IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Fábio Rodrigo C da Silva
March 9, 2024

Outline

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

Executive Summary

- Objective: predict if the Falcon 9 first stage will land successfully.
- Because SpaceX can reuse the first stage during launch, it's possible to save millions of dollars. So, this project intends to predict if the first stage will land successfully and, with that information, determine the cost of a launch. This prediction is based on data from previous launch.
- The data was collected using the SpaceX API. After cleaned, it was performed an exploratory data analysis using graphs and SQL queries. A visual analytics system was created to better understanding the relation between variables.
- At the final step, four machine learning classification models were used to predict the outcome of launches. The decision tree model was the best, and performed with an accuracy of 86% on training data, and 83% on test data.

GitHub

https://github.com/fabiorodrigocs/IBMCapstone

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
- Perform data wrangling
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

- The data were collected from the SpaceX API, and cleaned the requested data to make sure it was in the correct format.
- So, it resulted in a Pandas data frame that contains the dataset.
- It was performed through web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy Launches.
- It was performed Exploratory Data Analysis (EDA) to find patterns in the data and determine what would be the label for training supervised models.

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts
- 1) Import libraries and auxiliary functions
- 2) Now let's start requesting rocket launch data from SpaceX API with the following URL
- 3) Request and parse the SpacX launch data using the GET request
- 4) Filter the dataframe to only include Falcon 9 launches

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

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

[9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-sto

[10]: response.status_code

[11]: # Use json_normalize meethod to convert the json result into a dataframe
      data = pd.json_normalize(response.json())
```

Link: https://github.com/fabiorodrigocs/IBMCapstone/blob/10b29ca34c976d29672521c0a6b5b6dcd727df5c/W1%20-%201%20-%20jupyter-labs-spacex-data-collection-api.ipynb

Data Collection – SpaceX API

- Pandas dataframe after dealing with missing values.

```
[26]: # Show the head of the dataframe
      dataframe.head()
```

[26]: ,,,

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	L
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	Fa
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	Fa
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	Fa
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	Fa
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	Fa

Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/10b29ca34c976d29672521c0a6b5b6dcd727df5c/W1%20-%201%20-%20jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

Web scrap Falcon 9 launch records with BeautifulSoup:

1) Extract a Falcon 9 launch records HTML table from Wikipedia

2) Parse the table and convert it into a Pandas data frame

```
In [24]: df
```

Out[24]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10
...	...	...	...	...	...	...	...	...	...	...	...
222	117	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1051.10	Success	9 May 2021	06:42
223	118	KSC	Starlink	~14,000 kg	LEO	SpaceX	Success\n	F9 B5B1058.8	Success	15 May 2021	22:56
224	119	CCSFS	Starlink	15,600 kg	LEO	NASA	Success\n	F9 B5B1063.2	Success	26 May 2021	18:59
225	120	KSC	SpaceX CRS-22	3,328 kg	LEO	Sirius XM	Success\n	F9 B5B1067.1	Success	3 June 2021	17:29
226	121	CCSFS	SXM-8	7,000 kg	GTO	NaN	NaN	F9 B5	NaN	6 June 2021	04:26

227 rows × 11 columns

Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/10b29ca34c976d29672521c0a6b5b6dcd727df5c/W1%20-%202%20-%20jupyter-labs-webscraping.ipynb

Data Wrangling

- Some of the rows are missing values in our dataset, specially in PayloadMass column.
- The LandingPad column retained None values to represent when landing pads were not used.
- It was calculated the mean for the PayloadMass using the .mean().
- Then It was used the mean and the .replace() function to replace np.nan values in the data with the mean calculated.

In [33]:

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

Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/10b29ca34c976d29672521c0a6b5b6dcd727df5c/W1%20-%201%20-%20jupyter-labs-spacex-data-collection-api.ipynb

Data Wrangling

- 1): Calculated the number of launches on each site.

```
Out[11]: CCAFS SLC 40    55
         KSC LC 39A      22
         VAFB SLC 4E     13
```

- 2) Calculated the number and occurrence of each orbit.

```
Out[13]: GTO      27
         ISS      21
         VLEO     14
         PO        9
         LEO       7
         SSO       5
         MEO       3
         ES-L1     1
         HEO       1
         SO        1
         GEO       1
         Name: Orbit, dtype: int64
```

Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/10b29ca34c976d29672521c0a6b5b6dcd727df5c/W1%20-%201%20-%20jupyter-labs-spacex-data-collection-api.ipynb

Data Wrangling

- 3) Calculated the number and occurence of mission outcome of the orbits.

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

- 4): Created a landing outcome label from Outcome column

Out[21]:

	Outcome	Class
0	None None	0
1	None None	0
2	None None	0
3	False Ocean	0
4	None None	0
5	None None	0
6	True Ocean	1
7	True Ocean	1

Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/10b29ca34c976d29672521c0a6b5b6dcd727df5c/W1%20-%201%20-%20jupyter-labs-spacex-data-collection-api.ipynb

EDA with Data Visualization

Graphics used:

- **Scatter plot:** used to visualize the relationship between numerical variables and to check whether there is a correlation between them or identify patterns of points on the graph.

- **Bar plot:** represents the distribution of a categorical variable through bars. Each bar represents a specific category, and the height of the bar indicates the frequency or proportion of that category. The bar plot is useful for comparing the frequency or proportion between different categories and identifying patterns.

Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/10b29ca34c976d29672521c0a6b5b6dcd727df5c/W2%20-%202%20-%20jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

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

  %sql SELECT DISTINCT "LauncH_Site" FROM SPACEXTABLE;

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

  %sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 4;

  %sql SELECT DISTINCT "Landing_Outcome" FROM SPACEXTABLE;

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

  %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)';

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

  %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1%';

- **List the date when the first succesful landing outcome in ground pad was acheived.**

  %sql SELECT MIN(Date) FROM SPACEXTABLE WHERE "Mission_Outcome" = 'Success';

Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/10b29ca34c976d29672521c0a6b5b6dcd727df5c/W2%20-%201%20-%20jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA with SQL

- **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" FROM SPACEXTABLE WHERE "Mission_Outcome" = 'Success' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

- **List the total number of successful and failure mission outcomes**

```
%sql SELECT DISTINCT "Mission_Outcome" FROM SPACEXTABLE;
```

```
%sql SELECT COUNT(*) AS Success_Count, (SELECT COUNT(*) FROM SPACEXTABLE WHERE Mission_Outcome NOT LIKE 'Success%') AS Not_Success_Count FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Success%';
```

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

```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
```

Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/10b29ca34c976d29672521c0a6b5b6dcd727df5c/W2%20-%201%20-%20jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA with SQL

- **List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015**

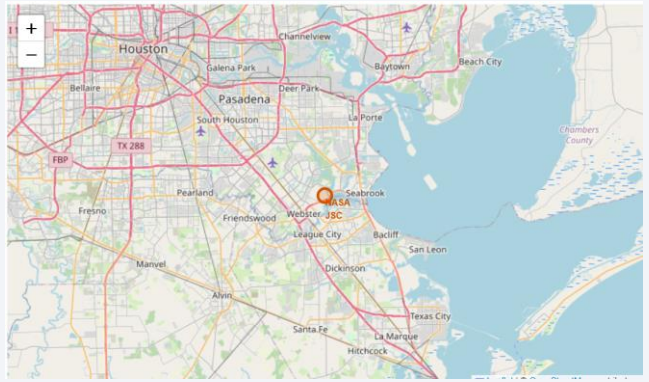
  %sql SELECT SUBSTR(Date, 6, 2) AS Month, "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE SUBSTR(Date, 0, 5) = '2015' AND "Landing_Outcome" = 'Failure (drone ship)';

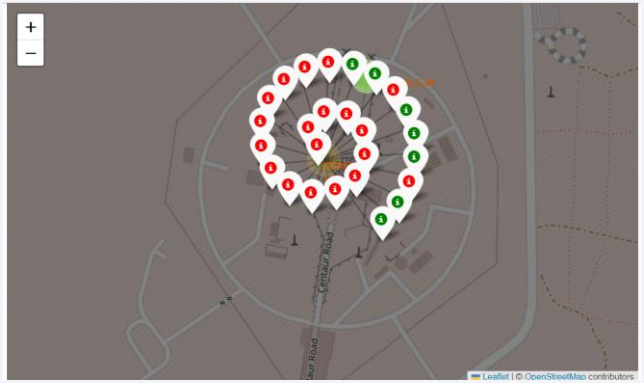
- **Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.**
- %sql SELECT Landing_Outcome, COUNT(*) AS Count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY Count DESC;

Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/10b29ca34c976d29672521c0a6b5b6dcd727df5c/W2%20-%201%20-%20jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- It was marked all launch sites on a map.
- It was used folium.Circle to add a highlighted circle area with a text label on a specific coordinate.
- It was created a red circle at NASA Johnson Space Center's coordinate with a popup label showing its name.
- For each launch result in spacex_df data frame, It was added a folium.Marker to marker_cluster.
- It was created a `folium.PolyLine` object using the coastline coordinates and launch site coordinate.





Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/709ccfcfa4077e8da37fd6663a3fe171a1eafa5d/W3-1%20lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- The dashboard contains components such as:
  - a dropdown list
  - a range slider
  - a pie chart
  - a scatter point chart
- It was added a Launch Site Drop-down Input Component and a a callback function to render success-pie-chart based on selected site dropdown.
- Besides, It was added a Range Slider to Select Payload and a callback function to render the success-payload-scatter-chart scatter plot

Predictive Analysis (Classification)

It was created a machine learning pipeline to predict if the first stage will land given the data from the preceding labs.

1 – It was created a NumPy array from the column **Class** in data (variable Y)
2 – It was **standardized** the data in X
3 – It was used train_test_split to **split** the data X and Y into training (80%) and test (20%) data.
4 – It was created a **Machine Learning object** and a **Grid Search** to its parameters
5 – It was **fitted** the object with train data to find the best parameters
6 – It was calculated the **accuracy** on the test data
7 – It was evalueted the **confusion matrix**

The Machine Models used were:

- Logistic Regression
- SVC (Support Vector Machine)
- Decidion Tree Classifier
- KNN (k-nearest neighbors)

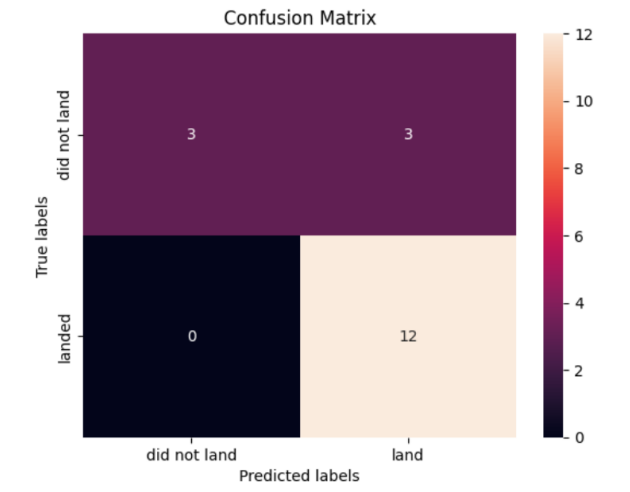
Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/8e13fa89cc9baa3a954b574fac4906e1c4684ef4/W4%20-%20SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20B.ipynb

Results

- Logistic Regression

