Winning Space Race with Data Science

Beata Wereszczyńska
27.06.2022

Outline

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

Executive Summary

Summary of methodologies

- Data Collection (API & Web Scraping)
- Data Wrangling
- Exploratory Data Analysis (SQL & Data Visualization)
- Interactive Visual Analytics (interactive map) with Folium
- Dashboard with Plotly Dash
- Machine Learning - Predictive Analysis (Classification)

Summary of all results

- Exploratory Data Analysis results
- Interactive analytics in screenshots
- Predictive analysis results

Introduction

- Project background and context

Lowering cost of launching space rockets is of interest. SpaceX advertises the Falcon 9 rocket launches (on its website) with a cost of 62 million dollars. Other providers' prices are above 165 million dollars per launch. Most of the savings comes from SpaceX reusing the first stage instruments. Therefore, determining if the first stage will land is crucial for determining the cost of a particular launch. The goal of the project is to create a machine learning model to predict from public data if the first stage will be reused by SpaceX in particular case.

- Important questions

1. What are the factors determining if the first stage will land successfully?
2. Is it possible to predict the landing outcome from publicly available data?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology

  Data was collected using SpaceX API and web scraping from Wikipedia.

- Data wrangling

  One hot encoding was applied to categorical features relevant.

- Exploratory data analysis (EDA) using visualization and SQL
- Interactive visual analytics using Folium and Plotly Dash
- Predictive analysis using classification models

  Building, tuning and evaluating machine learning classification models.

Data Collection

Step one - SpaceX API:

- Data collection was done using get request on the SpaceX API
- The response content was decoded as a Json format (using json() function) and transformed into pandas data frame (using json_normalize)
- The data was cleaned and checked for missing values (filled where necessary)

Step two - web scraping:

- Web scraping from Wikipedia was performed for Falcon 9 launch records (using BeautifulSoup)
- The launch records were extracted as HTML table, parsed and converted into pandas data frame for future analysis.

Data Collection – SpaceX API

The get request was used on the SpaceX API to collect data. The data was then cleaned and saved after basic data wrangling and formatting.

The notebook with the code is available on GitHub under this URL:

https://github.com/BeataWereszczynska/IBM_Coursera_Data_Science_Capstone/blob/main/Data%20Collection%20API%20notebook.ipynb

Getting Response from API

⬇

Converting Response to a .json file

⬇

Apply custom functions to clean data

⬇

Assign list to dictionary and then dataframe

⬇

Filter dataframe and export to .csv file

Data Collection - Scraping

Web scrapping was applied to webscrap Falcon 9 launch records using BeautifulSoup. The resulting table was parsed, converted into pandas dataframe and saved as .csv file.

The notebook with the code is available on GitHub under this URL:

https://github.com/BeataWereszczynska/IBM_Coursera_Data_Science_Capstone/blob/main/Web%20Scraping%20notebook.ipynb

Getting Response from HTML

⬇

Creating BeautifulSoup Object

⬇

Finding tables and getting column

⬇

names

⬇

Creation of dictionary

⬇

Converting dictionary to dataframe

Saving the dataframe as .CSV

Data Wrangling

In the data set analyzed, there are several different categories where the booster did or did not land successfully. Those outcomes were converted into **Training Labels** with **1** (the booster successfully landed) and **0** (the landing was unsuccessful).

The **number of launches** at each site was calculated as well as the **number and occurrence of each orbit**.

The **number** and **occurrence** of **mission outcome per orbit type** was calculated.

Landing outcome label created

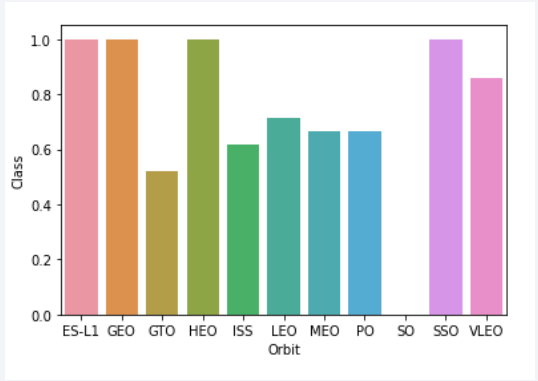
Success rates calculated.

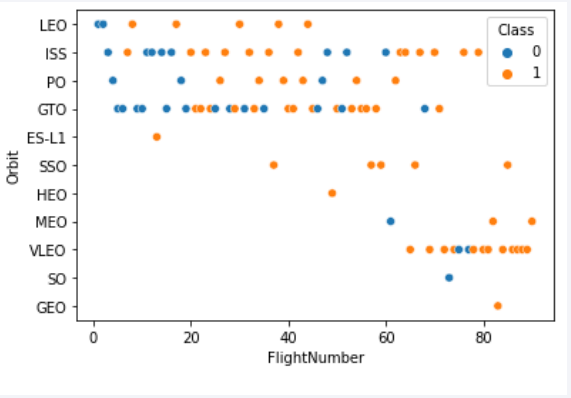
The notebook with the code is available on GitHub under this URL:

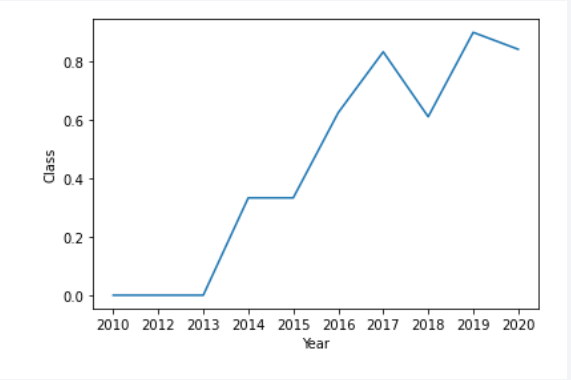
https://github.com/BeataWereszczynska/IBM_Coursera_Data_Science_Capstone/blob/main/Data%20Wrangling%20notebook.ipynb

EDA with Data Visualization

1. Scatter plots for variables correlation visualization:
   - Flight Number vs. Payload Mass
   - Flight Number vs. Launch Site
   - Launch Site vs. Payload Mass
   - Flight Number vs. Orbit
   - Payload Mass vs. Orbit
2. Bar chart comparison of the success rate of each orbit.
3. Line chart for visualization of success rate evolution in time.







The notebook with the code is available on GitHub under this URL:

https://github.com/BeataWereszczynska/IBM_Coursera_Data_Science_Capstone/blob/main/EDA%20with%20Visualization%20notebook.ipynb

EDA with SQL

SQL queries were used to get insight from the data:

- **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 acheived**
- **List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000**
- **Listing the total number of successful and failure mission outcomes**
- **List the names of the booster versions which have carried the maximum payload mass (using 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.**

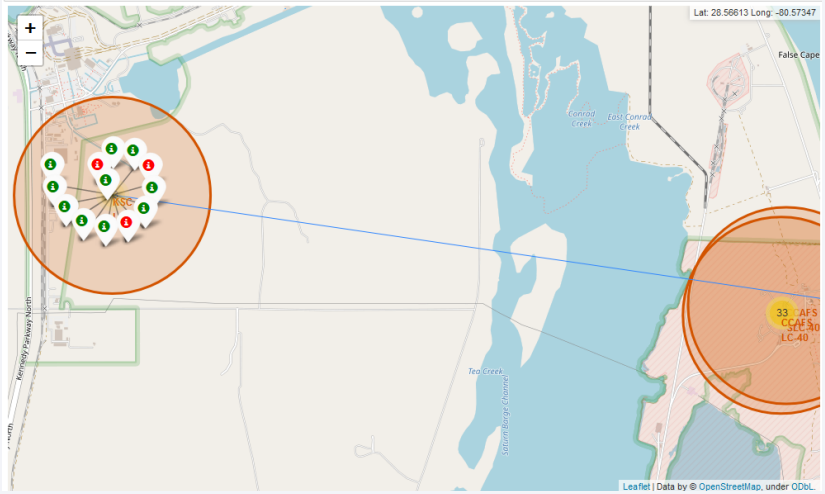
The notebook with the code is available on GitHub under this URL:

https://github.com/BeataWereszczynska/IBM_Coursera_Data_Science_Capstone/blob/main/EDA%20with%20SQL%20notebook%20net.ipynb

Interactive Map with Folium

To visualize the Launch Data into an interactive map, the latitude and longitude coordinates at each launch site were added as a circle marker around each launch site with a label of the name of the launch site.
Launch outcomes (failures or successes) were assigned to classes 0 and 1 with Green and Red markers on the map.
The distances from the launch sites to various landmarks were calculated to uncover trends. Lines are drawn on the map to measure distances to landmarks.

