Winning Space Race with Data Science

<Anthony Lo>
<21/08/22>

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

SpaceX is a Multi Billion Corporation that has multiple launches, each costing millions of dollars. The goal of my project is to create a machine learning/ Predicitive analysis pipeline and visualize it to predict if the first stage will land successfully.

- Problems you want to find answers

What will determine the risk of launch failure

Success factors

How do we ensure and maintain low risk by depicting data into code onto visuals

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
- Data from SpaceX API
- Perform data wrangling
- One Hot Encoding
- 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

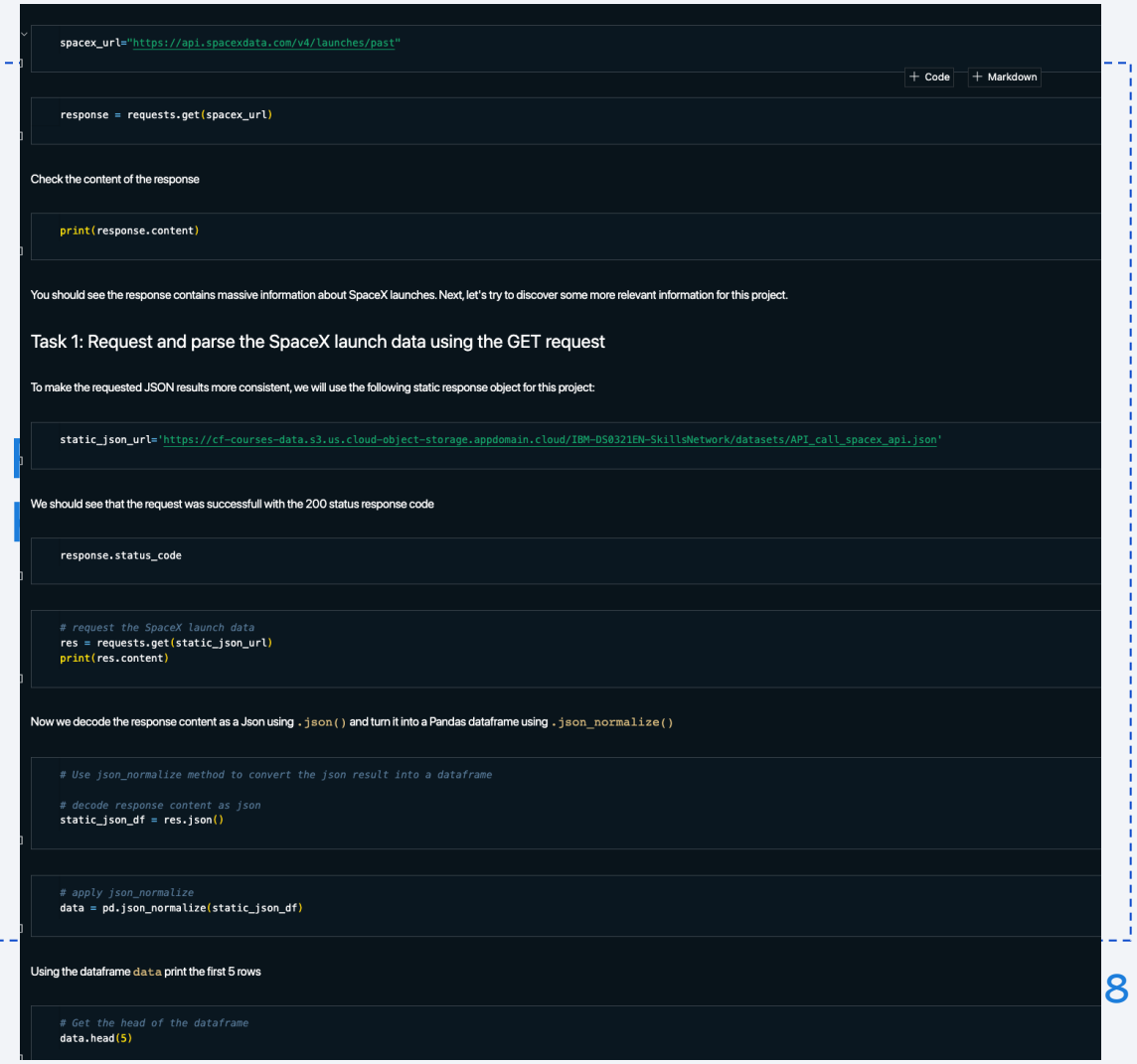
Data Collection

- Describe how data sets were collected.
- SpaceX API is used for data collection
- You need to present your data collection process use key phrases and flowcharts
- Pandas dataframe encoding
- Cleanup
- Beautiful soup method
- Visualise the gathered data simply for future predicitions

Data Collection – SpaceX API

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

- https://github.com/tonesgainz/Capstone-SpaceX/blob/d27b3357ae2481f43cd1cd49330fb13fdf925083/DatacollectAPI.ipynb



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

+ Code + Markdown

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

Check the content of the response

```
print(response.content)
```

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

```
# request the SpaceX launch data
res = requests.get(static_json_url)
print(res.content)
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
# Use json_normalize method to convert the json result into a dataframe

# decode response content as json
static_json_df = res.json()
```

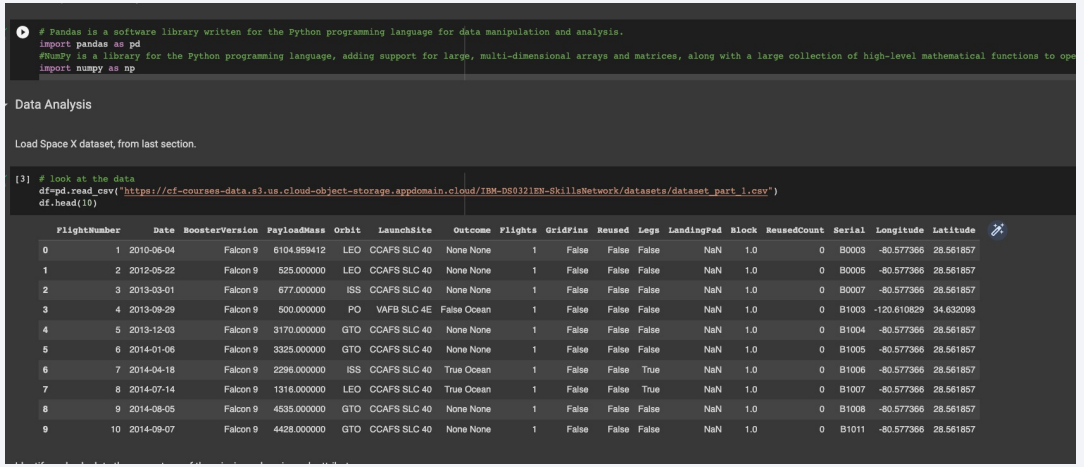
```
# apply json_normalize
data = pd.json_normalize(static_json_df)
```

Using the dataframe `data` print the first 5 rows

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

Data Collection – Scraping & Wrangling

- Launch records with Beautiful soup
- Parsed and pandas dataframe depiction!
- https://github.com/tonesgainz/Capstone-SpaceX/blob/d27b3357ae2481f43cd1cd49330fb13fdf925083/Data_Wrangling.ipynb