```
[16]: print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
      print("accuracy :",logreg_cv.best_score_)
      train_accuracy = logreg_cv.best_score_

tuned hpyerparameters :(best parameters)  {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8196428571428571
```



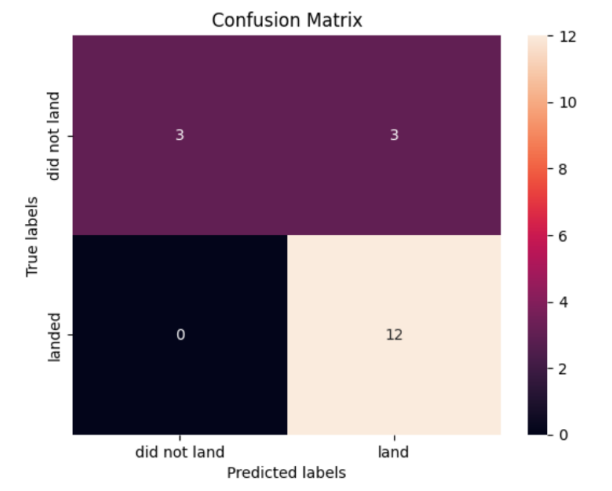
Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/8e13fa89cc9baa3a954b574fac4906e1c4684ef4/W4%20-%20SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20B.ipynb

Results

- SVC (Support Vector Machine)

```
[24]: print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
      print("accuracy :",svm_cv.best_score_)
      train_accuracy = svm_cv.best_score_
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.03162277660168379, 'gamma': 0.001, 'kernel': 'linear'}
accuracy : 0.8196428571428571
```



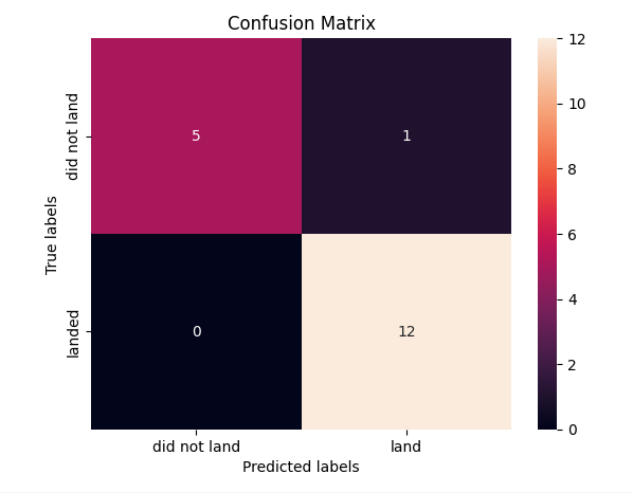
Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/8e13fa89cc9baa3a954b574fac4906e1c4684ef4/W4%20-%20SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20B.ipynb

Results

- Decision Tree Classifier

```
[31]: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
      print("accuracy :",tree_cv.best_score_)
      train_accuracy = tree_cv.best_score_
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'random'}
accuracy : 0.8625
```



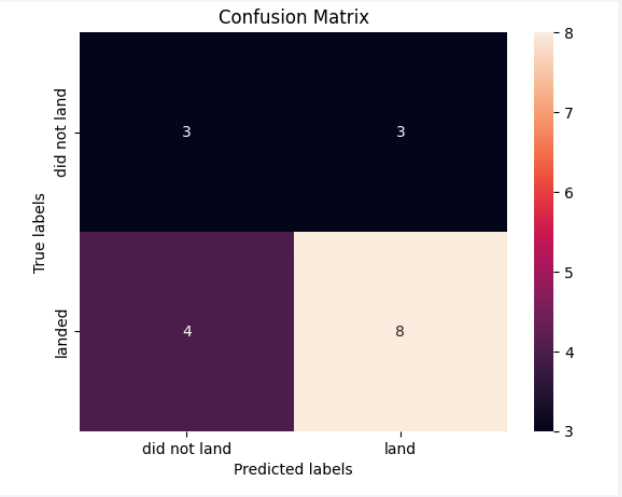
Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/8e13fa89cc9baa3a954b574fac4906e1c4684ef4/W4%20-%20SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20B.ipynb

Results

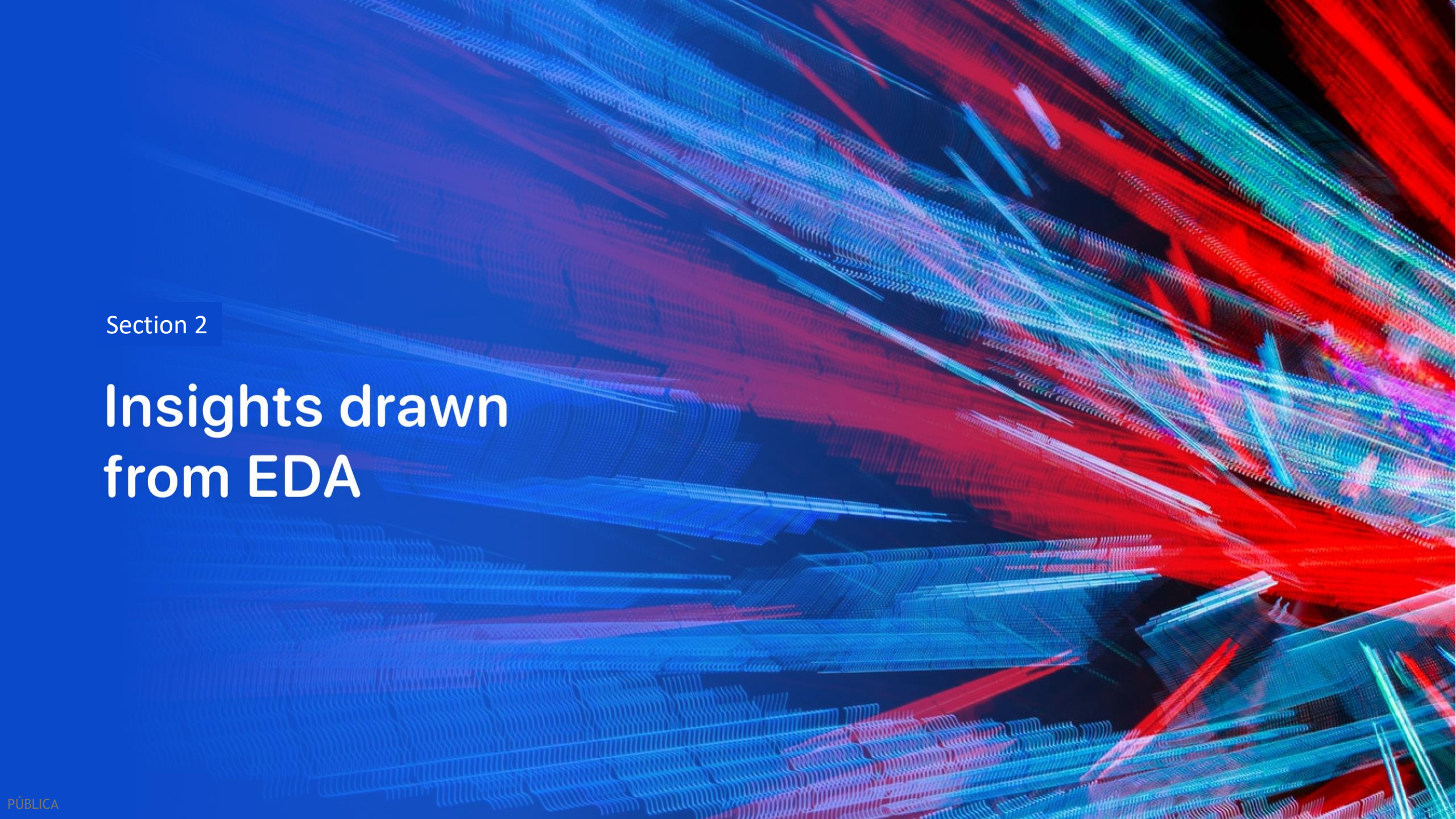
- KNN (k-nearest neighbors)