The notebook with the code is available on GitHub under this URL:
https://github.com/BeataWereszczynska/IBM_Coursera_Data_Science_Capstone/blob/main/Interactive%20Visual%20Analytics%20with%20Folium%20notebook.ipynb



Build a Dashboard with Plotly Dash

Interactive dashboard with Plotly dash was built to provide interactive insight into the launches data:

- pie charts showing the total launches and success statistics of launches by sites (to be picked by user interactively);
- scatter plot showing the Payload Mass (Kg) vs. outcome of the launch for the different booster versions (also interactively changing for launch site or total).

The notebook with the code is available on GitHub under this URL:

https://github.com/BeataWereszczynska/IBM_Coursera_Data_Science_Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

The data was loaded using numpy and pandas, transformed, split our into training and testing datasets.

Different machine learning models were built and tuned with different hyper parameters using GridSearchCV.

Accuracy was used as the metric for the model, which was improved using feature engineering and algorithm tuning.

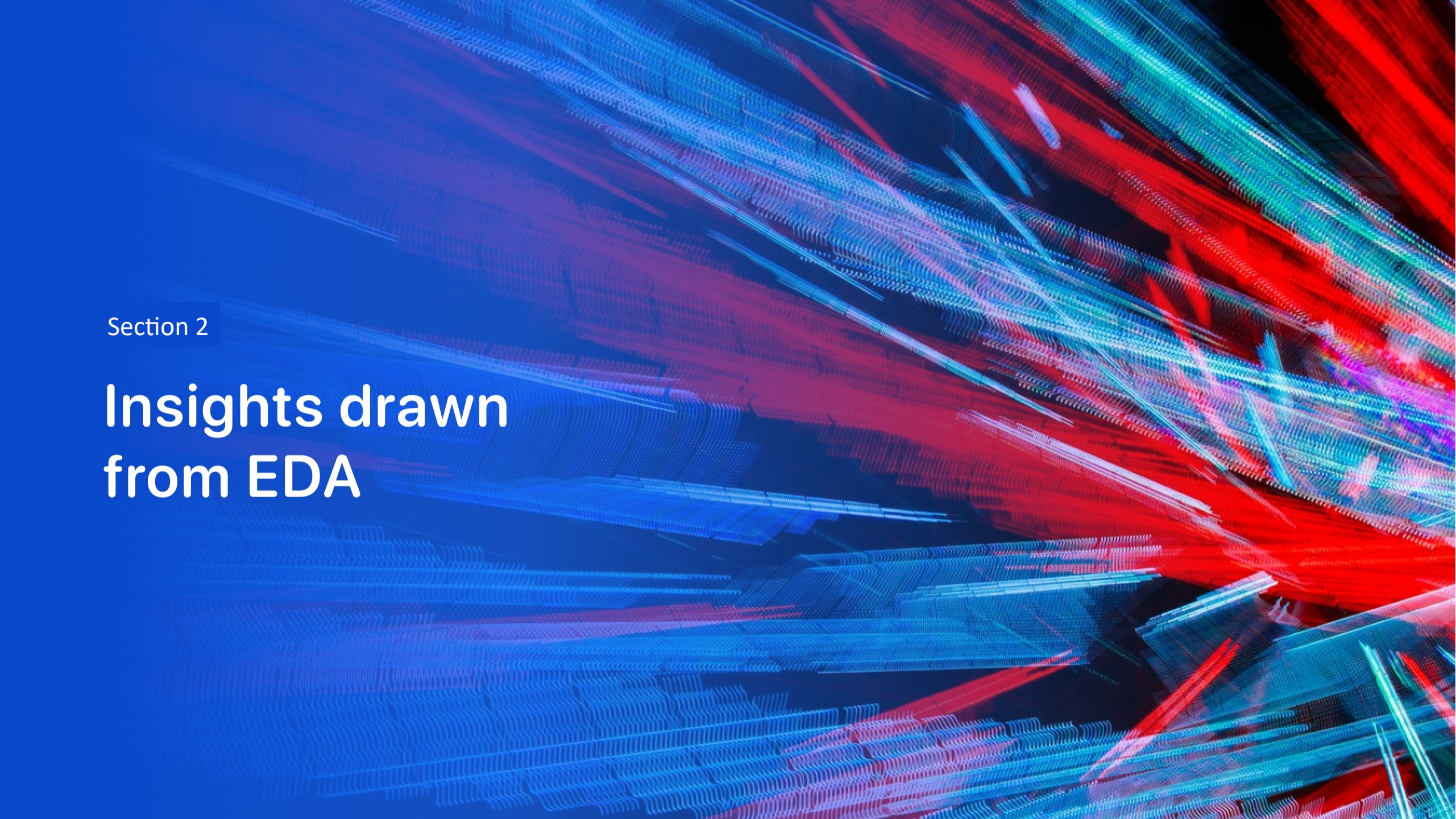
The best performing classification model was found.

The notebook with the code is available on GitHub under this URL:

https://github.com/BeataWereszczynska/IBM_Coursera_Data_Science_Capstone/blob/main/Machine%20Learning%20Prediction%20notebook.ipynb