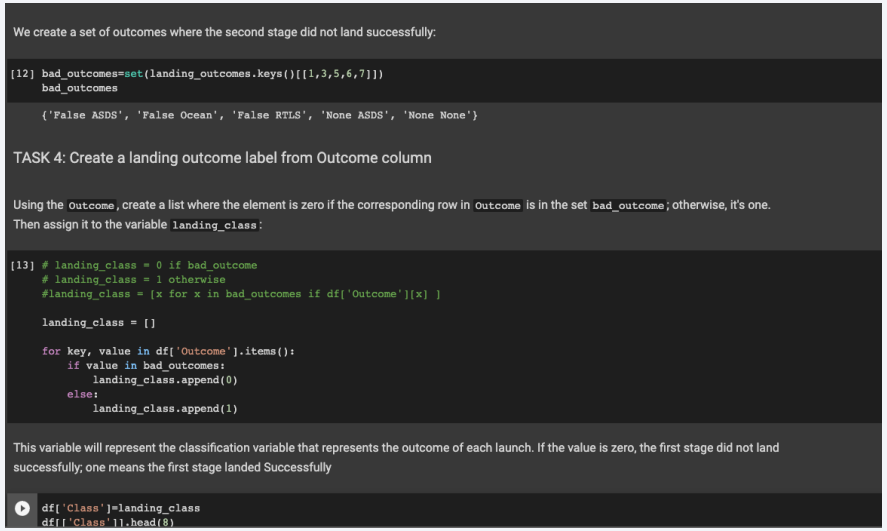
```
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
#NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to ope
import numpy as np
```

Data Analysis

Load Space X dataset, from last section.

```
[3] # look at the data
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005	-80.577366	28.561857
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006	-80.577366	28.561857
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007	-80.577366	28.561857
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008	-80.577366	28.561857
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011	-80.577366	28.561857



We create a set of outcomes where the second stage did not land successfully:

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

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

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one.
Then assign it to the variable `landing_class`:

```
[13] # landing_class = 0 if bad_outcome
     # landing_class = 1 otherwise
     #landing_class = [x for x in bad_outcomes if df['Outcome'][x] ]

     landing_class = []

     for key, value in df['Outcome'].items():
         if value in bad_outcomes:
             landing_class.append(0)
         else:
             landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

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

Data Wrangling

- Results shown through the success rate of pandas df

```
df.head(5)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

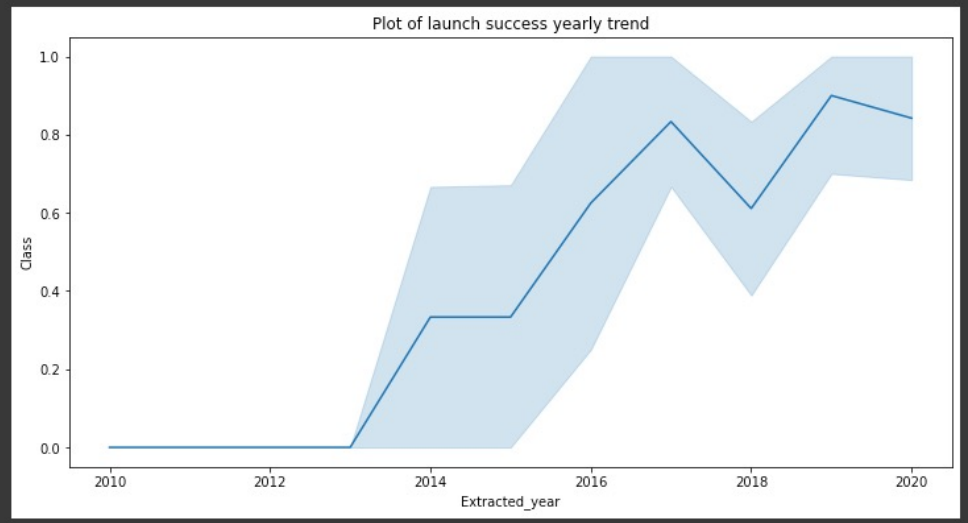
We can use the following line of code to determine the success rate:

```
[16] # determine success rate of launch
     df["Class"].mean()

     0.6666666666666666
```

EDA with Data Visualization

- Visualised the flights and launch sites/launch and formed the relationship for success and types
- https://github.com/tonesgainz/Capstone-SpaceX/blob/5fd37456f90cd62d76b0941f81748ae9769d1532/EDA_with_DataVisualization.ipynb

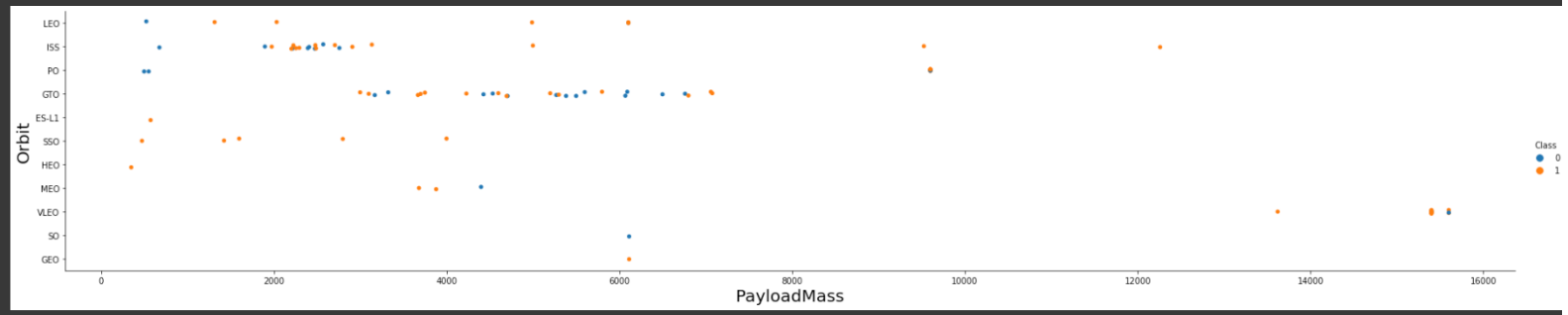
```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
df_copy = df.copy()
df_copy['Extracted_year'] = pd.DatetimeIndex(df['Date']).year