```
[37]: print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
      print("accuracy :",knn_cv.best_score_)
      train_accuracy = knn_cv.best_score_
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 3, 'p': 1}
accuracy : 0.6642857142857143
```



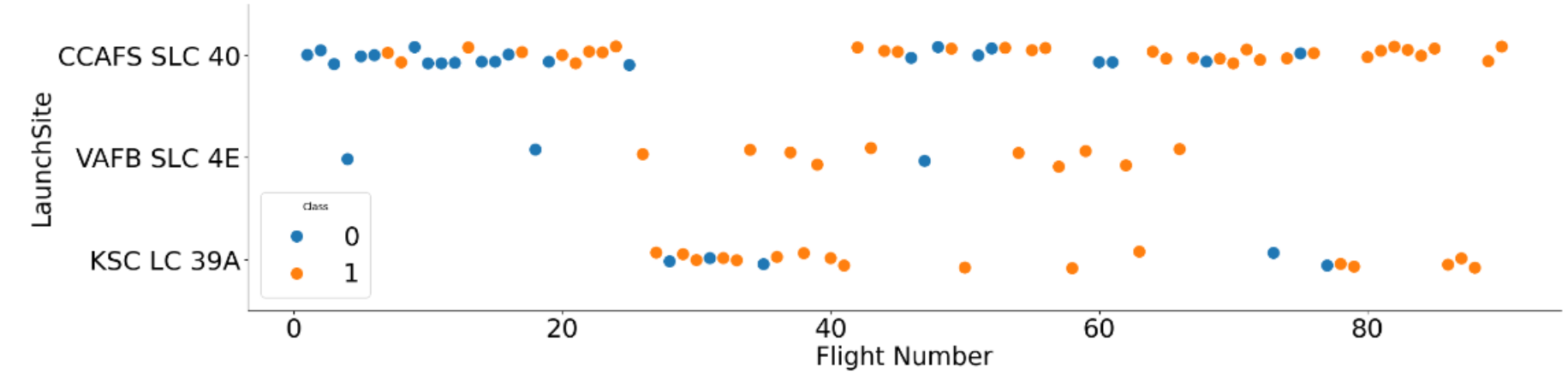
Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/8e13fa89cc9baa3a954b574fac4906e1c4684ef4/W4%20-%20SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20B.ipynb



Section 2

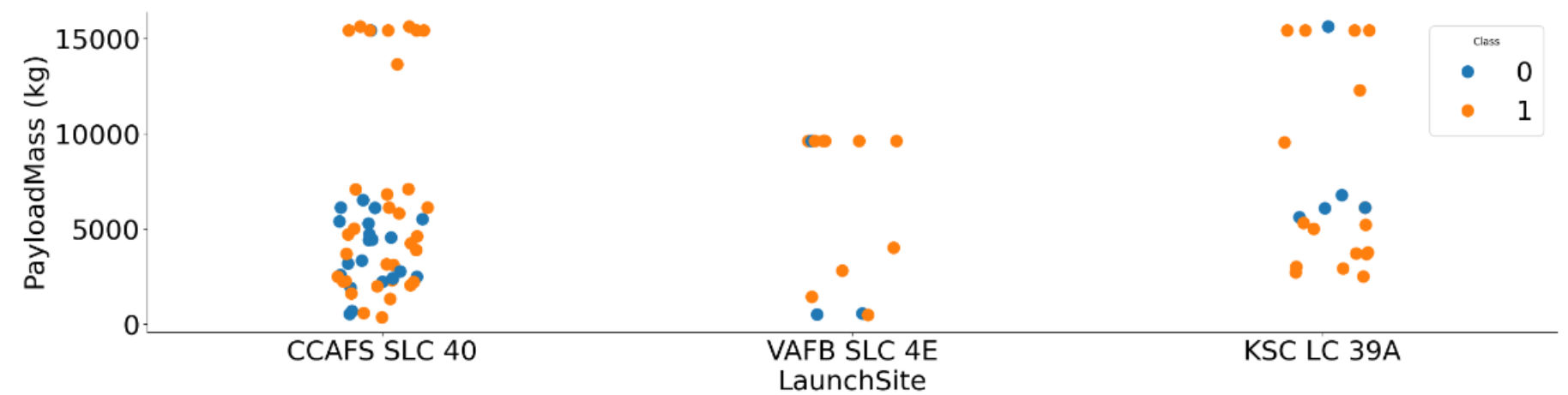
Insights drawn from EDA

Flight Number vs. Launch Site



- As can be seen in the graph, launches on the VAFB SLC 4E website were more successful.
- Furthermore, as the number of launches increases, the number of successes increases. This can be clearly observed in CCAFS SLC 40, where the first launches resulted in failure and the last ones in success.

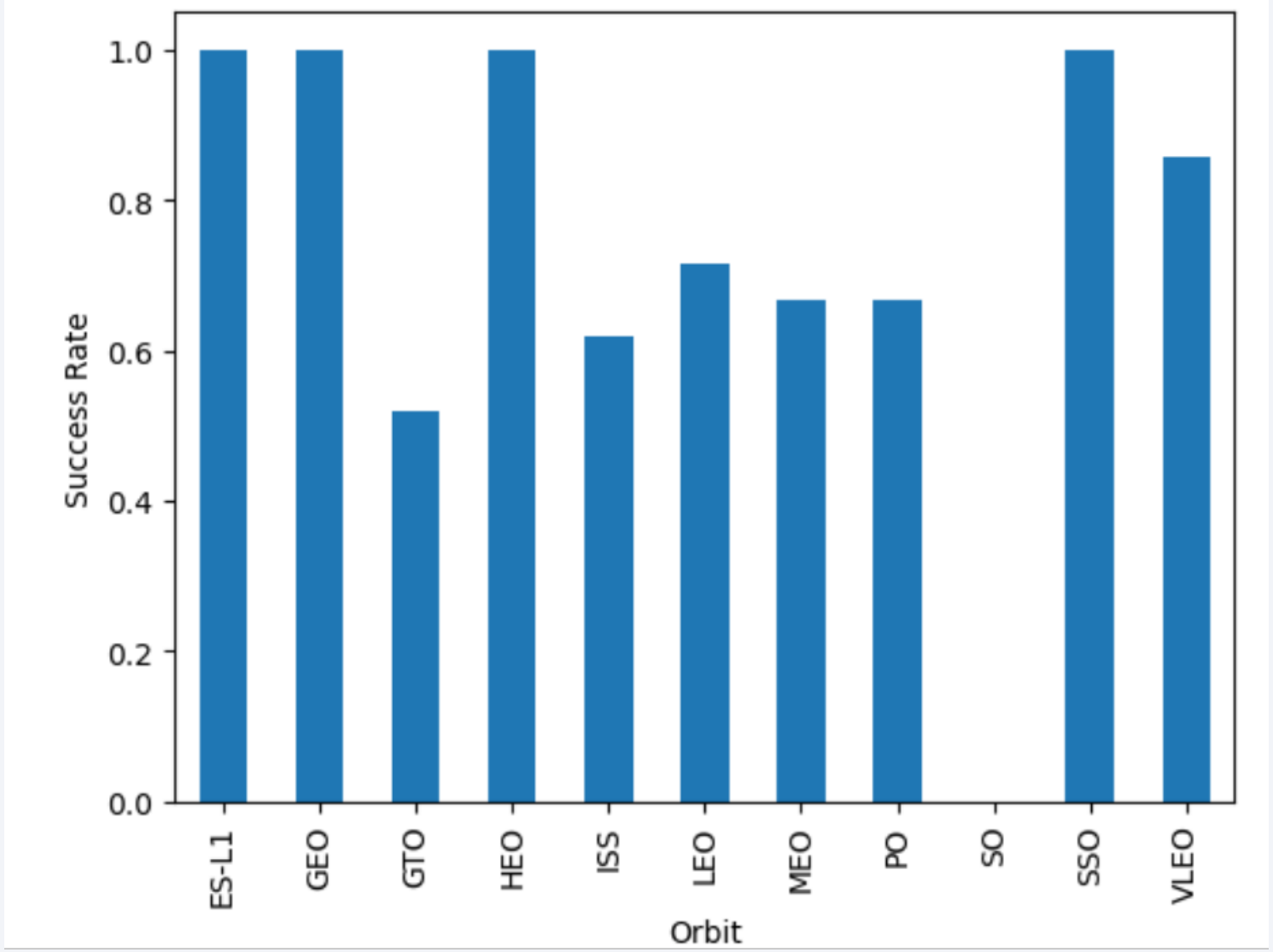
Payload vs. Launch Site



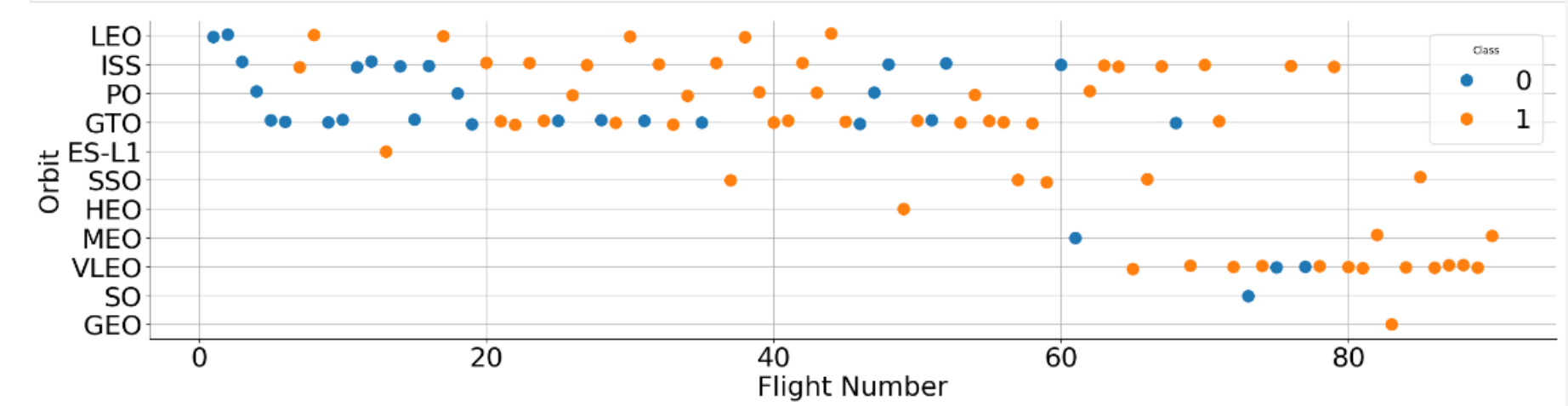
- As can be seen in the graph, the largest number of launches occurred on the CCFAS SLC 40 site.
- A higher payload mass indicates that the chance of success is greater.
