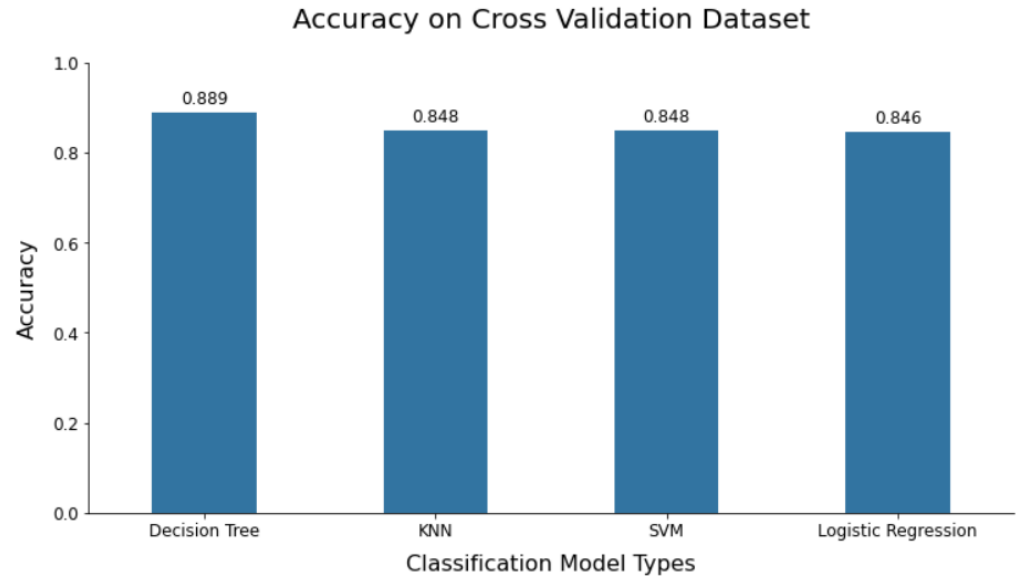
Classification Accuracy

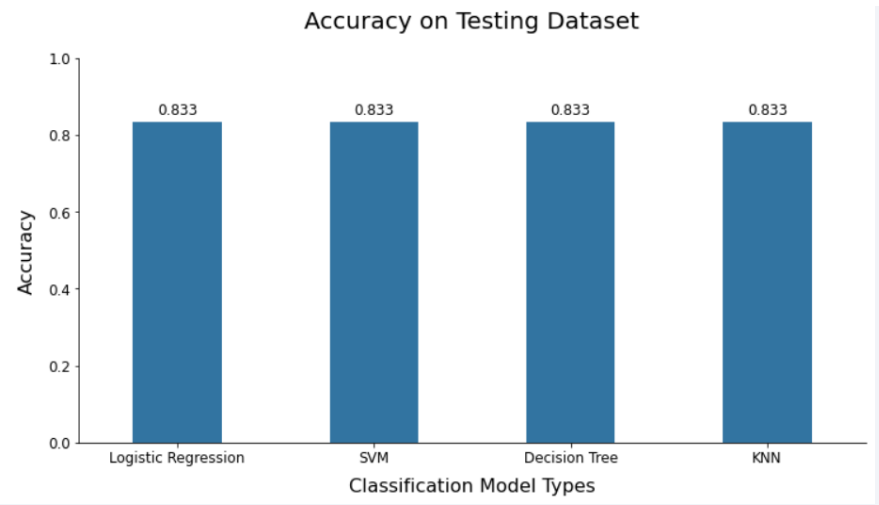
Link GitHub

As we can see, among the four classification models, the Decision Tree stands out in cross-validation scores (0.889) and the test score remained the same as the others (0.833). Therefore, we can use this trained model to predict the outcome of landing SpaceX rockets in the future.

	CV Score	Test Score
Decision Tree	0.889286	0.833333
KNN	0.848214	0.833333
SVM	0.848214	0.833333
Logistic Regression	0.846429	0.833333

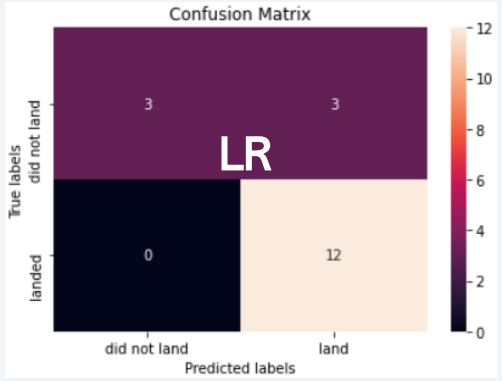
```
Decision Tree cv score: 0.889
           test score: 0.833

Model Hyperparameters:
         criterion: gini
         max_depth: 8
      max_features: sqrt
  min_samples_leaf: 1
 min_samples_split: 2
          splitter: random
```