# plot line chart
fig, ax=plt.subplots(figsize=(12,6))
sns.lineplot(data=df_copy, x='Extracted_year', y='Class')
plt.title('Plot of launch success yearly trend');
plt.show()
```



you can observe that the sucess rate since 2013 kept increasing till 2

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

EDA with SQL

- Following the Lab we determined

  Names of unique launch sites in the space mission.

  Total payload mass carried by boosters launched by NASA (CRS)

  Total number of successful and failure mission outcomes

  Tailed landing outcomes in drone ship, their booster version and launch site names.

Build an Interactive Map with Folium

- Markedall launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- Assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Color marker clusters, we identified which launch sites have relatively high success rate.
- Calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

  Link is here

  https://github.com/tonesgainz/Capstone-SpaceX/blob/2e5e37a80eda75d109975595ac1e4558b41f8a9e/InteractiVisual_Analytics_with_Folium.ipynb

Build a Dashboard with Plotly Dash

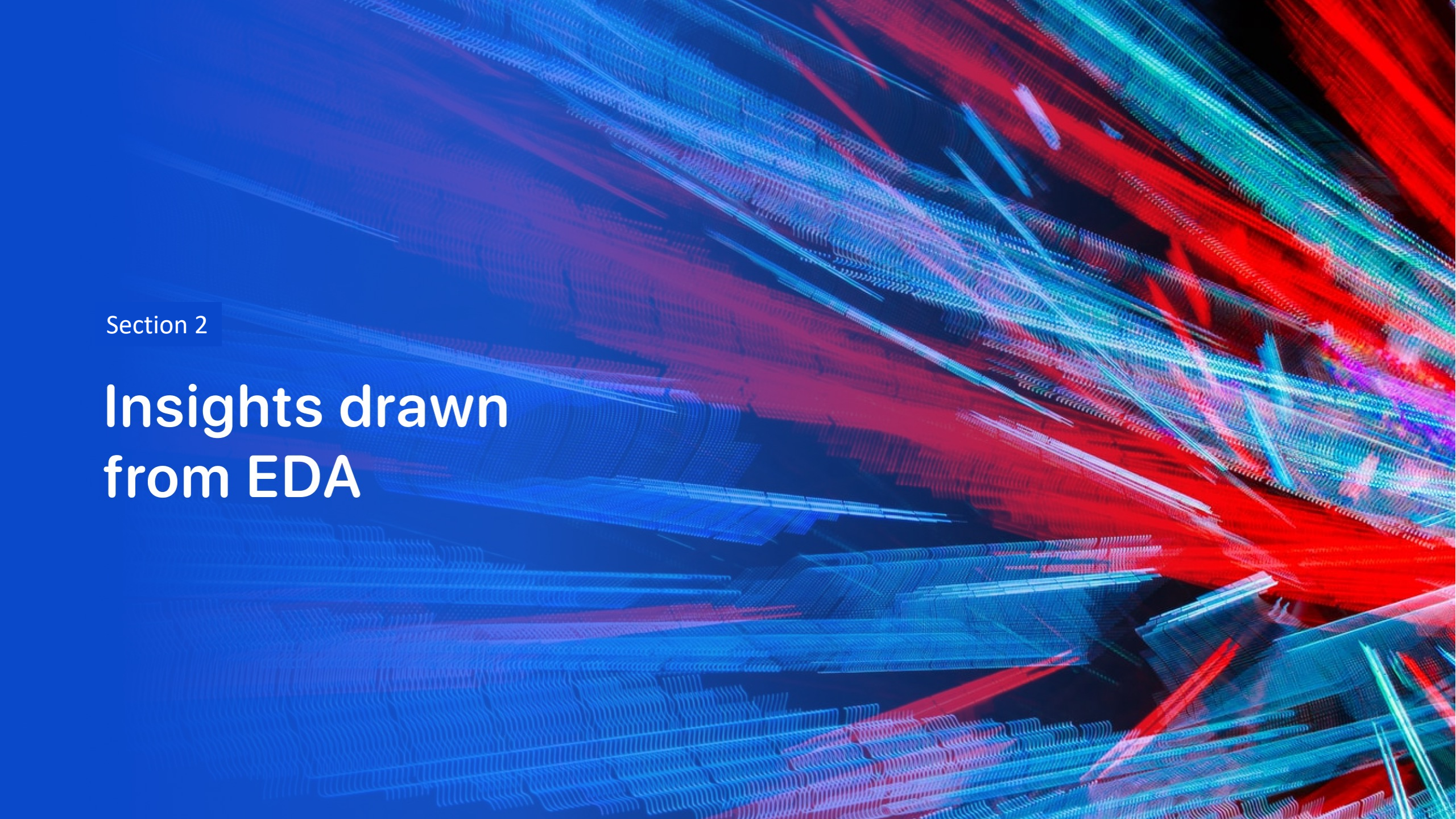
- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook –
- https://github.com/tonesgainz/Capstone-SpaceX/blob/996d27d493eec14699ae042aebb72adb0bdb78d2/app.py

Predictive Analysis (Classification)

- Loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- Built different machine learning models and tune different hyperparameters using GridSearchCV.
- Accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- https://github.com/tonesgainz/Capstone-SpaceX/blob/bfb62de806ae7db362c19d5a097c201f85ce6197/MachineLearning_Prediction.ipynb

Results

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

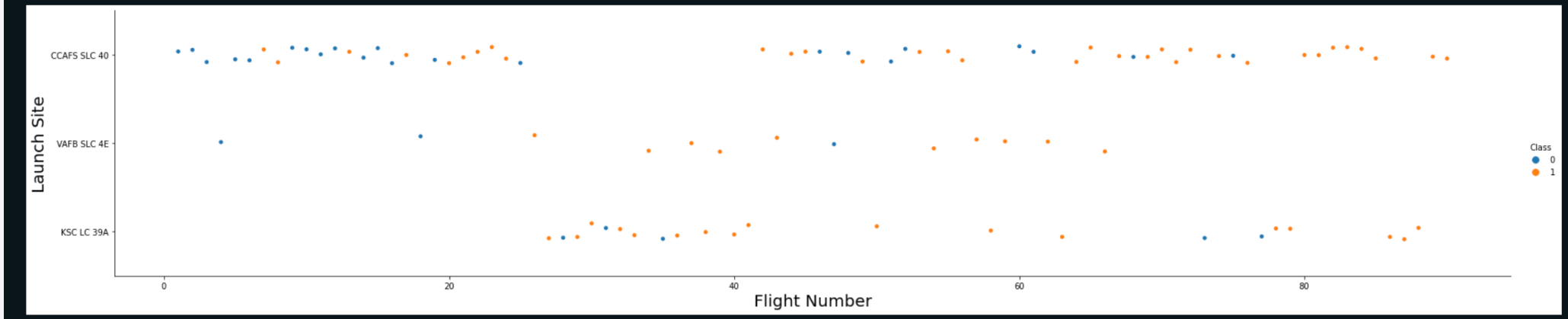

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

```python
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```

Python



Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

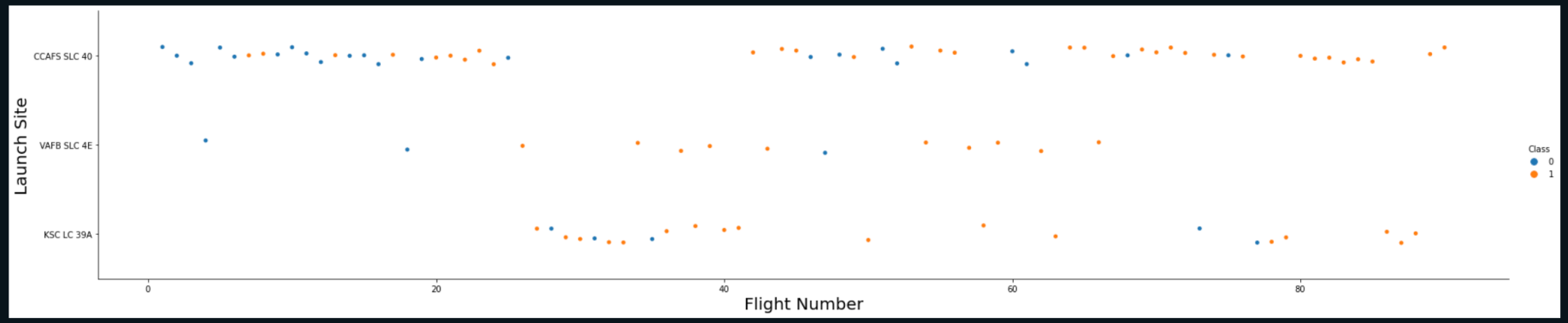
Payload vs. Launch Site

TASK 2: Visualize the relationship between Payload and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

[5] Python

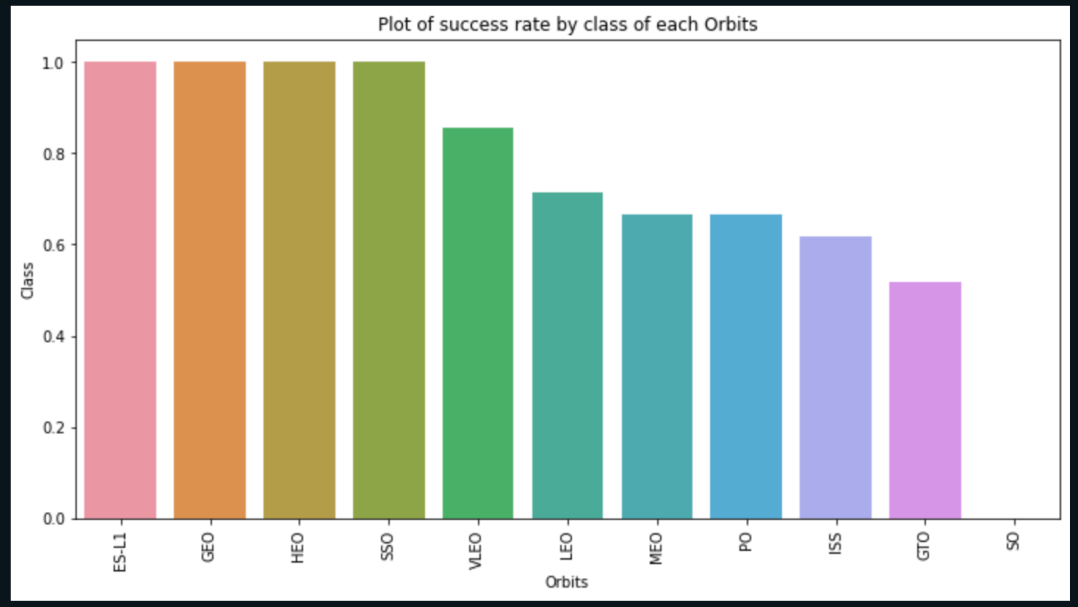


Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

```
# Use groupby method on Orbit column and get the mean of Class column
grouped_orbits = df.groupby(by=['Orbit'])['Class'].mean().sort_values(ascending=False).reset_index()
fig, ax=plt.subplots(figsize=(12,6))
ax = sns.barplot(x = 'Orbit', y = 'Class', data=grouped_orbits)
ax.set_title('Plot of success rate by class of each Orbits', fontdict={'size':12})
ax.set_ylabel('Class', fontsize = 10)
ax.set_xlabel('Orbits', fontsize = 10)
ax.set_xticklabels(ax.get_xticklabels(), fontsize = 10, rotation=90);
```

[8]



Analyze the ploted bar chart try to find which orbits have high success rate.