- VABF launchsite had the lowest number of launches.

Success Rate vs. Orbit Type

- As can be seen in the graph, a high success rate was achieved for the ES-L1, GEO, HEO and SSO orbits.
- The FTO and ISS orbits had a lower success rate.

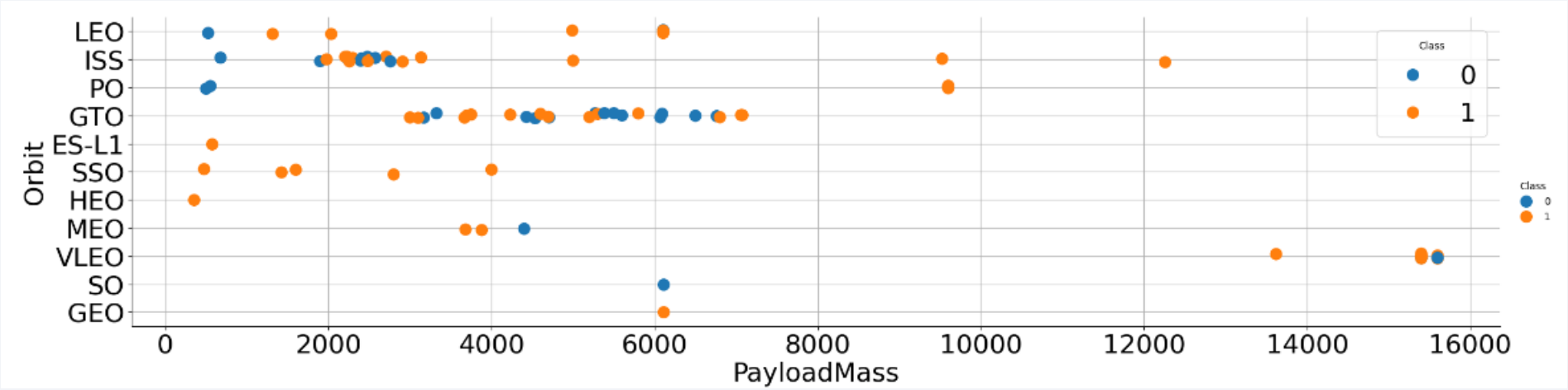


Flight Number vs. Orbit Type



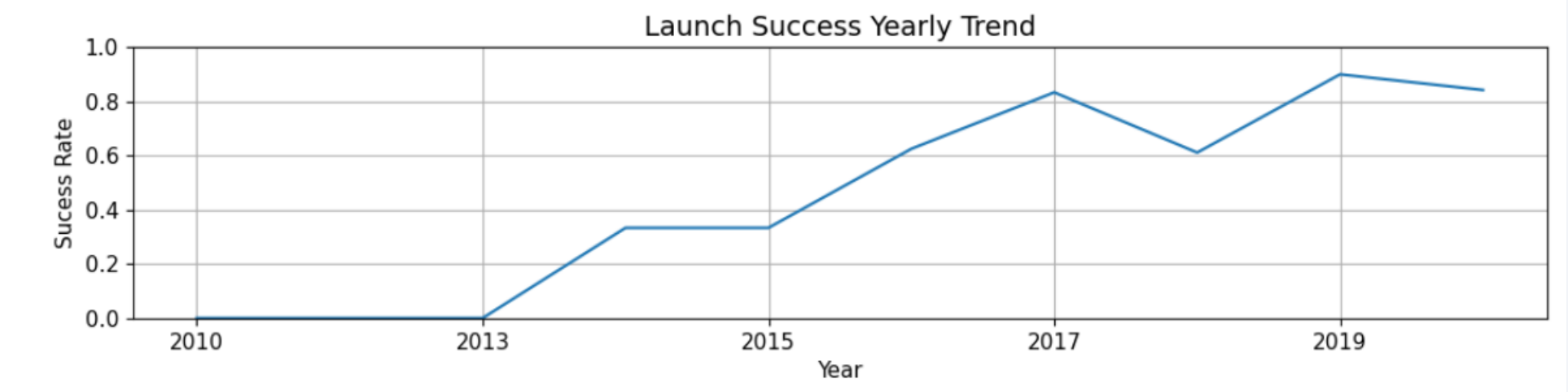
- As can be seen in the graph, most launches occur in GTO, PO, ISS and LEO orbits.
- There is a tendency for failure in the first launches of each orbit
- As the number of flights increases, successful launches increase.

Payload vs. Orbit Type



- As can be seen in the graph, most loads are up to 8000 kg
- From this graph, it cannot be concluded that there is a correlation between Payloadmass and launch success
- The VLEO orbit concentrated the highest Payloadmass values

Launch Success Yearly Trend



- As can be seen in the graph, the annual success rate clearly increases over the years, reflecting a learning curve in launches.

All Launch Site Names

- The names of all launch sites was obtained using a SELECT DISTINCT query.

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