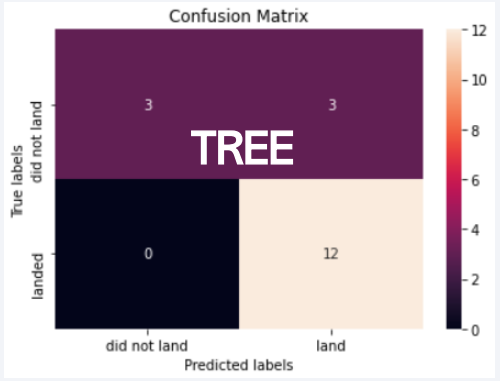


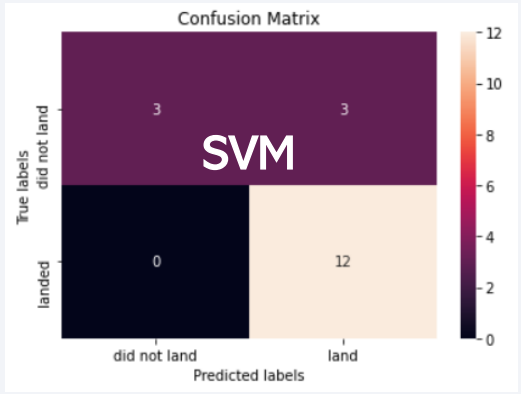


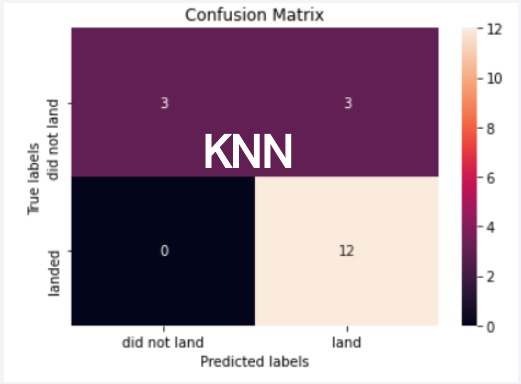
Confusion Matrix

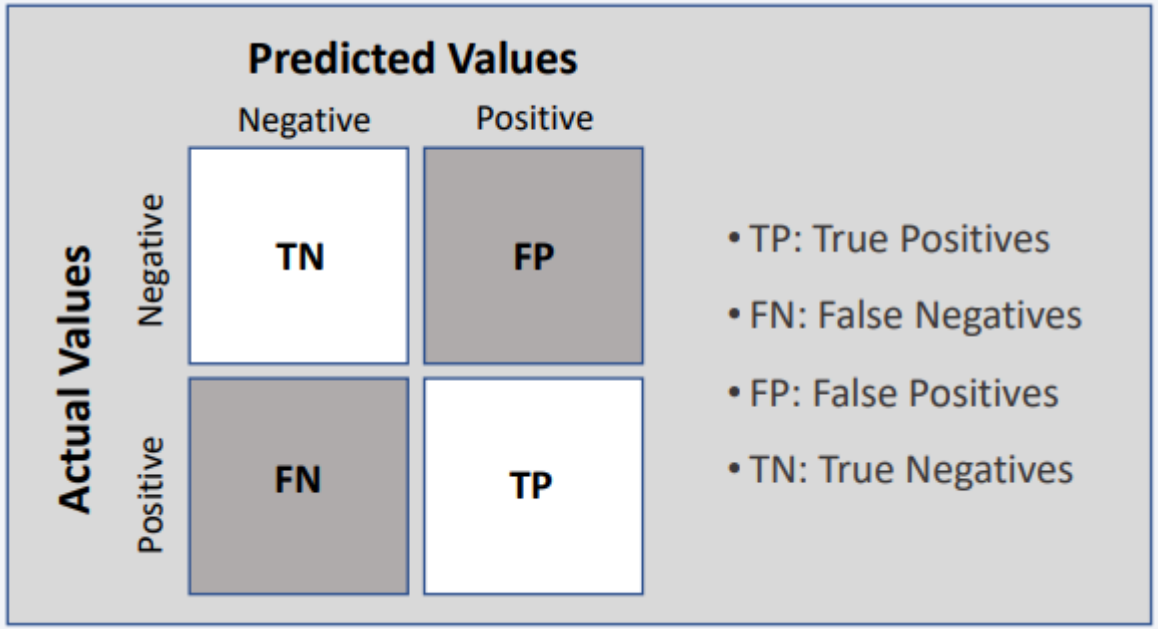
Link GitHub











Conclusions

- All launch sites are far away from their neighboring cities; but in close proximity to railways and coastlines; relatively far from highways.
- Orbit type ES-L1, GEO, HEO, SSO has the highest first stage successful landing rate.
- Low weighted payloads perform better than those heavier ones in terms of successful launches.
- Launch site KSC LC-39A had the most successful launch counts of all.
- The launch success rate is increasing by years since 2013.
- The Decision Tree classifier is the best performing model for predicting the first stage landing outcome of SpaceX Rocket.

Thank you!