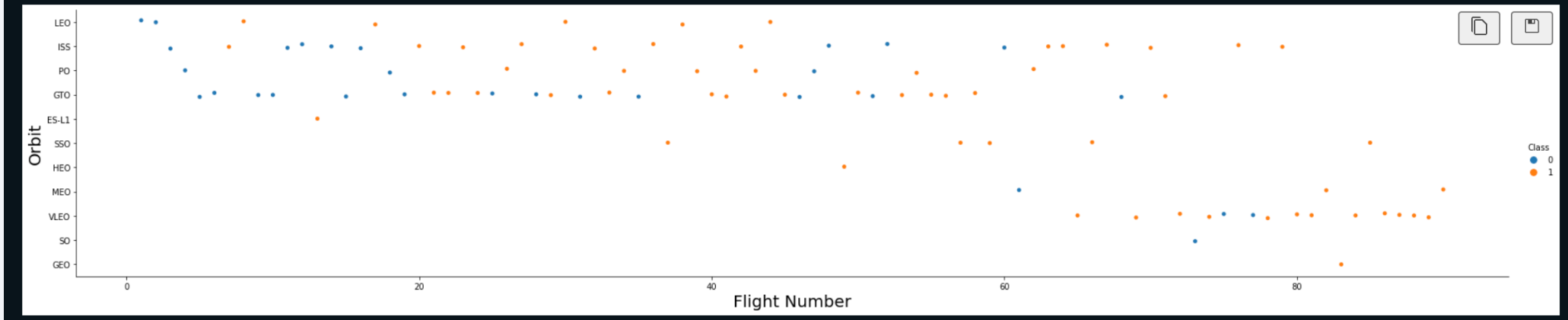
Looking at the plot, we can see that `ES-L1`, `GEO`, `HEO`, `SSO`, and `VLEO` are the Orbits that have high success rate. The `SO` has the least success rate amongst the orbits.

Flight Number vs. Orbit Type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

Python



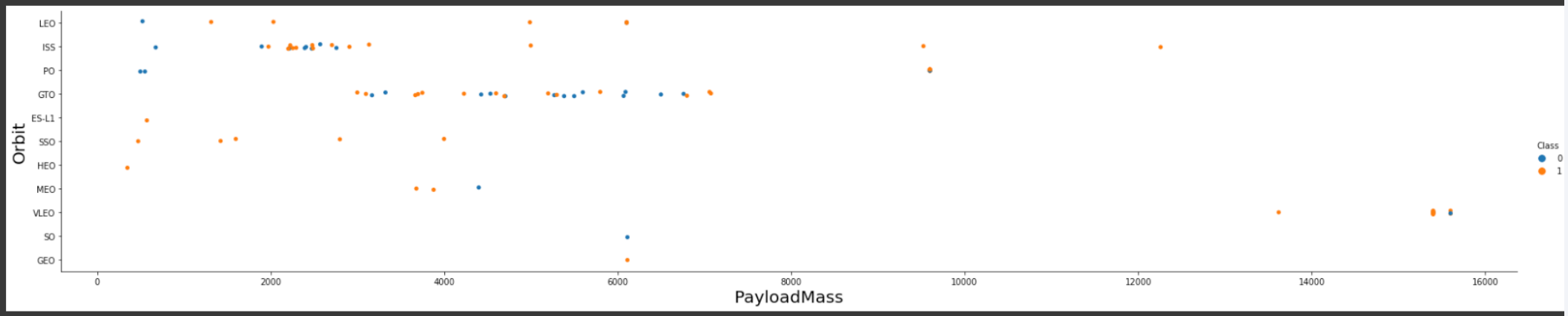
You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

+ Code + Markdown

Payload vs. Orbit Type

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

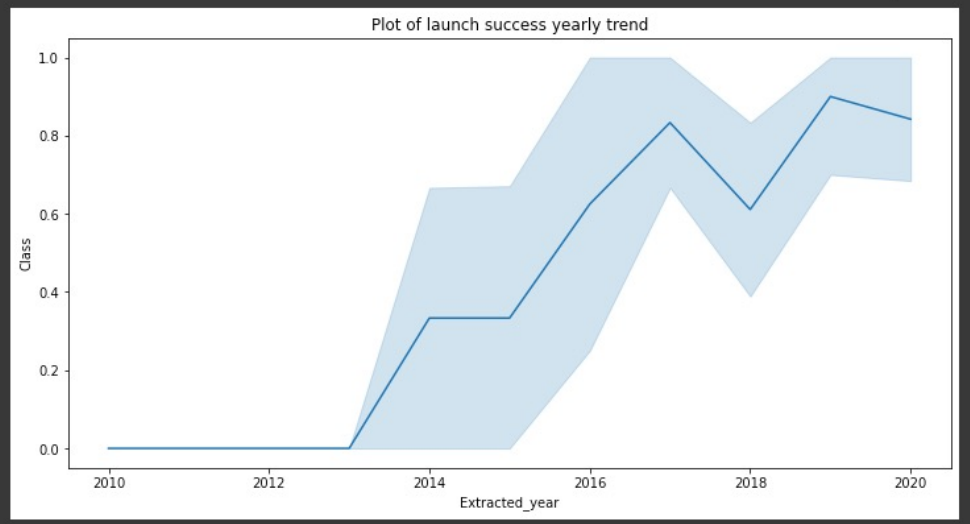


With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
df_copy = df.copy()
df_copy['Extracted_year'] = pd.DatetimeIndex(df['Date']).year

# plot line chart
fig, ax=plt.subplots(figsize=(12,6))
sns.lineplot(data=df_copy, x='Extracted_year', y='Class')
plt.title('Plot of launch success yearly trend');
plt.show()
```



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

- Find the names of the unique launch sites

```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

[4] Python

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610746

Launch Site Names Begin with 'CCA'

```
[12]
df.head(5)
```

Python

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366

We can use the following line of code to determine the success rate:

```
[13]
# determine success rate of launch
df["Class"].mean()
```

Python

```
0.6666666666666666
```

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

```
# Calculate the mean value of PayloadMass column
PayloadMass = pd.DataFrame(data_falcon9['PayloadMass'].values.tolist()).mean(1)
print(PayloadMass)
```

[28] Python

```
0    5919.165341
dtype: float64
```

Average Payload Mass by F9 v1.1

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

- The average payload mass carried by booster version F9 v1.1 as 2928.4

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- I found that the First successful landing using the ground pad was on 22nd December 2015

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

Writing the code with "WHERE" to find the results of successfull drone ship landing

"AND" is used to find the results of successful landings inbetween payload mass of 4000< x <6000

F9 FT B1022

F9 FT B1026

F9 FT B1021.2

F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- % is used which gave me the result of
- 100 Successful and 1 Failed outcome.,

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- I used the WHERE clause and the MAX() function
- F9 B5 1048.4 – F9 B51060.3