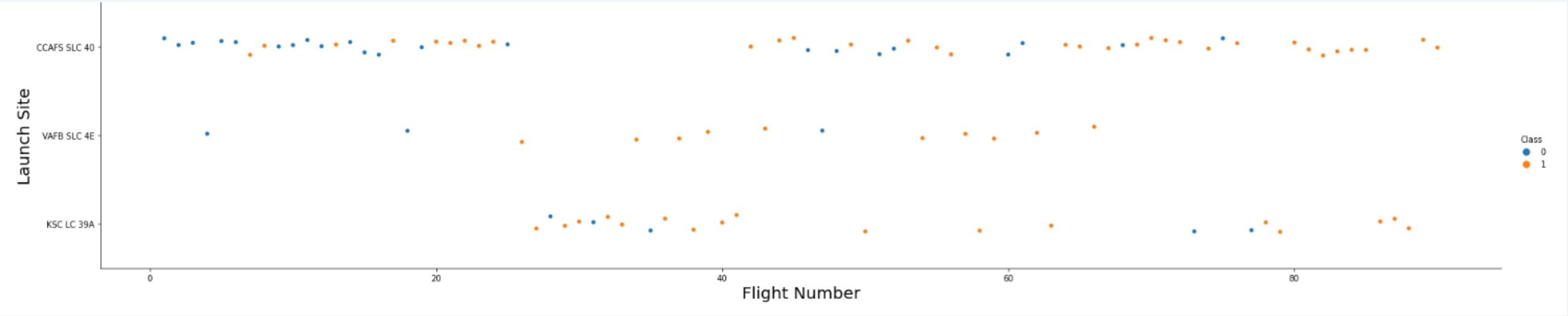
Results

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


Section 2
Insights drawn from EDA

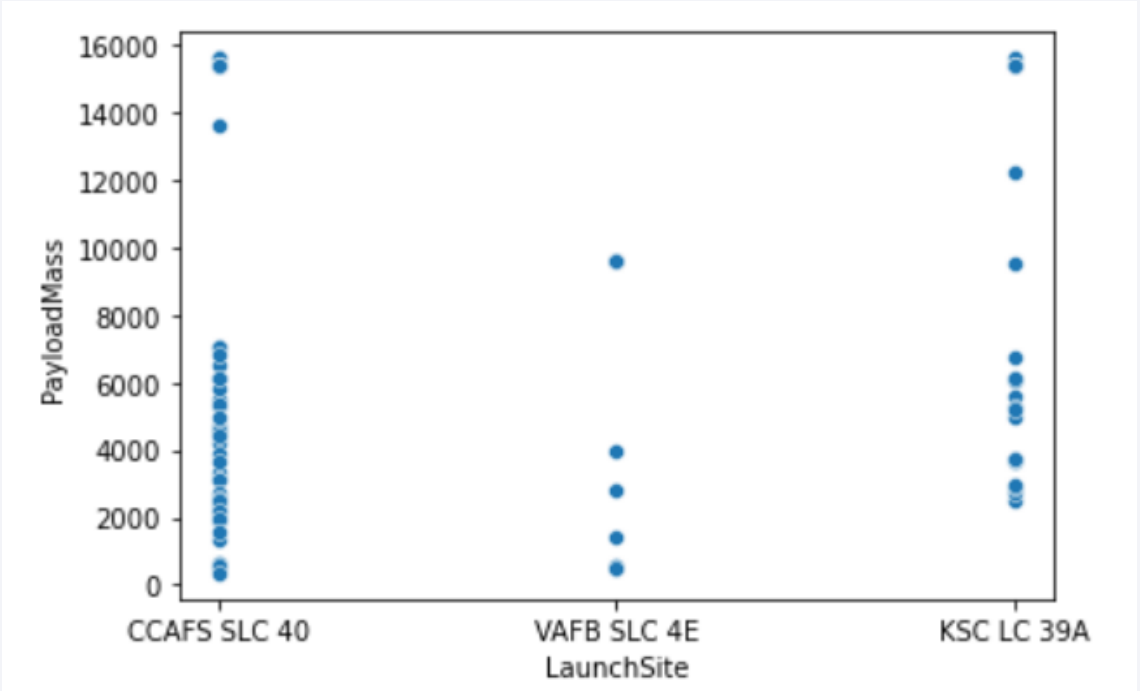
Flight Number vs. Launch Site

From the plot it can be concluded that, after initial difficulties period, successful launches outnumber unsuccessful launches for each of launch sites.



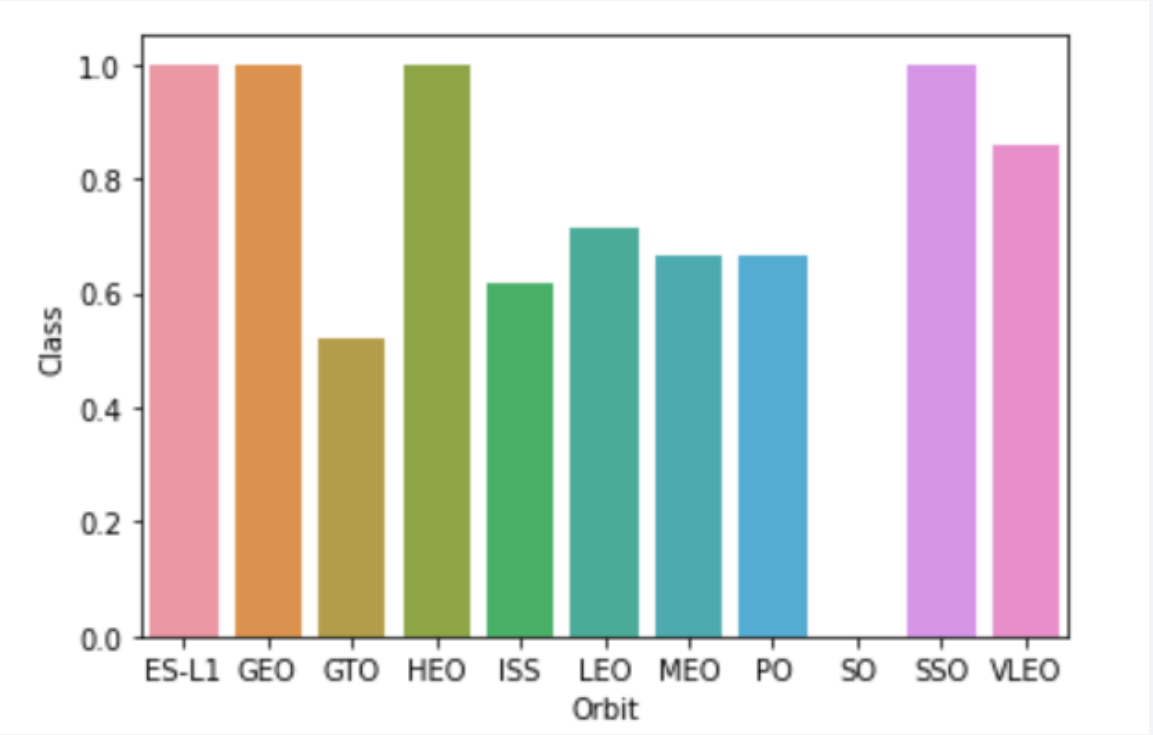
Payload vs. Launch Site

- Certain launch sites are preferred for certain payloads (e.g. no light flights launch from KSC LC 39A and no heavy flights launch from VAFB SLC 4E)



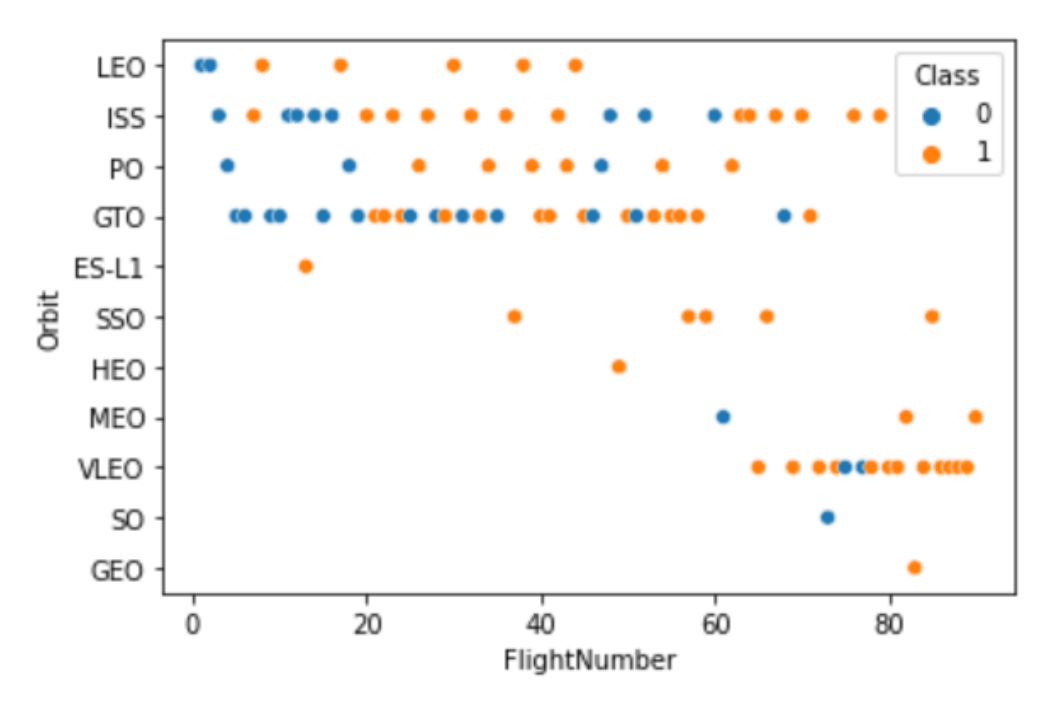
Success Rate vs. Orbit Type

- Orbits ES-L1, GEO, HEO and SSO have best success rate.