```
[32]: %sql SELECT DISTINCT "LaunchH_Site" FROM SPACEXTABLE;
```

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

```
[32]: ,,,,,,,,,,,,,,,,,,,
```

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

Launch Site Names Begin with 'CCA'

- The 5 records where launch sites begin with `CCA` were obtained using a SELECT query plus a LIMIT keyword.

```
[50]: %sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 4;
 * sqlite:///my_data1.db
Done.
[50]: ''
```

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- The total payload carried by boosters from NASA was obtained using a SELECT query plus the SUM() function.

```
[36]: %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)';
 * sqlite:///my_data1.db
Done.
[36]: ,,,,,,,,,,,
SUM(PAYLOAD_MASS__KG_)
45596
```

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 was obtained using a SELECT query plus a AVG() funcition and a LIKE keyword.

```
[40]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1%';
       * sqlite:///my_data1.db
      Done.
[40]: ,,,,,,,,,,
      AVG(PAYLOAD_MASS__KG_)
      2534.6666666666665
```

First Successful Ground Landing Date

- The date of the first successful landing outcome on ground pad was obtained using a SELECT query plus a MIN(Date) function and WHERE.

```
[41]: %sql SELECT MIN(Date) FROM SPACEXTABLE WHERE "Mission_Outcome" = 'Success';
 * sqlite:///my_data1.db
Done.
[41]: ,,,,,,,,,,
```

MIN(Date)
2010-06-04

Successful Drone Ship Landing with Payload between 4000 and 6000

- The names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
[42]: %sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE "Mission_Outcome" = 'Success' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
 * sqlite:///my_data1.db
```

F9 v1.1 B1014	F9 FT B1032.2
F9 v1.1 B1016	F9 B4 B1040.2
F9 FT B1020	F9 B5 B1046.2
F9 FT B1022	F9 B5 B1047.2
F9 FT B1026	F9 B5 B1048.3
F9 FT B1030	F9 B5 B1051.2
F9 FT B1021.2	F9 B5B1060.1
F9 FT B1032.1	F9 B5 B1058.2
F9 B4 B1040.1	F9 B5B1062.1

Total Number of Successful and Failure Mission Outcomes

- The total number of successful and failure mission outcomes were obtained using two queries. The first get the distinct Mission_Outcome from the table, and the second count the numbers of success and unsucess using the COUNT() function and LIKE keyword.

```
[44]: %sql SELECT DISTINCT "Mission_Outcome" FROM SPACEXTABLE;
```

```
[46]: %sql SELECT COUNT(*) AS Success_Count, (SELECT COUNT(*) FROM SPACEXTABLE WHERE Mission_Outcome NOT LIKE 'Success%')
      AS Not_Success_Count FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Success%';
```

Success_Count	Not_Success_Count
100	1

Boosters Carried Maximum Payload

- The names of the booster which have carried the maximum payload mass were obtained using a nested SELECT query and WHERE keyword.
- Present your query result with a short explanation here

```
[49]: %sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
 * sqlite:///my_data1.db
```

Booster_Version	
F9 B5 B1048.4	F9 B5 B1049.5
F9 B5 B1049.4	F9 B5 B1060.2
F9 B5 B1051.3	F9 B5 B1058.3
F9 B5 B1056.4	F9 B5 B1051.6
F9 B5 B1048.5	F9 B5 B1060.3
F9 B5 B1051.4	F9 B5 B1049.7

2015 Launch Records

- The failed landing outcomes in drone ship for in year 2015 were obtained using a SELECT query plus a SUBSTR() / AS function

```
[54]: %sql SELECT SUBSTR(Date, 6, 2) AS Month, "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE SUBSTR(Date, 0, 5) = '2015' AND "Landing_Outcome" = 'Failure (drone ship)'
 * sqlite:///my_data1.db
```

Month	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

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

- The rank of landing outcomes between 2010-06-04 and 2017-03-20, in descending order, was obtained using a SELECT query plus COUNT() function and plus WHERE/BETWEEN/AND keywords.

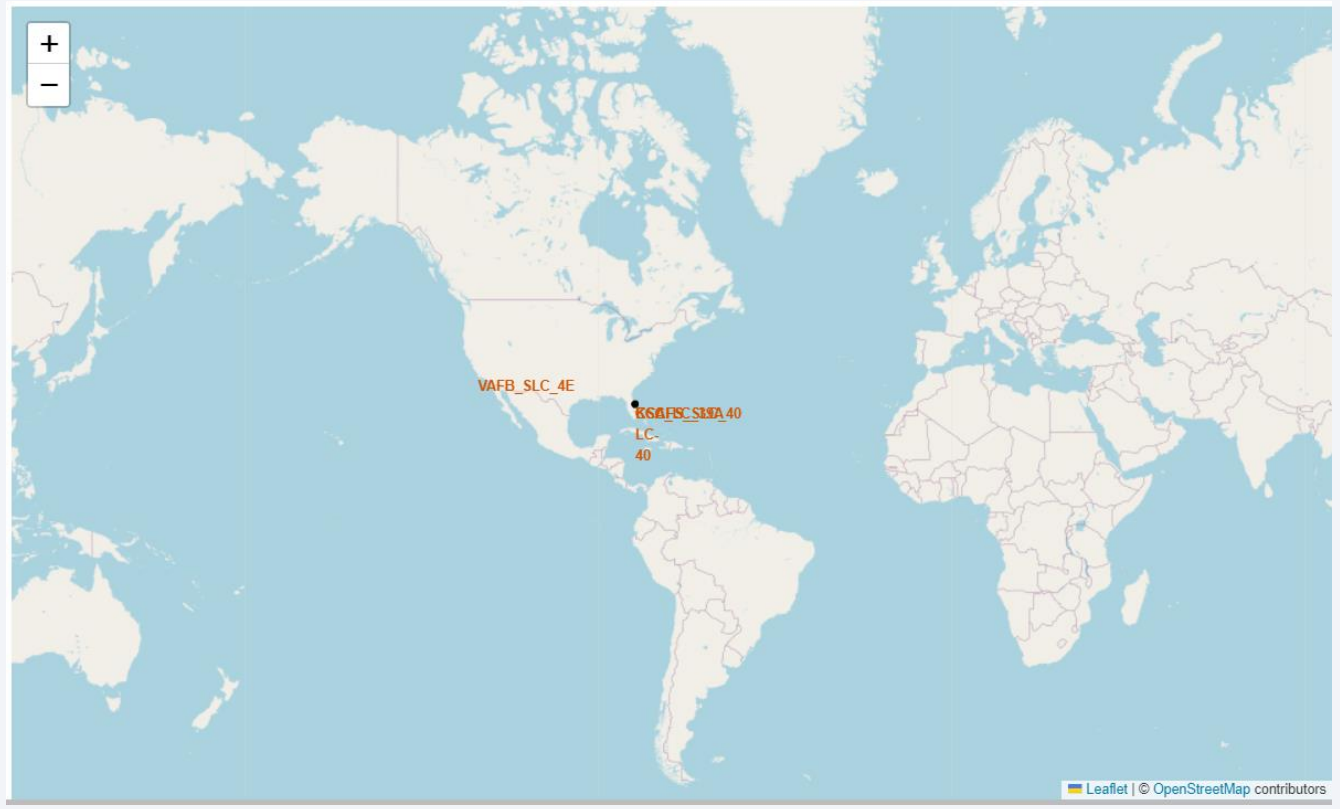
```
[55]: %sql SELECT Landing_Outcome, COUNT(*) AS Count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY Count DESC;
 * sqlite:///my_data1.db
```