Payload mass equates to 15600kg

2015 Launch Records

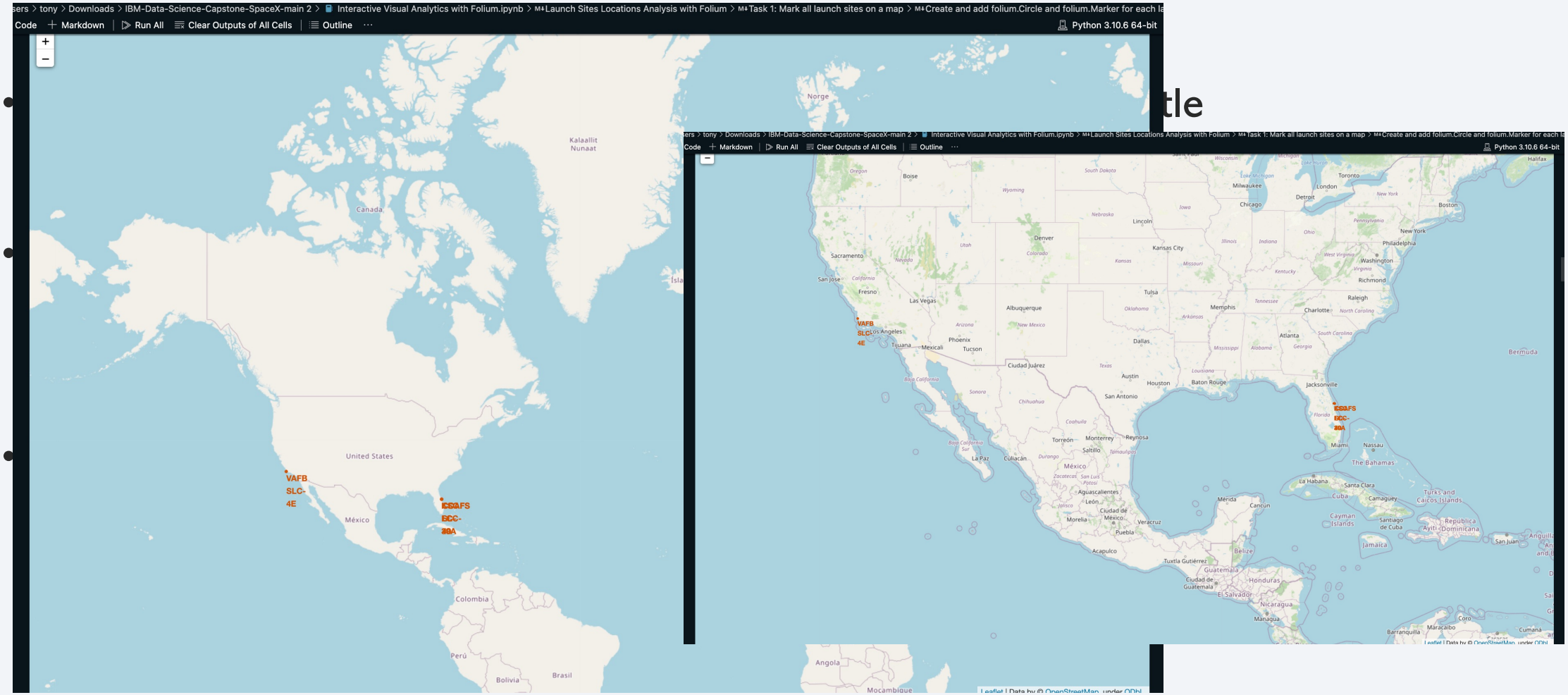
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- To find the failed outcomes I used the WHERE, LIKE, AND, and BETWEEN functions

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

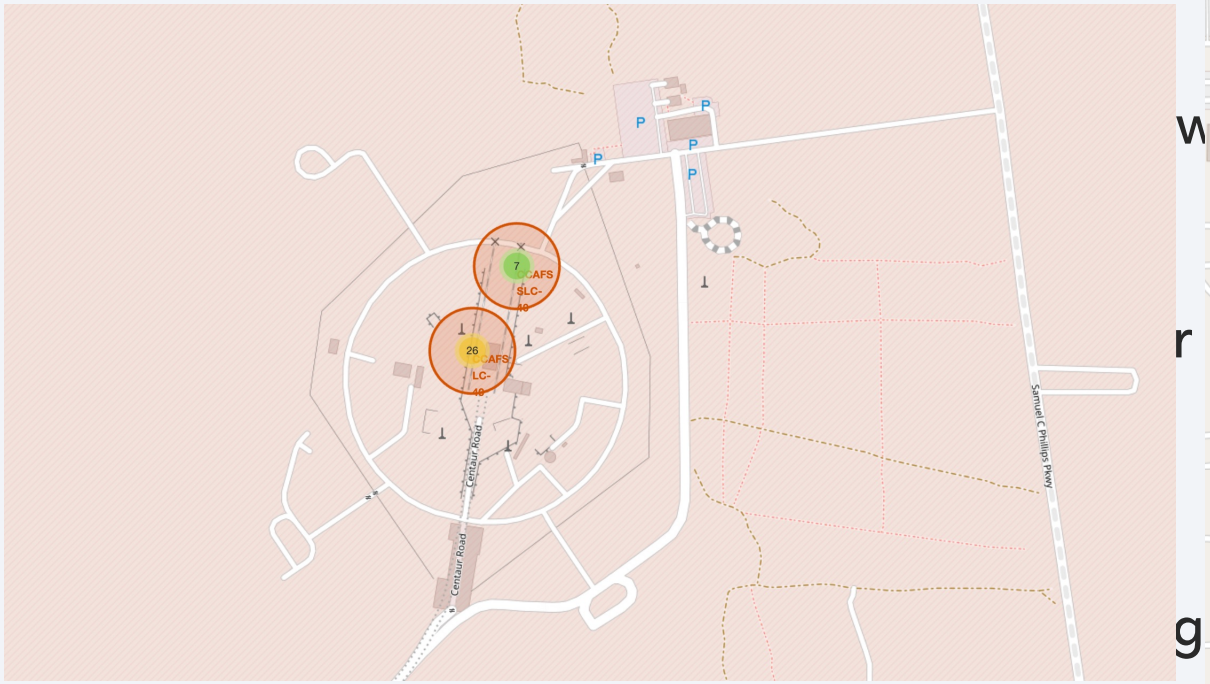
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order
- Using GROUP BY
- And
- ORDER BY