Flight Number vs. Orbit Type

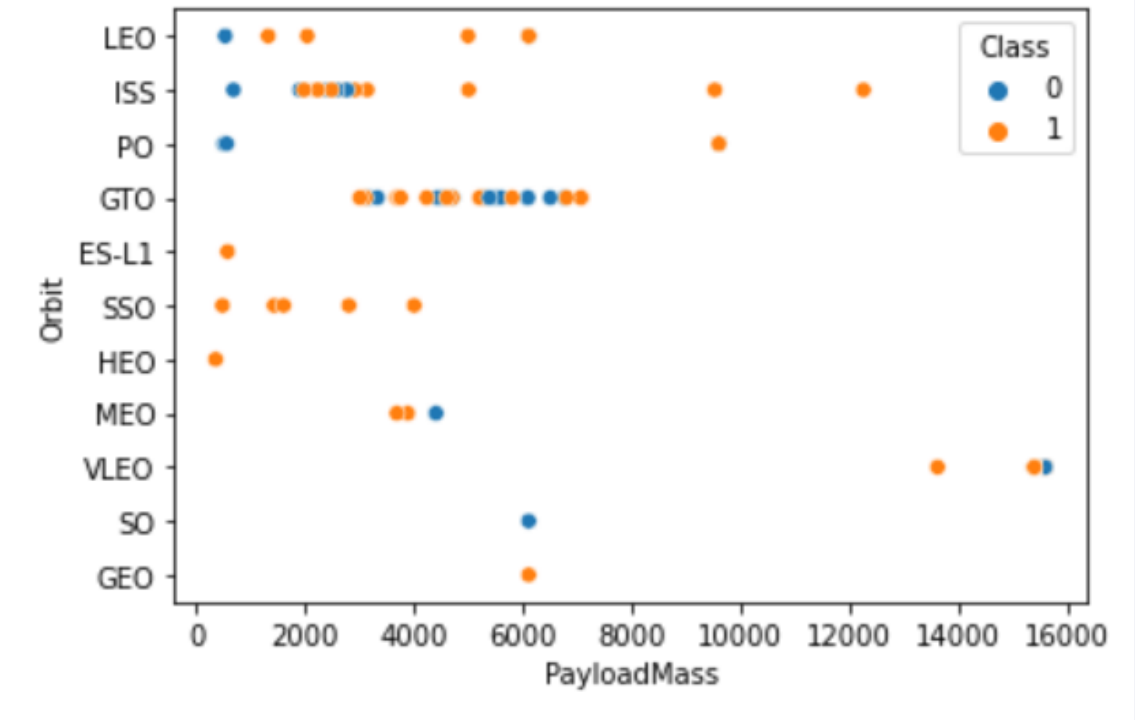
In the LEO orbit the Success appears related to the number of flights but it is not the rule for other orbits.



Payload vs. Orbit Type

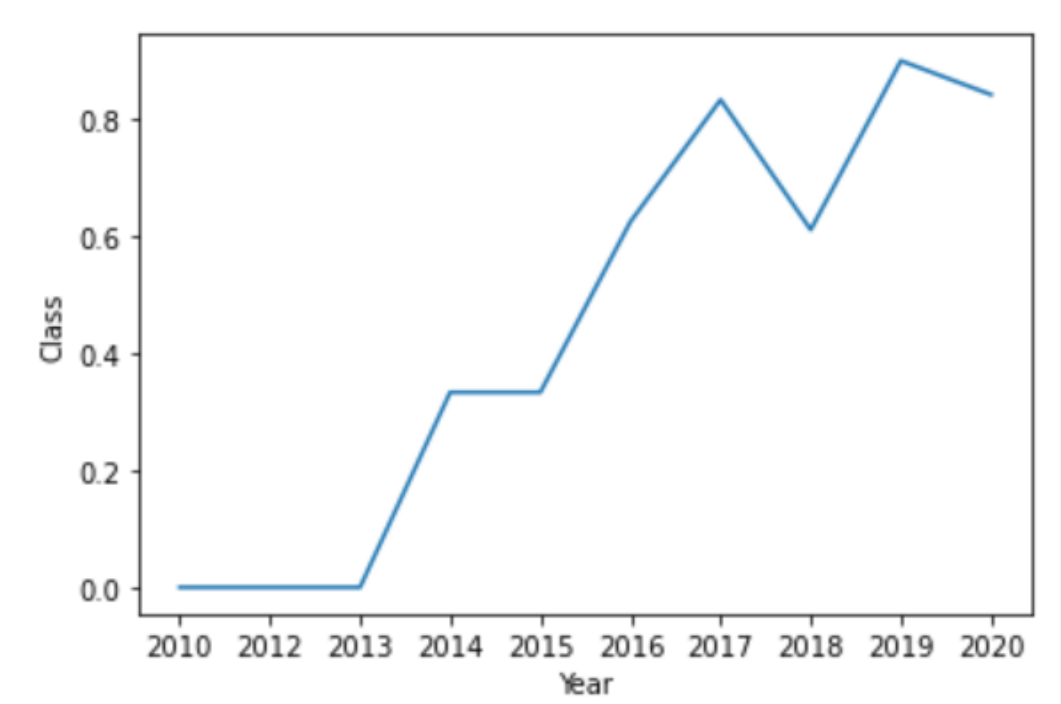
With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

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

Generally the success rate since 2013 kept increasing till 2020 (that is end of the data analyzed).



All Launch Site Names

The key word DISTINCT was used to show only unique launch sites’names from the SpaceX data.

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

In [6]:
```
%sql SELECT DISTINCT launch_site FROM SPACEXDATASET1
```

```
 * db2://tzs64899:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB
Done.
```

Out[6]:

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

Launch Site Names Begin with 'CCA'

In the query phrase *WHERE launch_site LIKE 'CCA%'* allowed to pick launch sites with name starting with *CCA*, while *limit 5* limited results to 5 lines.

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

In [7]:
```
%sql SELECT * FROM SPACEXDATASET1 WHERE launch_site LIKE 'CCA%' limit 5
```

```
 * db2://tzs64899:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB
Done.
```

Out[7]:

DATE	Time (UTC)	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

The total payload carried by boosters from NASA was calculated as 45596 kg using the query below.

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

```
In [8]: %sql SELECT SUM(payload_mass__kg_) FROM SPACEXDATASET1 WHERE CUSTOMER='NASA (CRS)'
```

```
 * db2://tzs64899:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB
Done.
```

Out[8]:

1
45596

Average Payload Mass by F9 v1.1

The average payload mass carried by booster version F9 v1.1 was calculated as 2928.4 kg using the query below.

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

```
In [9]: %sql SELECT AVG(payload_mass__kg_) FROM SPACEXDATASET1 WHERE booster_version='F9 v1.1'
```

```
 * db2://tzs64899:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB
Done.
```

Out[9]:

1
2928

First Successful Ground Landing Date

The date of the first successful landing outcome on ground pad was 22nd December 2015.

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

Hint:Use min function

In [50]:
```
a = %sql SELECT DATE FROM SPACEXDATASET1 WHERE "Landing_Outcome"='Success (ground pad)'
from datetime import datetime
import pandas as pd
a = pd.DataFrame(a)[0].tolist()
dates_list = [datetime.strptime(x, '%d-%m-%Y').date() for x in a]
min(dates_list)
```

```
 * db2://tzs64899:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB
Done.
```

Out[50]:
```
datetime.date(2015, 12, 22)
```

Successful Drone Ship Landing with Payload between 4000 and 6000

WHERE clause was used to filter for boosters which have successfully landed on drone ship and AND condition was applied to determine successful landing with payload mass greater than 4000 but less than 6000. The resulting list is visible below.

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

In [52]:
```
%sql select booster_version FROM SPACEXDATASET1 WHERE "Landing_Outcome"='Success (drone ship)' AND payload_mass__kg_>4000 AND pay
```

```
 * db2://tzs64899:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB
Done.
```

Out[52]:

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

Total Number of Successful and Failure Mission Outcomes

LIKE 'Success%' was used to filter for WHERE mission outcome was a success or not. There were 100 successes and 1 failure.

List the total number of successful and failure mission outcomes

In [58]:
```
a = %sql SELECT COUNT(*) FROM SPACEXDATASET1 WHERE mission_outcome LIKE 'Success%'
b = %sql SELECT COUNT(*) FROM SPACEXDATASET1 WHERE mission_outcome NOT LIKE 'Success%'
print("Successes:", a, "\n Failures:", b)
```

```
 * db2://tzs64899:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB
Done.
 * db2://tzs64899:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB
Done.
Successes: +-----+
|  1  |
+-----+
| 100 |
+-----+
 Failures: +---+
| 1 |
+---+
| 1 |
+---+
```

Boosters Carried Maximum Payload

Boosters that have carried the maximum payload were listed (below) using a subquery in the *WHERE* clause and *MAX()* function.

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

In [59]:
```
%sql select booster_version FROM SPACEXDATASET1 WHERE payload_mass__kg_=(SELECT MAX(payload_mass__kg_) FROM SPACEXDATASET1)
```

```
 * db2://tzs64899:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB
Done.
```

Out[59]:

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

2015 Launch Records

The below query containing WHERE clause, AND & LIKE was used to retrieve table of booster version and launch site name for failed landing outcomes in drone ship in year 2015.