Landing_Outcome	Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

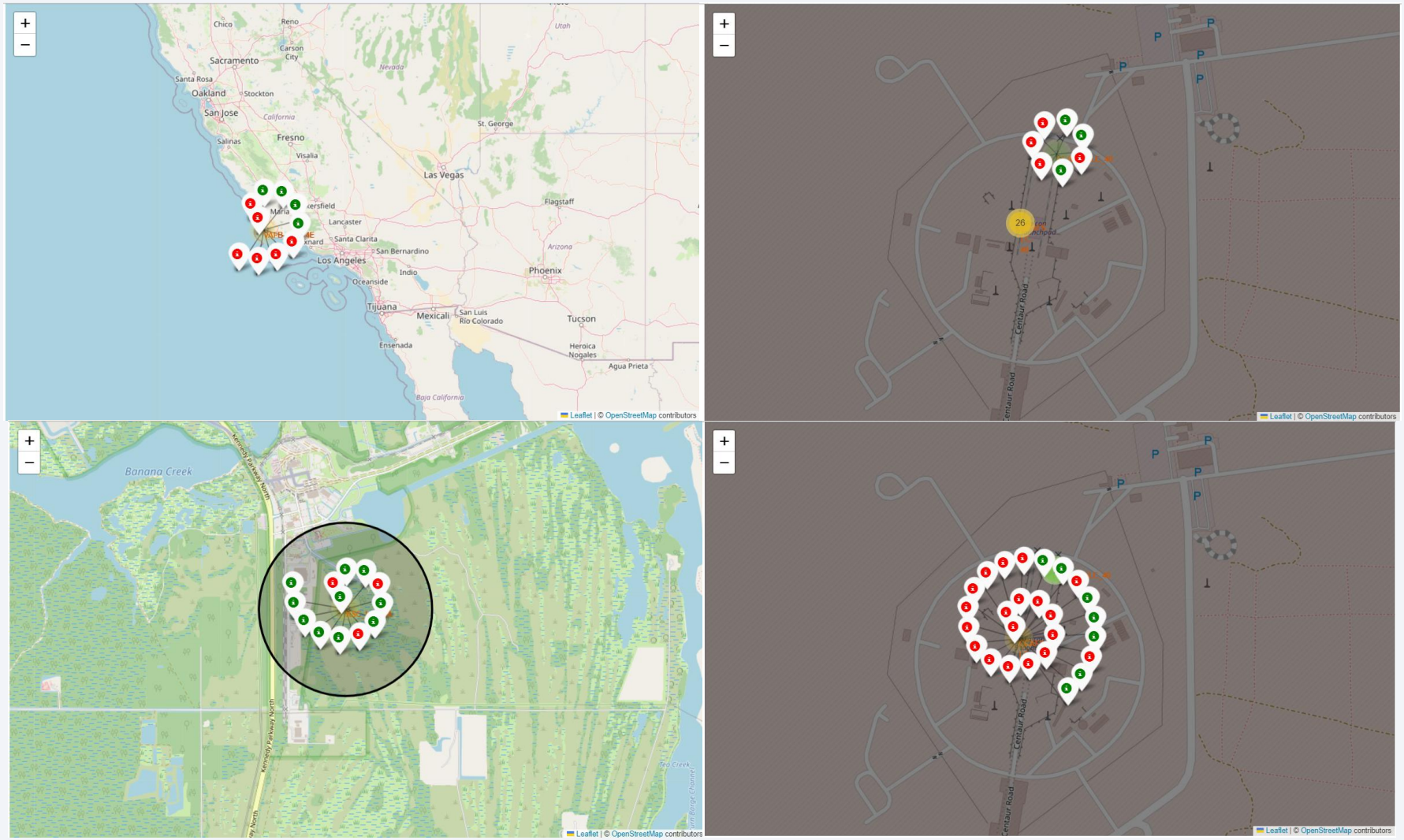
Launch Sites Proximities Analysis

Global map showing the position of launch sites



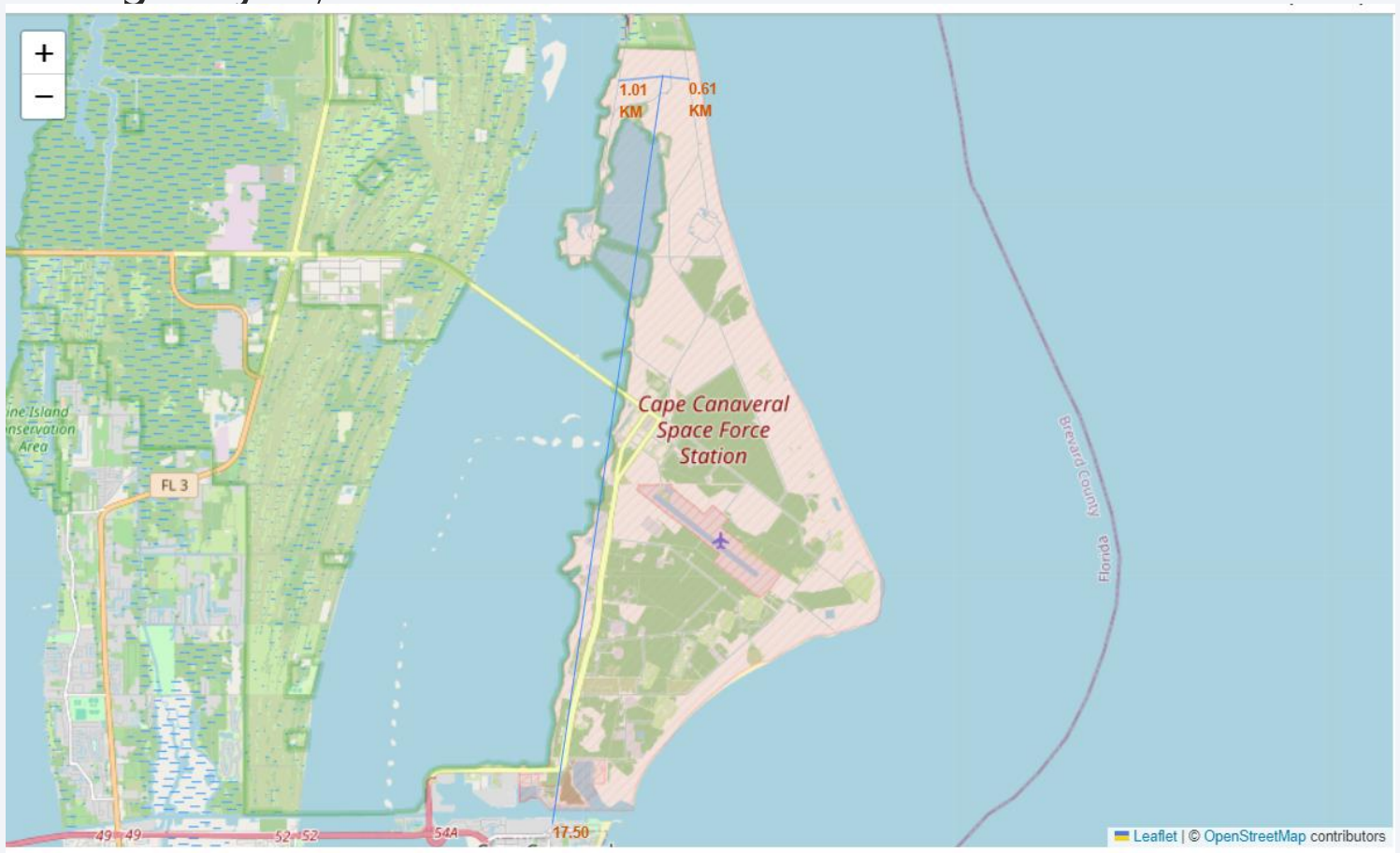
- As can be seen on the map, the launch sites are close to the Equator line.

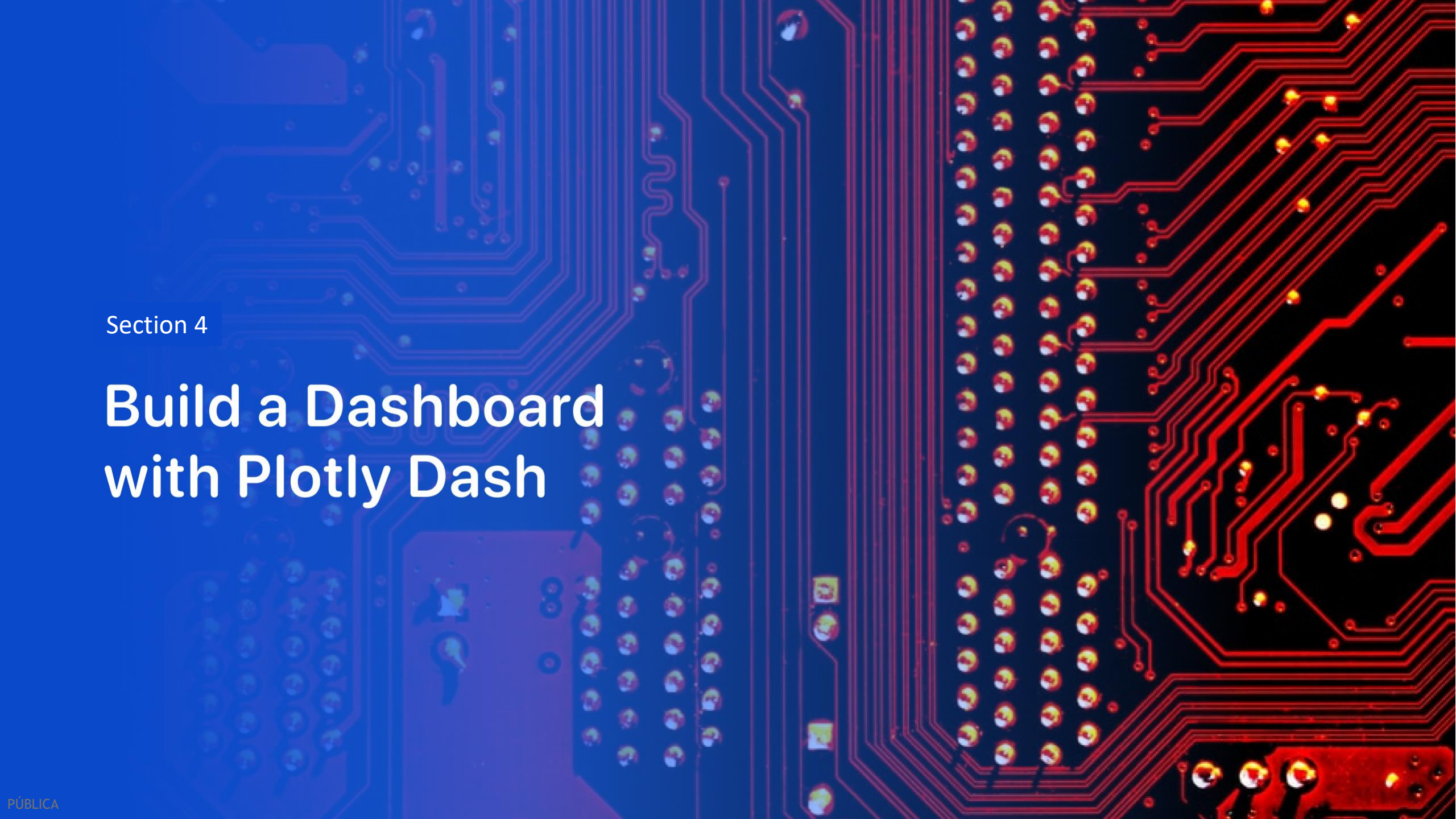
Launch outcomes on the map



Distances to the neighborhood of the CCAFS LC-40 Launch Site

- Distance to closest city (Cape Canaveral): 17,5 km
- Distance to closest railway: 0,61 km
- Distance to closest highway: 1,01 km