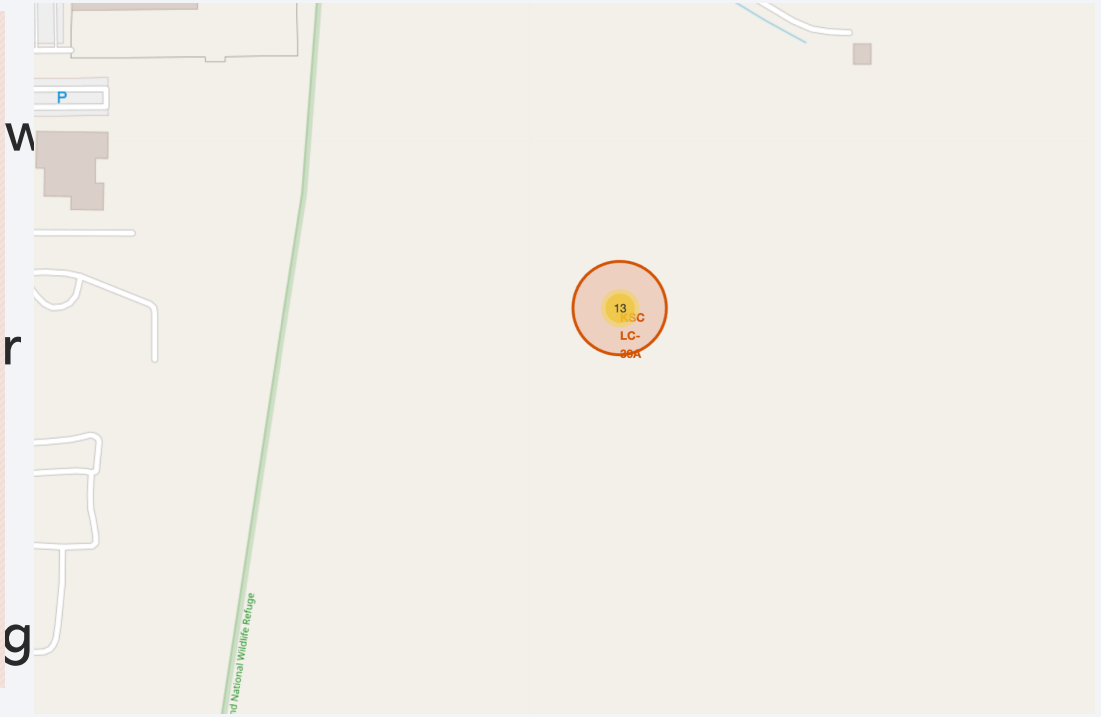

Section 3
Launch Sites
Proximities Analysis

SpaceX Launch Sites Global



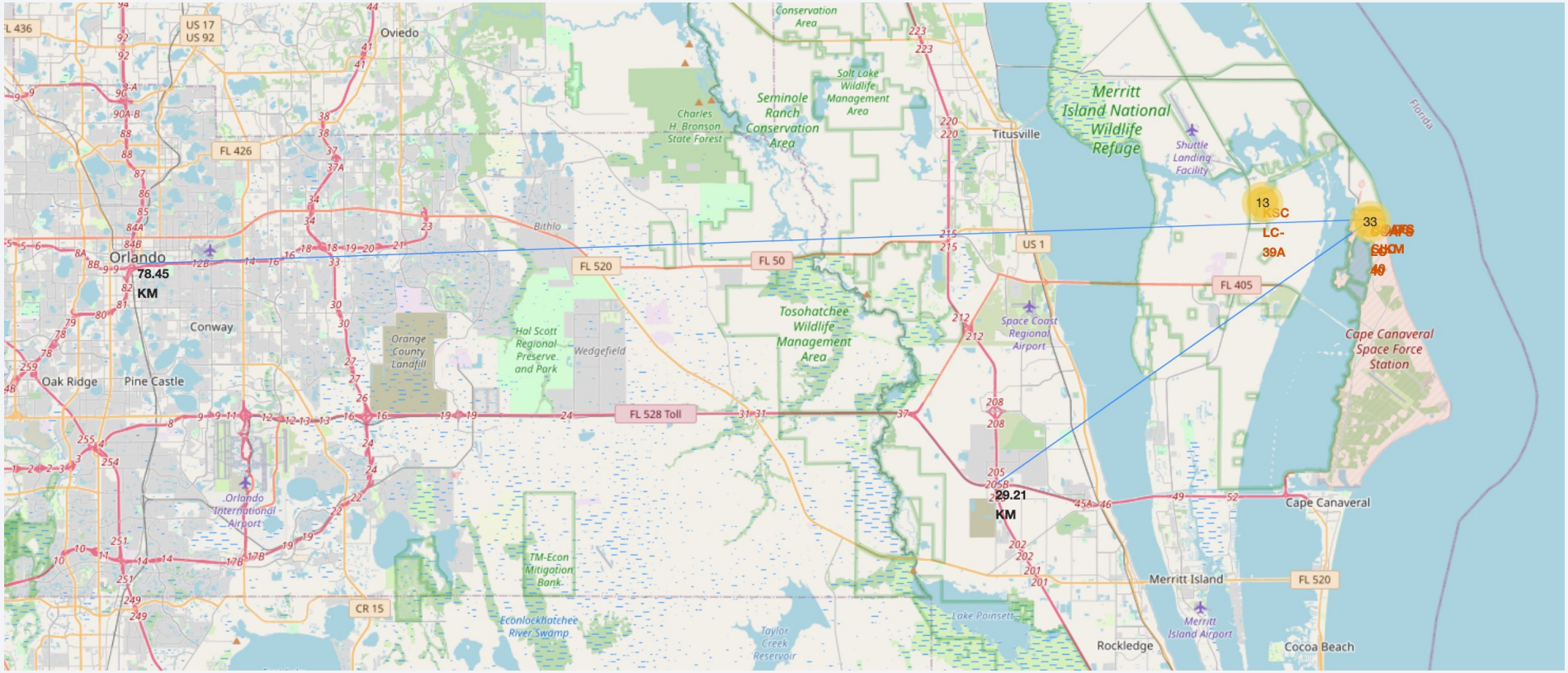
Colored marked Launch Sites



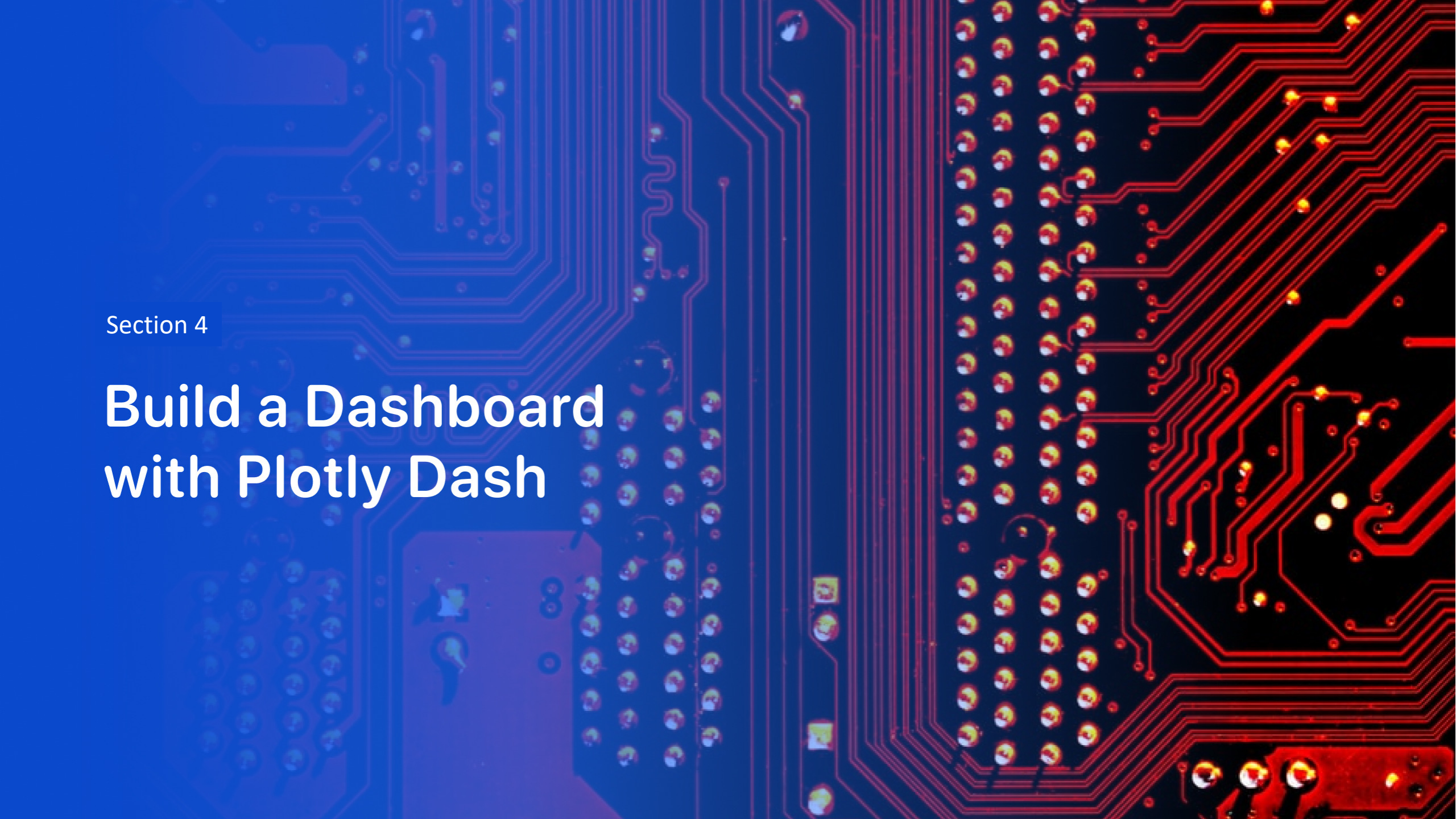



P
Kelp Road
Agena Way
Dix Road
Surf Road
Agena Way
Space Launch
Complex
4
Dix Road
10
VAFB
SLC-
4E
Launch
Tower
Surf Road
Spring Canyon Road