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

In [62]:
```
%sql select booster_version, launch_site FROM SPACEXDATASET1 WHERE "Landing_Outcome"='Failure (drone ship)' AND DATE LIKE '%2015'
```

```
 * db2://tzs64899:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB
Done.
```

Out[62]:

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

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

The count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) were ranked between the date 2010-06-04 and 2017-03-20, in descending order. The code is specific due to some issues with uploading dates to DB2.

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

In [89]:
```
all_dates_data = %sql SELECT "Landing_Outcome", DATE FROM SPACEXDATASET1
all_dates_data = pd.DataFrame(all_dates_data)
all_dates_data[1]= [datetime.strptime(x, '%d-%m-%Y').date() for x in all_dates_data[1]]

# wszystko przez to, że IBM DB2 nie umiało wczytać dat w tym formacie jako dat
from datetime import timedelta, date
def daterange(date1, date2):
    for n in range(int ((date2 - date1).days)+1):
        yield date1 + timedelta(n)

data = all_dates_data[[x in daterange(date(2010, 6, 4), date(2017, 3, 20)) for x in all_dates_data[1].tolist()]]

data[0].value_counts()
```

```
 * db2://tzs64899:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/BLUDB
Done.
```

Out[89]:

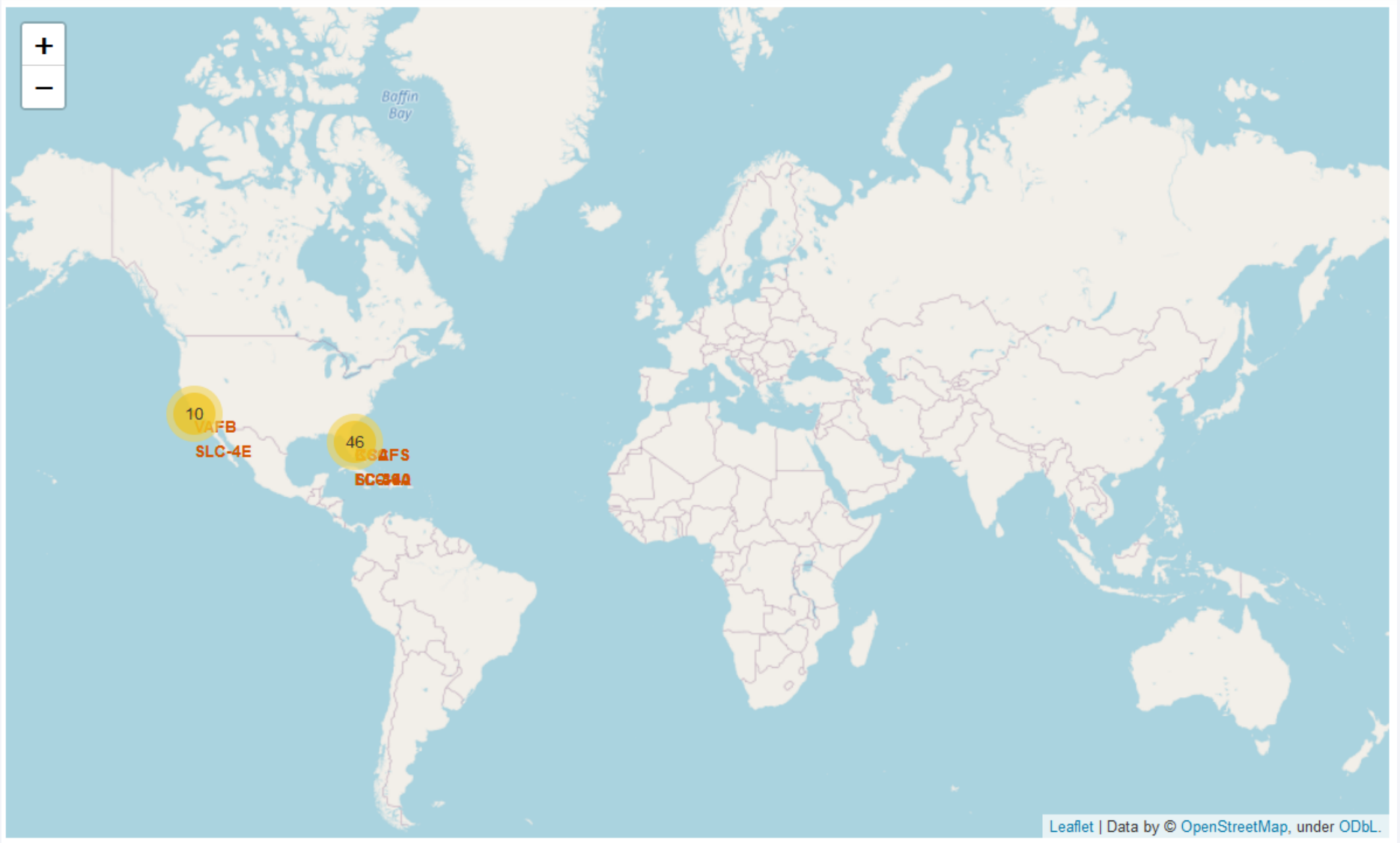
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Name: 0, dtype: int64