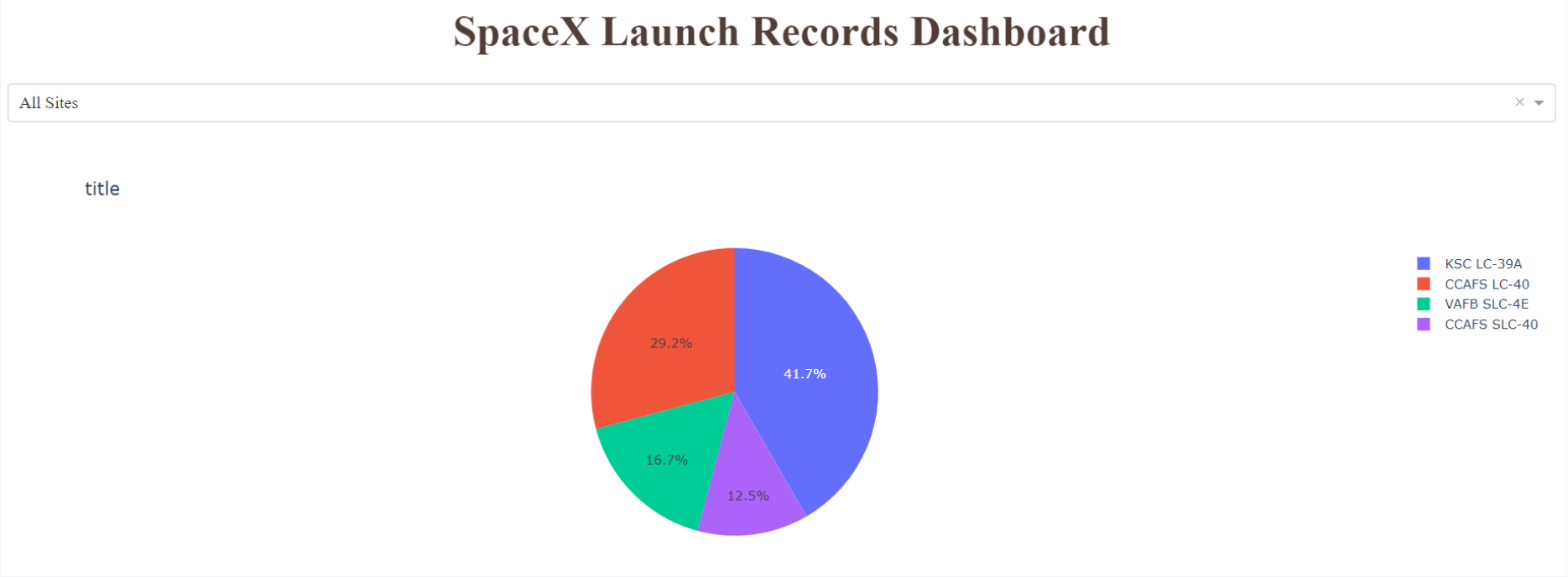




Section 4

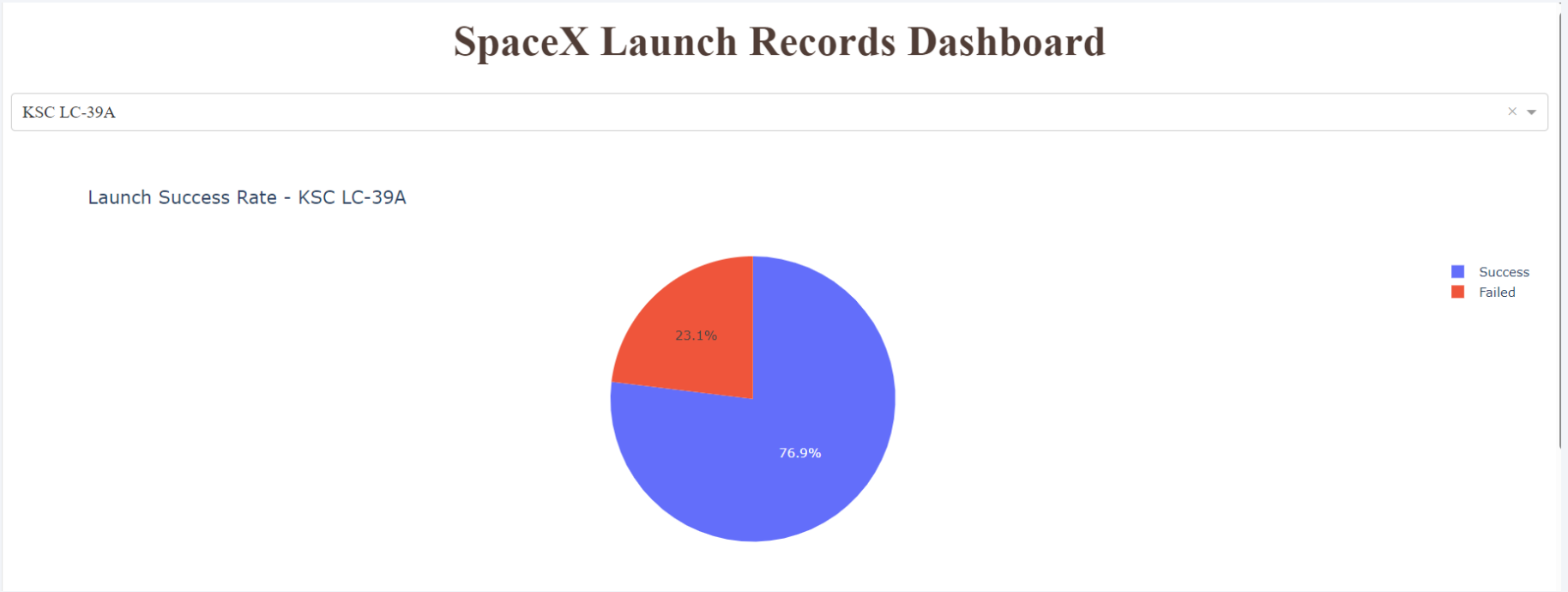
Build a Dashboard with Plotly Dash

Success Count



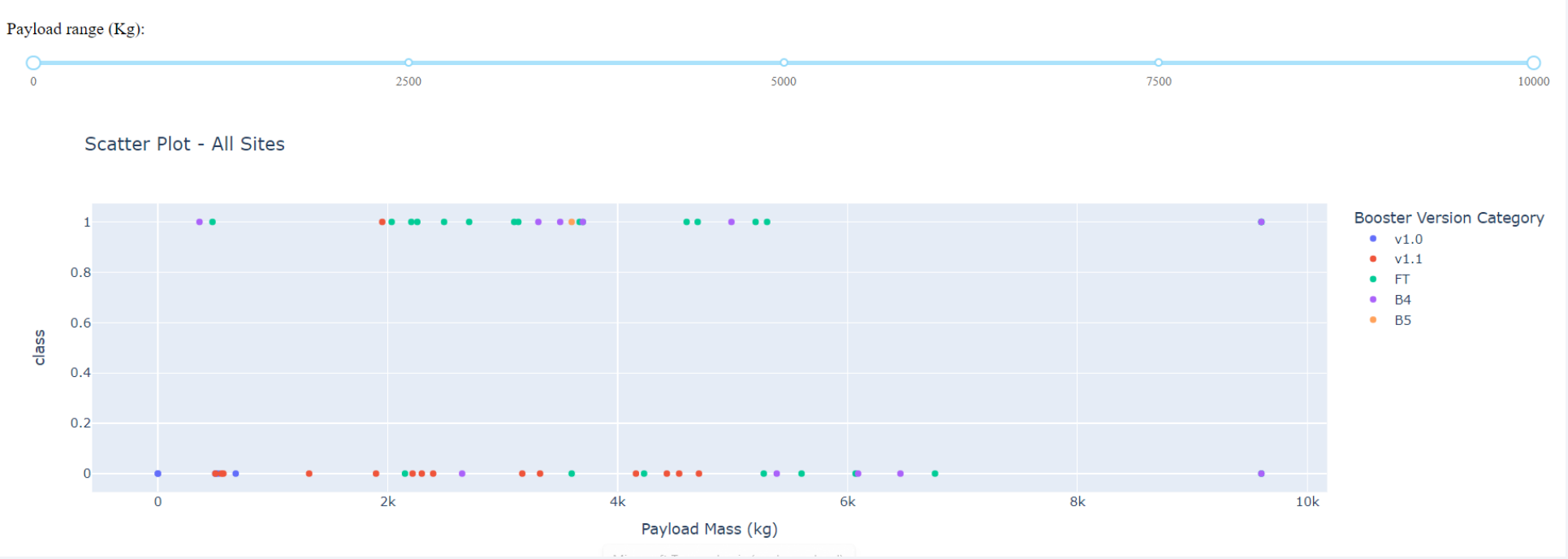
- The piechart shows four different launch sites.
- KSC LC-39A has the largest success count.
- CCAFS SLC-40 has the lowest success count.

Success and failure for KSC LC-39A



- The piechart shows the success and failed for KSC LC-39A site.
- This site has achieved a successful rate of 76.9%.

Payload vs. Launch Outcome



- The scatter plot shows Payload vs. Launch Outcome for all sites, with payload selected in the range slider
- Although it is not a guarantee of success, most launches that have resulted in success are in the payload mass range between 2000 and 4000 kg.
- FT booster version have the largest success rate
- v1.1 was only successful, although it launched across a wide range of payload masses.