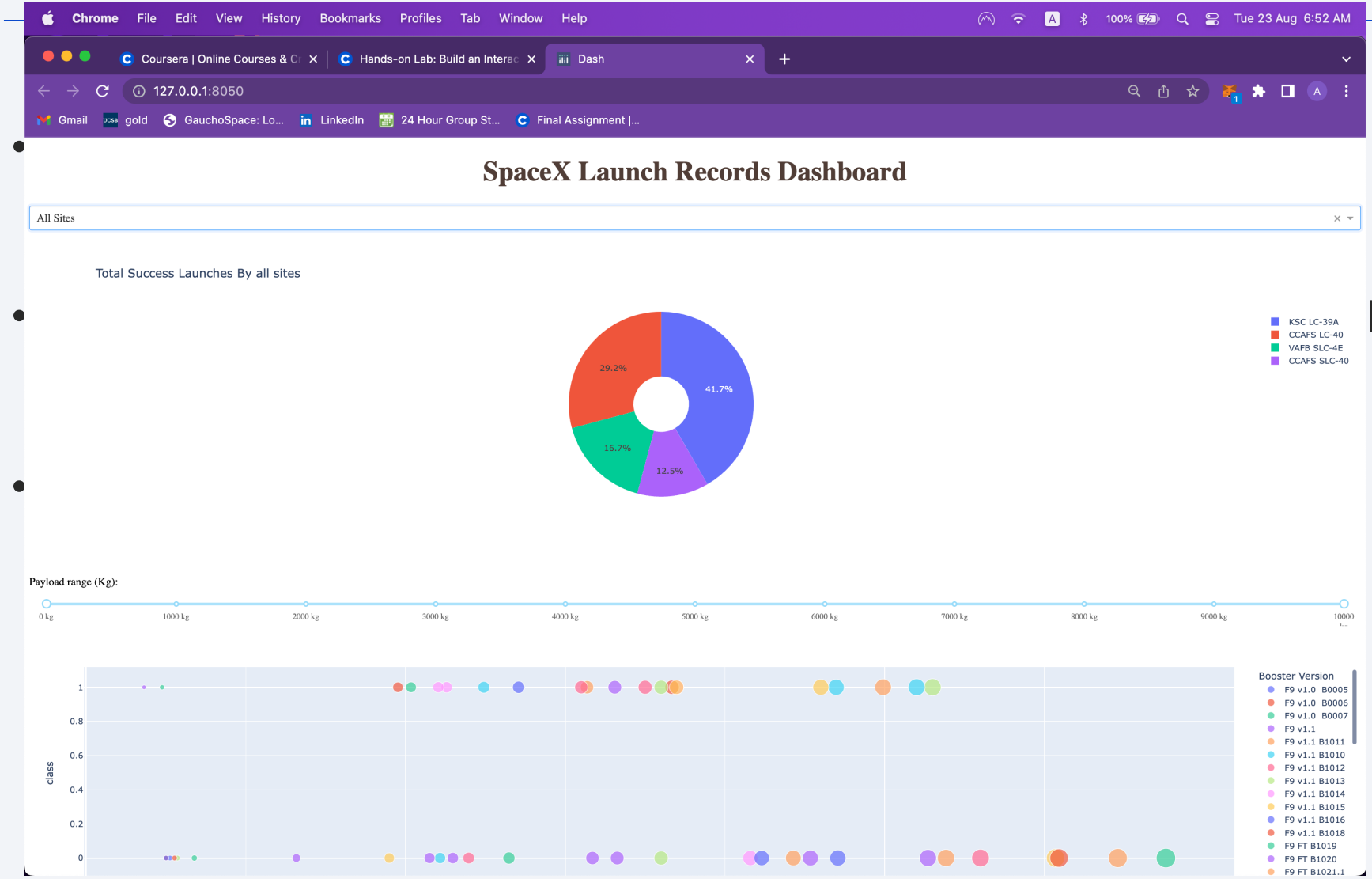
Distance Between Launch Sites & Markpoint



* Are launch sites in close proximity to railways? No
* Are launch sites in close proximity to highways? No
* Are launch sites in close proximity to coastline? Yes
* Do launch sites keep certain distance away from cities? Yes