Section 3

Launch Sites Proximities Analysis

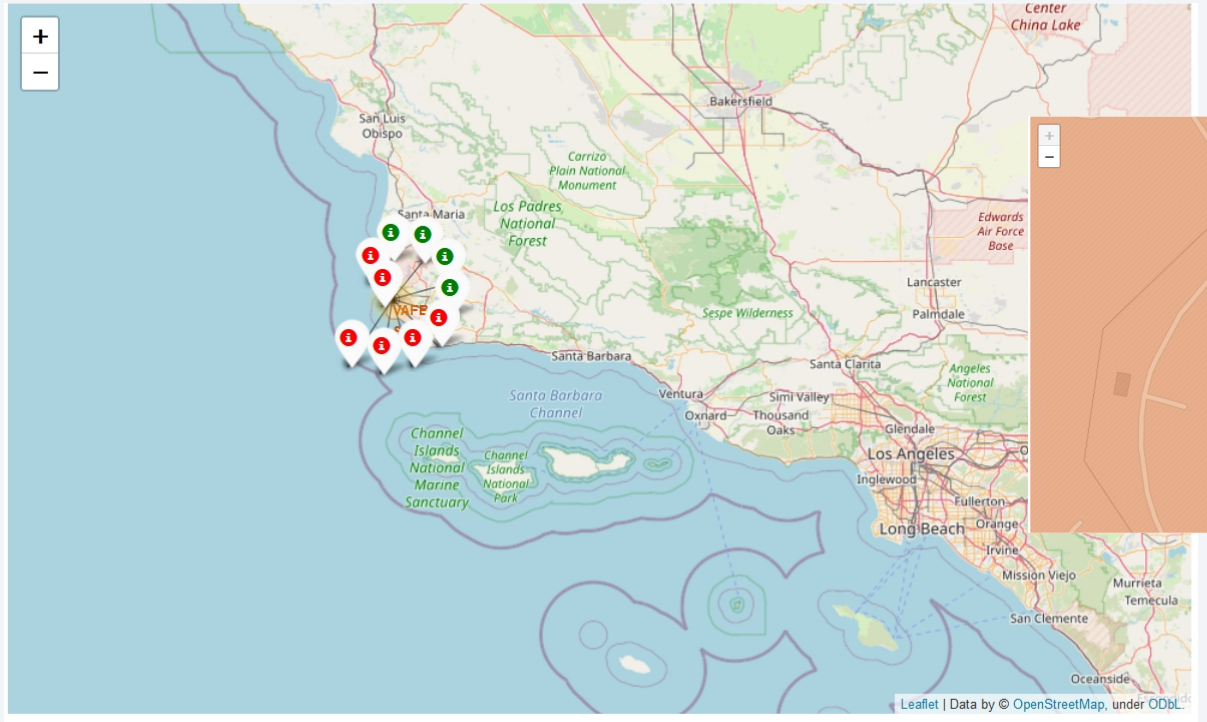
All launch sites - global map

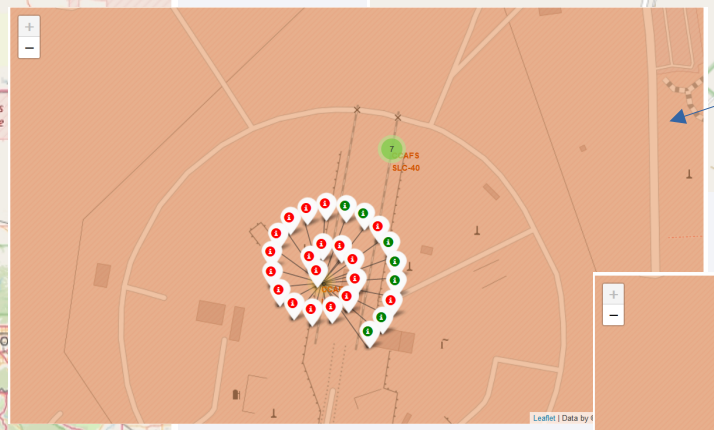


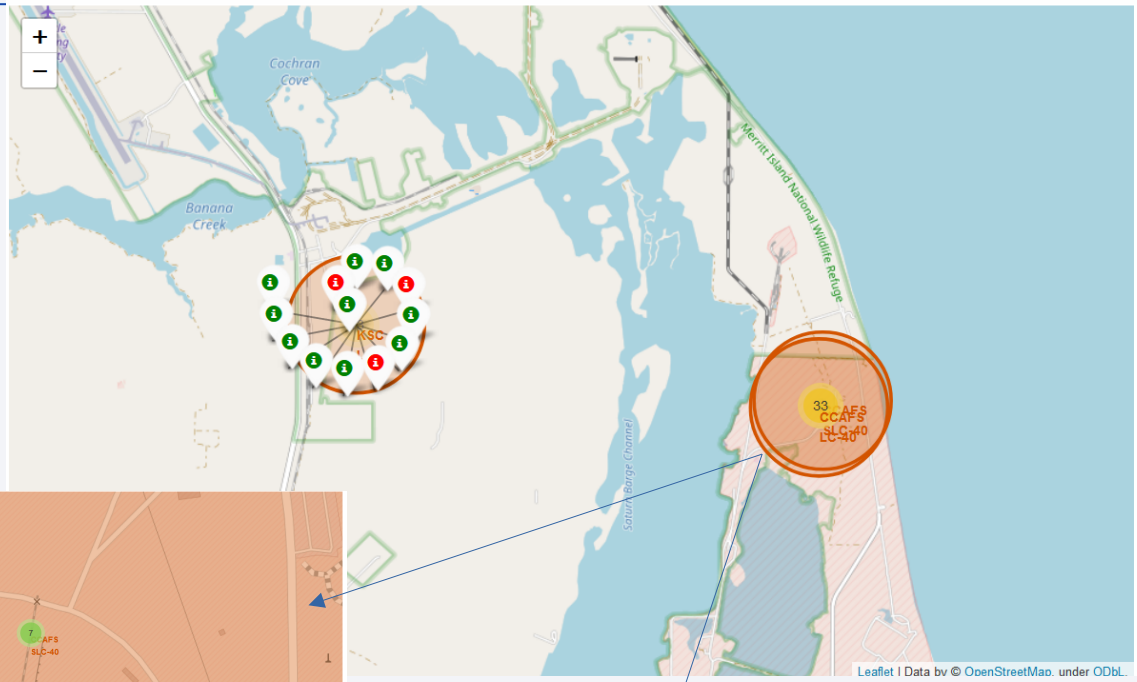
All SpaceX launch sites are on the coasts or the USA.

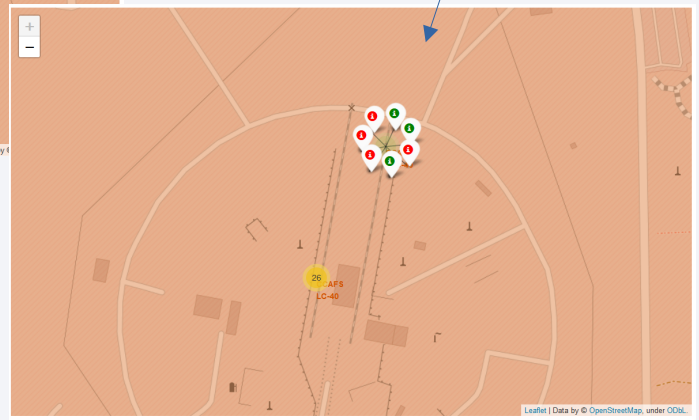
Color-labeled launch outcomes on the map

Green Markers show successful launches and Red Markers show failures. Visibly the most successful launch site from presented is CCAFS SLC-40.