Section 5

Predictive Analysis (Classification)

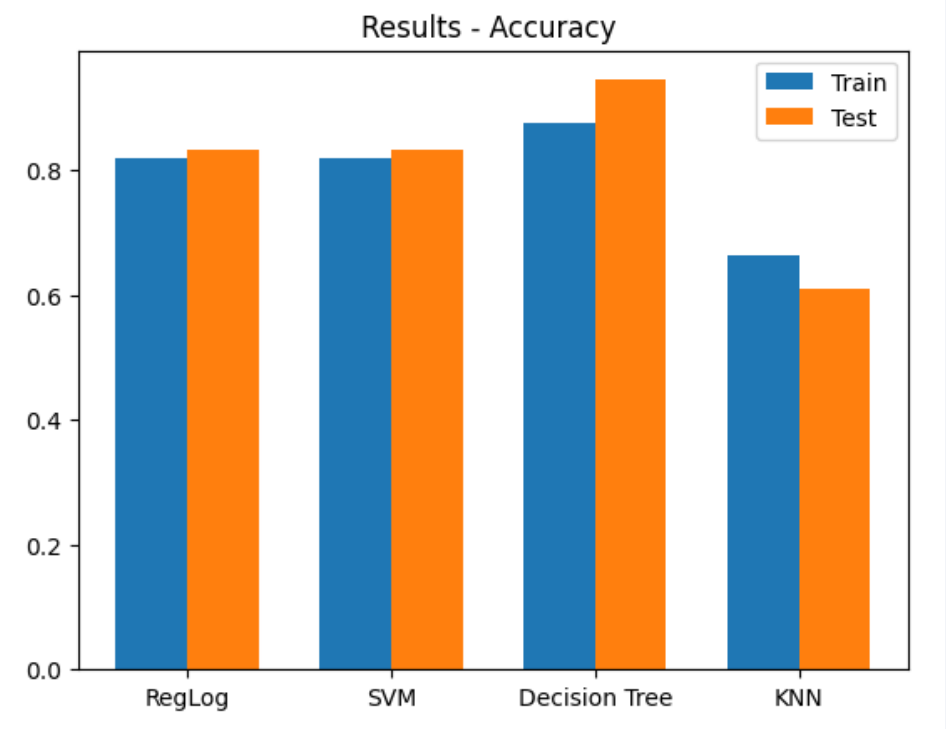
Classification Accuracy

- Accuracy for all built classification models, in a bar chart

```
[46]: results
```

[46]:

	Model	Train Acuracy	Test Acuracy
0	RegLog	0.819643	0.833333
1	SVM	0.819643	0.833333
2	Decision Tree	0.875000	0.944444
3	KNN	0.664286	0.611111

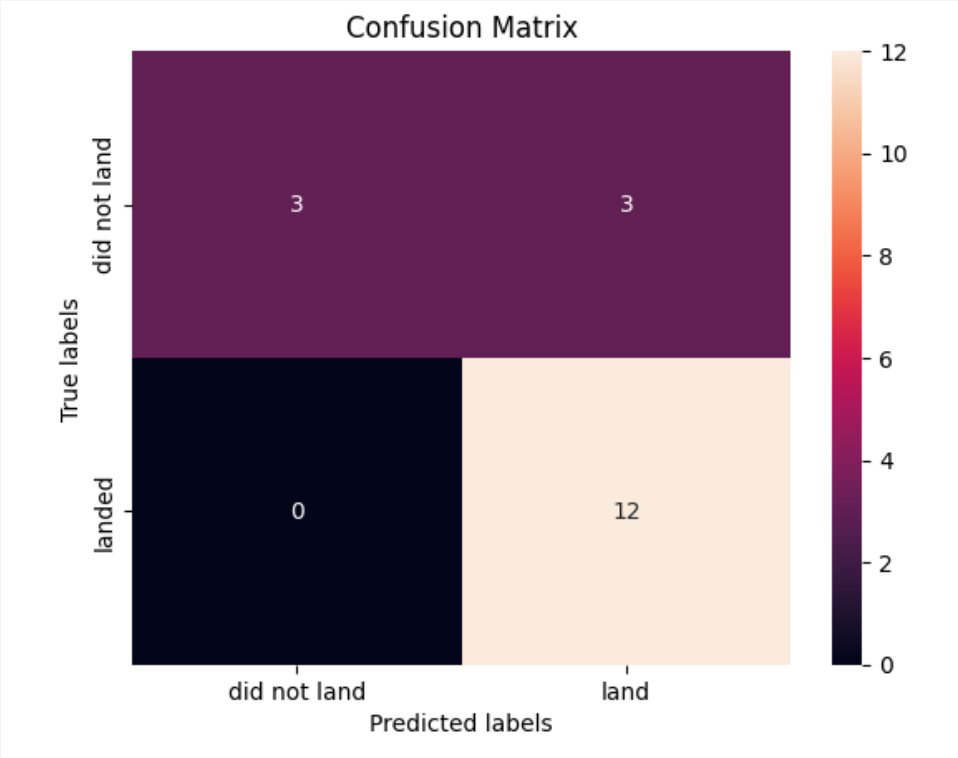


- The Decision Tree model presented the highest accuracy and was considered the best model.

Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/8e13fa89cc9baa3a954b574fac4906e1c4684ef4/W4%20-%20SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20B.ipynb

Confusion Matrix

- As can be seen in the confusion matrix of the decision **Logistic Regression e SVC Models**, there are **03 false positives**, that is, the model predicts success (land) when there was failure (did not land). Thus, the models were only able to correctly predict 50% of failure cases (did not land).

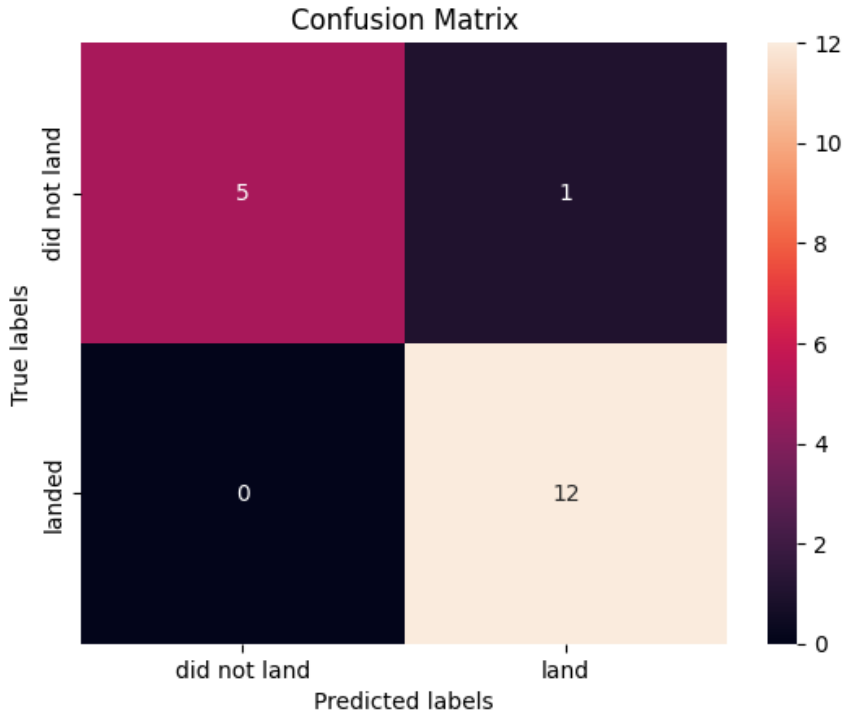


- On the other hand, for all successful cases (land), the models predicted correctly, i.e., **no false negatives**.

Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/8e13fa89cc9baa3a954b574fac4906e1c4684ef4/W4%20-%20SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20B.ipynb

Confusion Matrix

- As can be seen in the confusion matrix of the **Decision Tree** model, there is **01 false positives**, that is, the model predicts success (land) when there was failure (did not land). Thus, the model was able to correctly predict 83% of failure cases (did not land).



- On the other hand, for all successful cases (land), the model predicted correctly, i.e., **no false negatives**.

Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/8e13fa89cc9baa3a954b574fac4906e1c4684ef4/W4%20-%20SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20B.ipynb

Conclusions

- Except the KNN model, all models obtained good results in terms of accuracy.
- The Decision Tree Model predict correctly all successes launches and 83% of failures and achieved 94% of accuracy on test data.
- The number of launches increases, the number of successes increases.
- The annual success rate clearly increases over the years, reflecting a learning curve in launches.
- A higher payload mass indicates that the chance of success is greater.
- a high success rate was achieved for the ES-L1, GEO, HEO and SSO orbits.
- There is a tendency for failure in the first launches of each orbit.
- The launch sites are close to the Equator line.
- KSC LC-39A has the largest success count. This site has achieved a successful rate of 76.9%.
- FT booster version have the largest success rate.

Link:
https://github.com/fabiorodrigocs/IBMCapstone/blob/8e13fa89cc9baa3a954b574fac4906e1c4684ef4/W4%20-%20SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20B.ipynb

Thank you!