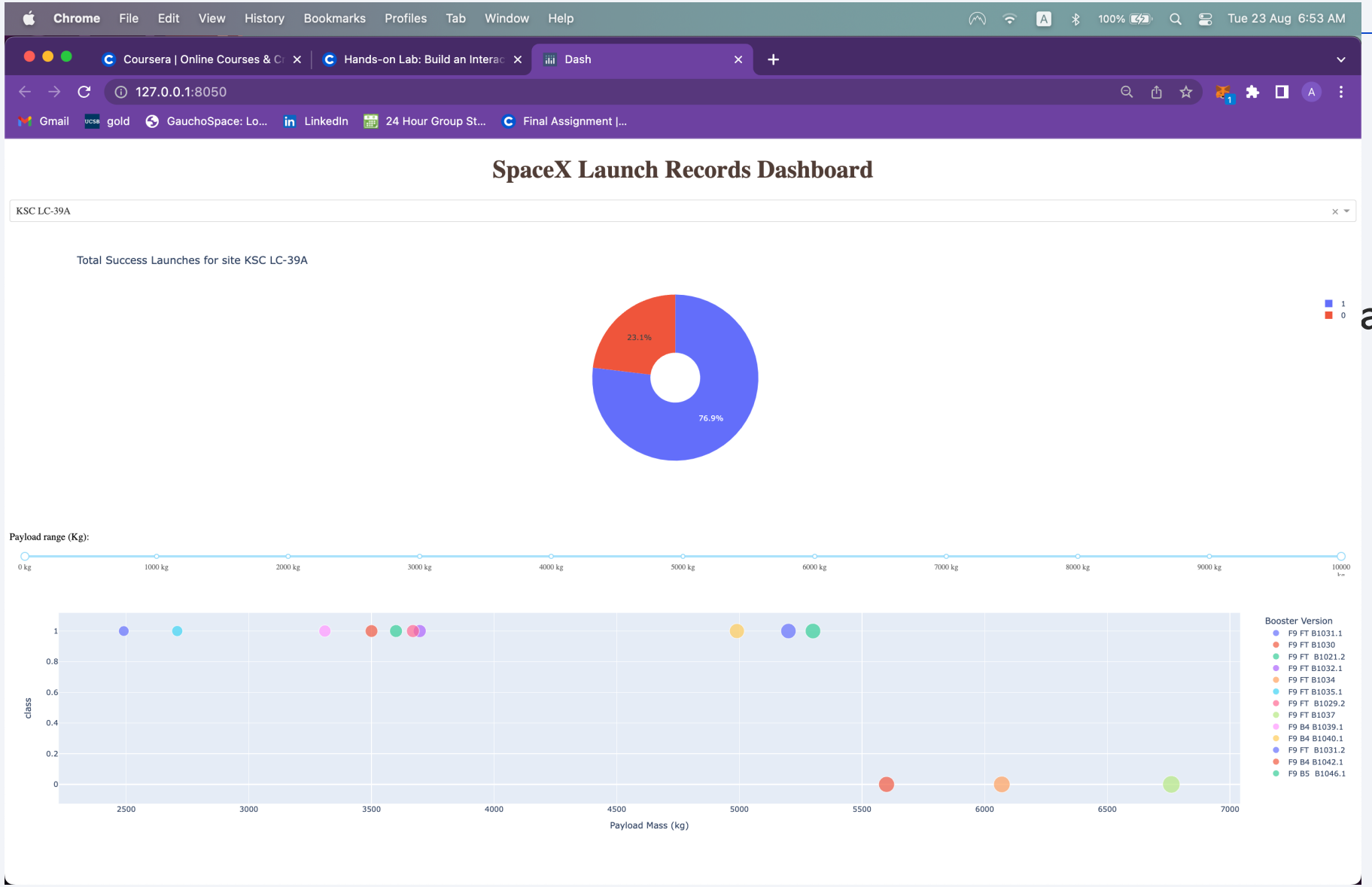

Section 4
Build a Dashboard with Plotly Dash

Plotly Success all sites

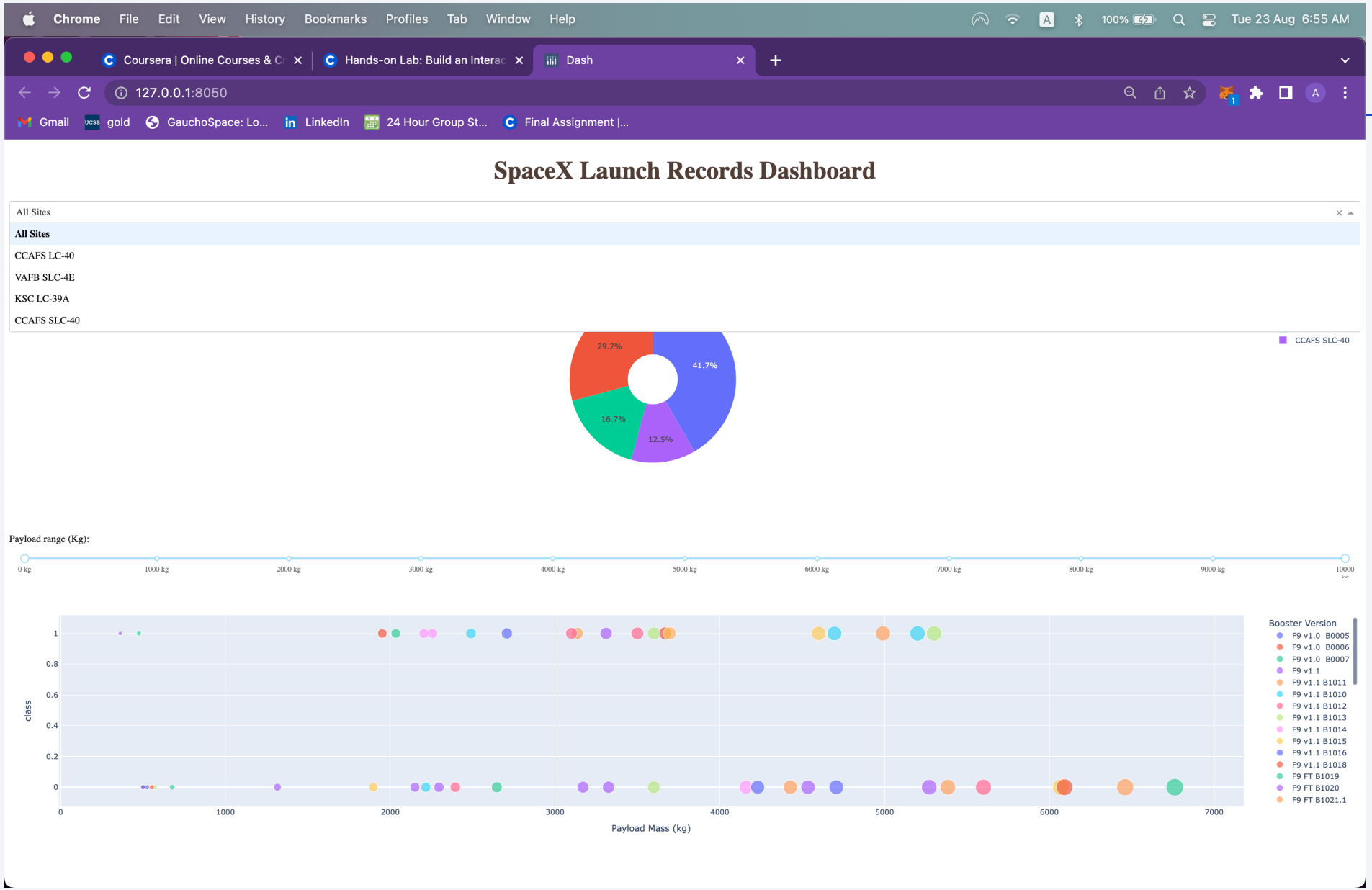


-
- hart
-

Highest Success – KSC LC-39A



launch success


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Tue 23 Aug 6:55 AM
Coursera | Online Courses &
Hands-on Lab: Build an Interac
Dash
127.0.0.1:8050
Gmail
gold
GauchoSpace: Lo...
LinkedIn
24 Hour Group St...
Final Assignment |...
SpaceX Launch Records Dashboard
All Sites
All Sites
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
29.2%
41.7%
16.7%
12.5%
CCAFS SLC-40
Payload range (Kg):
0 kg
1000 kg
2000 kg
3000 kg
4000 kg
5000 kg
6000 kg
7000 kg
8000 kg
9000 kg
10000
class
Payload Mass (kg)
Booster Version
F9 v1.0 B0005
F9 v1.0 B0006
F9 v1.0 B0007
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1010
F9 v1.1 B1012
F9 v1.1 B1013
F9 v1.1 B1014
F9 v1.1 B1015
F9 v1.1 B1016
F9 v1.1 B1018
F9 FT B1019
F9 FT B1020
F9 FT B1021.1

Section 5

Predictive Analysis (Classification)

Classification Accuracy (Best Method)

Find the method performs best:

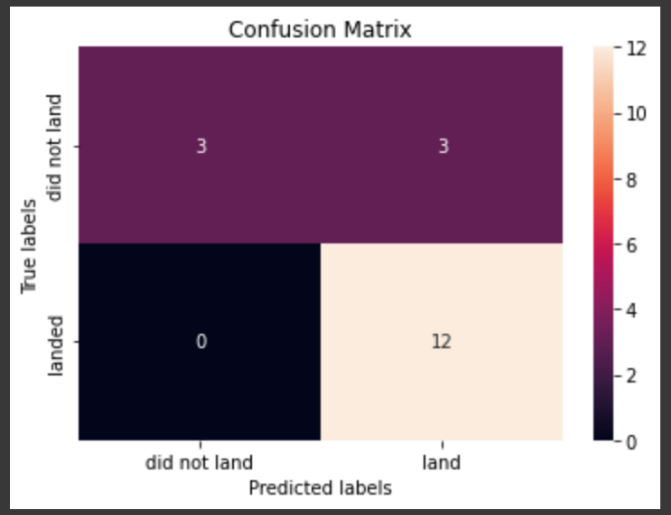
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

We can plot the confusion matrix

```
yhat_knn = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test, yhat_knn)
```



an

Find the method performs best:

```
[25] models = {'KNeighbors':knn_cv.best_score_,
                   'DecisionTree':tree_cv.best_score_,
                   'LogisticRegression':logreg_cv.best_score_,
                   'SupportVector': svm_cv.best_score_}

     bestalgorithm = max(models, key=models.get)
     print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
     if bestalgorithm == 'DecisionTree':
         print('Best params is :', tree_cv.best_params_)
     if bestalgorithm == 'KNeighbors':
         print('Best params is :', knn_cv.best_params_)
     if bestalgorithm == 'LogisticRegression':
         print('Best params is :', logreg_cv.best_params_)
     if bestalgorithm == 'SupportVector':
         print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- High Success goes to KSC LC39A
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!