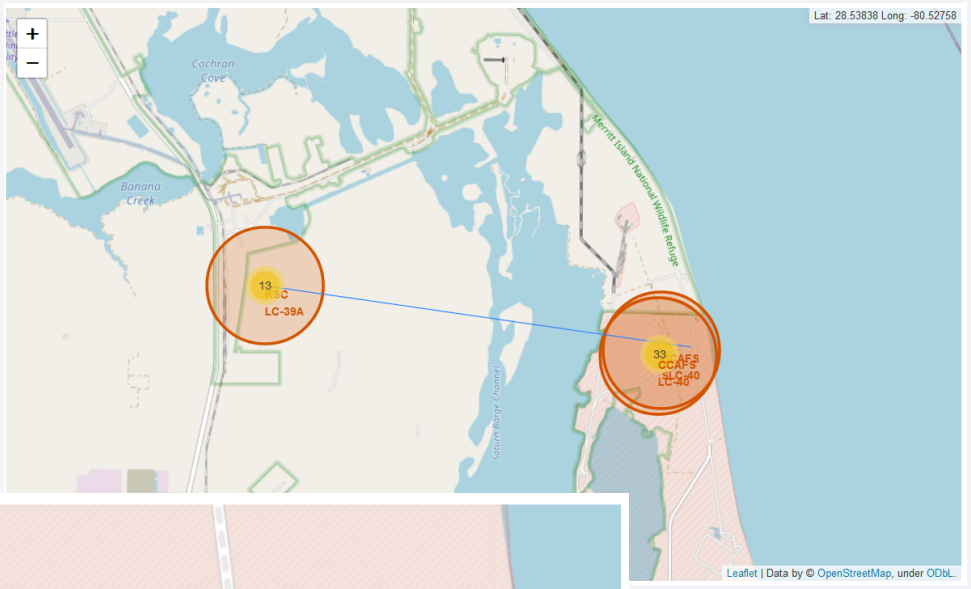


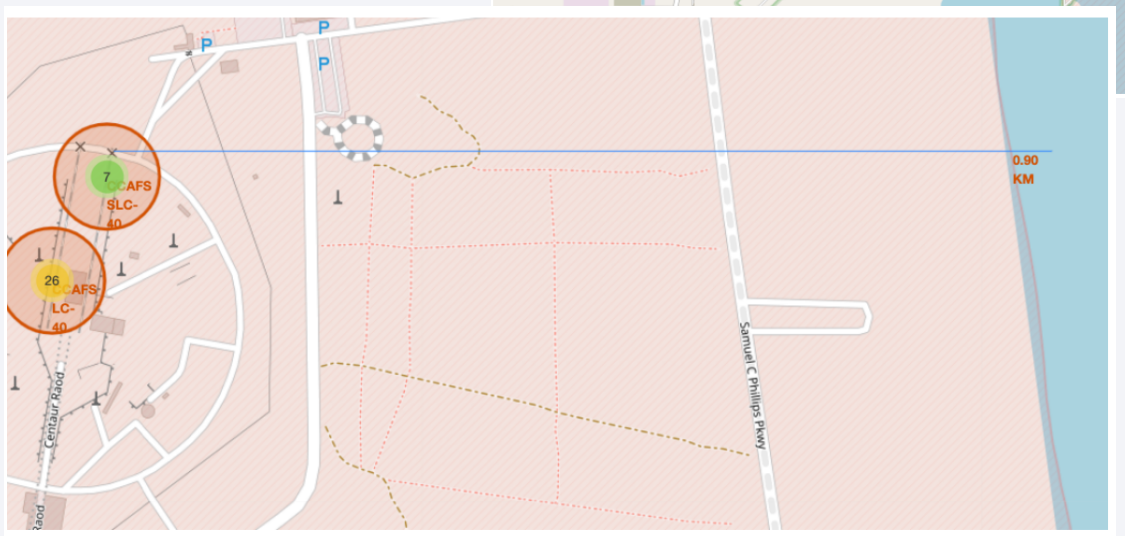


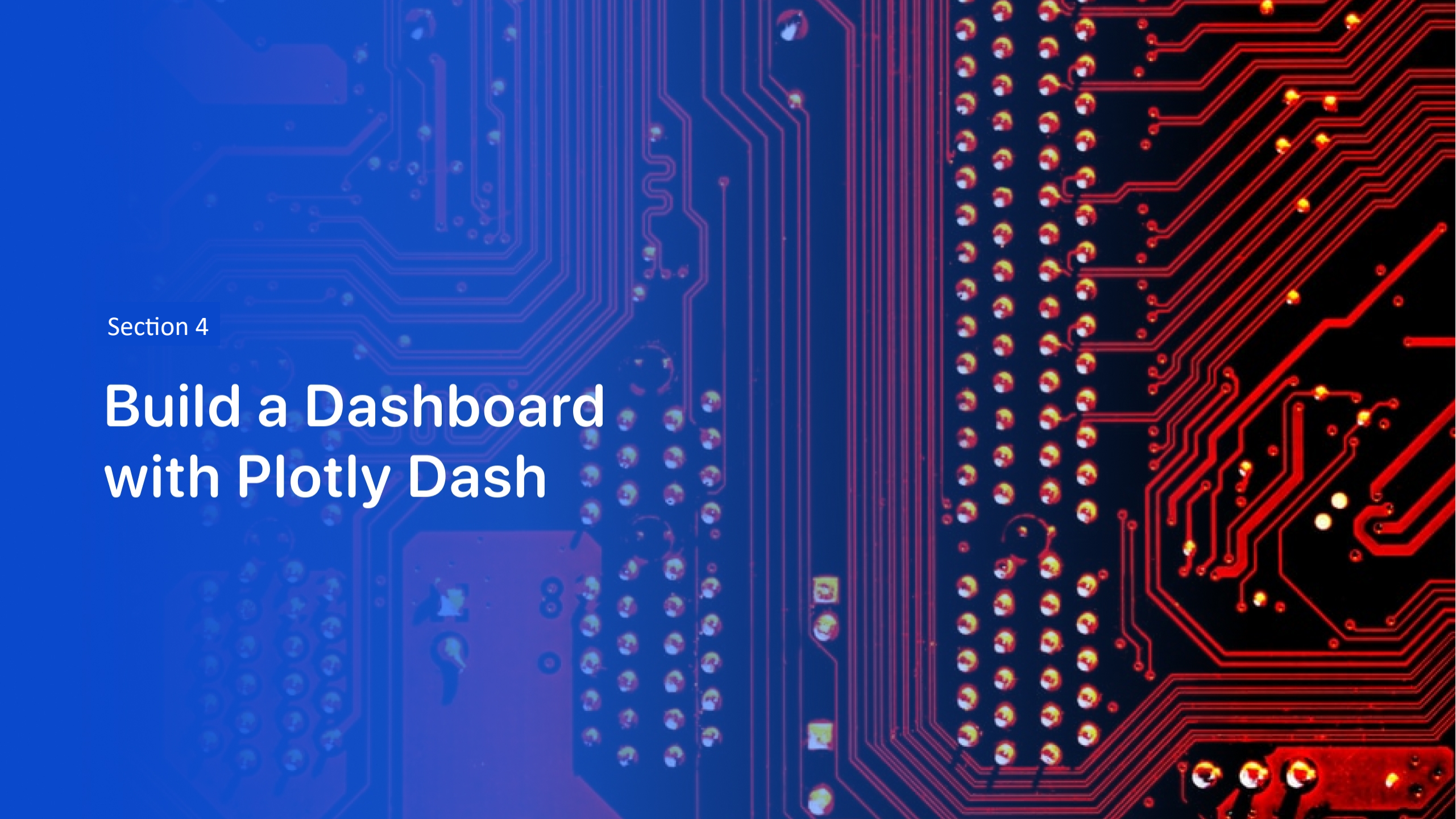


Importance of proximity to coast

All lunch sites are away from cities as expected. Proximity to railways or highway are not a rule but being in close distance to coast is (probably due to safety). Nevertheless the most successful site is not the closest to the sea.

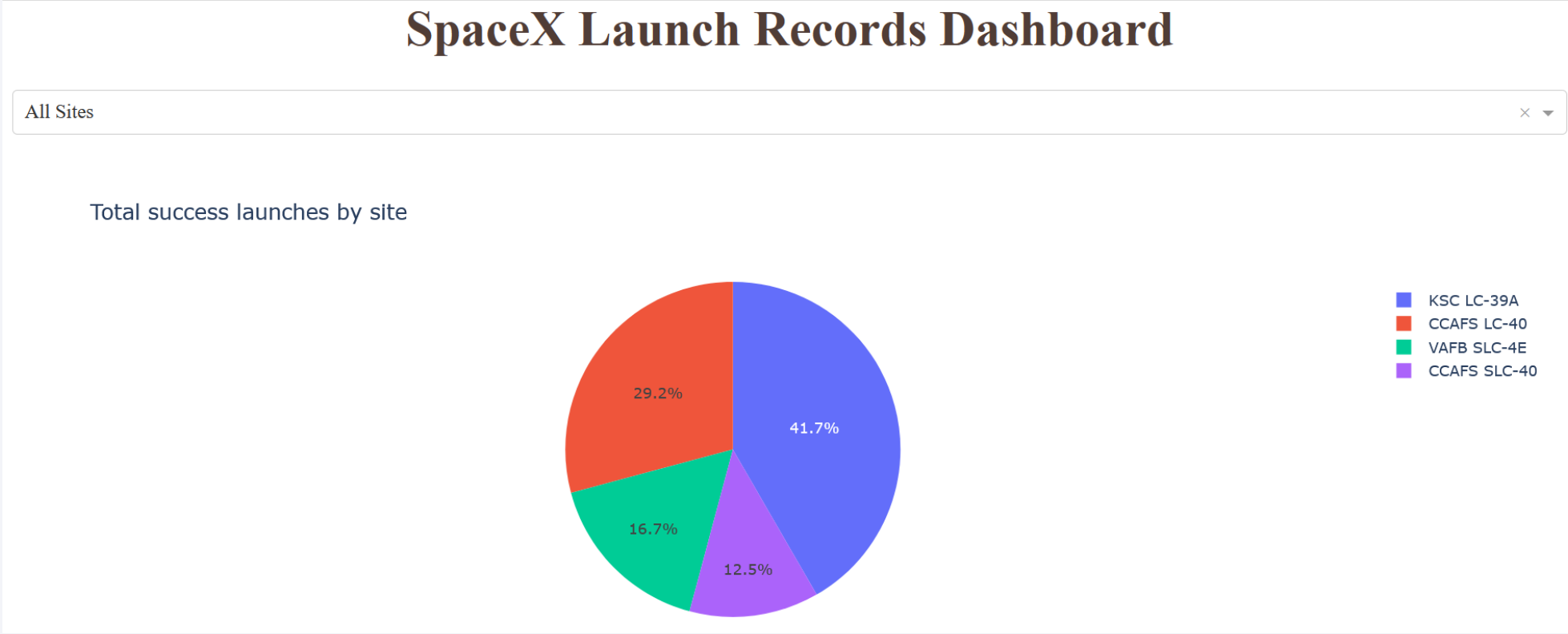





Section 4
Build a Dashboard with Plotly Dash

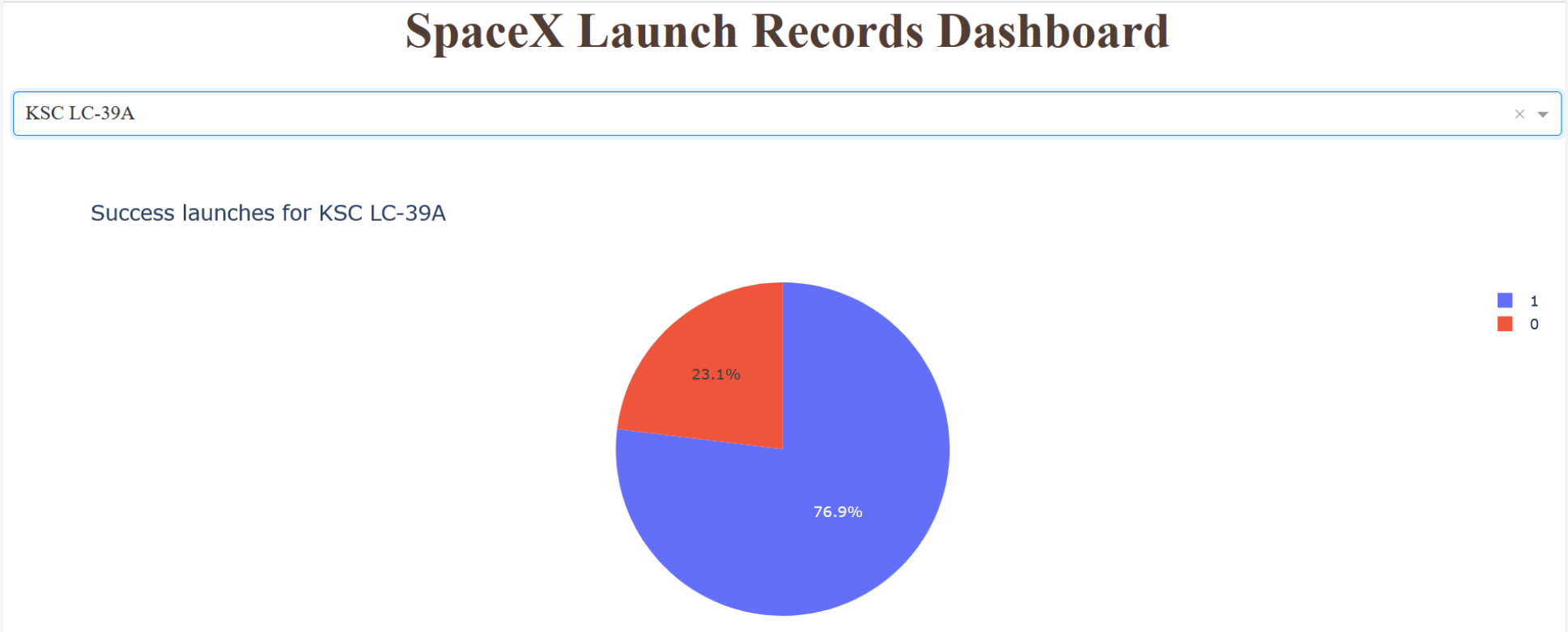
Launch success count for all sites

The highest success count falls to KSC LC-39A, CCAFS LC-40 is next.



KSC LC-39A success ratio

76.9% of launches from KSC LC-39A was successful.



Payload vs. Launch Outcome (for all sites and boosters)

For low weighted payload (0 – 3000 kg) success rate was 38%.

For medium weighted payload (3001 – 6000 kg) success rate was 50%.

For heavy weighted payload (6001 – 10000 kg) success rate was 7,6%.

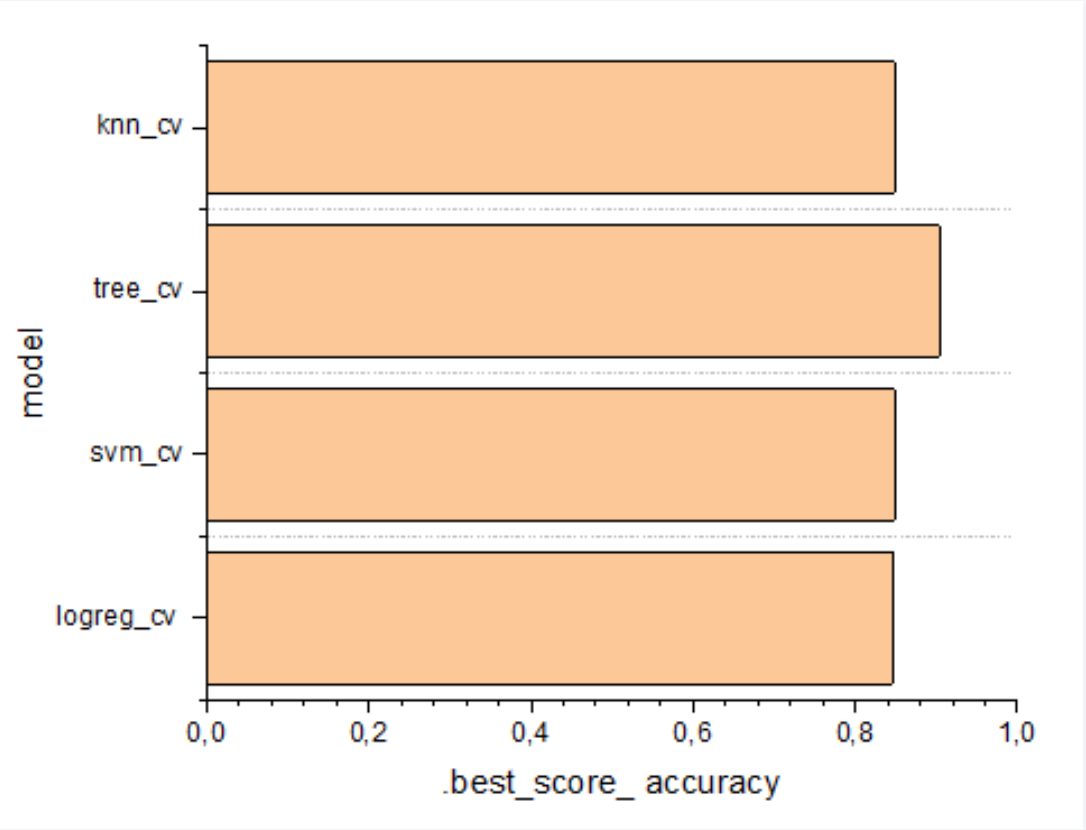
Section 5

Predictive Analysis (Classification)

Classification Accuracy

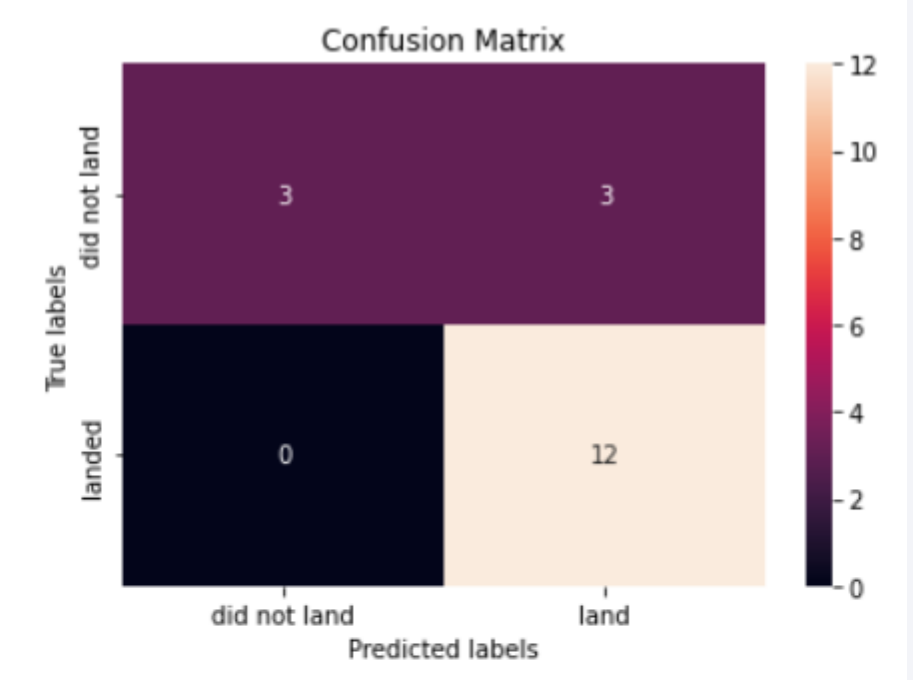
Model accuracy for all built classification models were calculated using two methods: *.best_score_* and *.score*. Results from *.score* calculations did not differ. According to *.best_score_* the decision tree classifier has the highest classification accuracy.

model	.best_score_ accuracy	.score accuracy
logreg_cv	0,8464285714285713	0,8333333333333334
svm_cv	0,8482142857142856	0,8333333333333334
tree_cv	0,9035714285714287	0,8333333333333334
knn_cv	0,8482142857142858	0,8333333333333334



Confusion Matrix of the decision tree

The confusion matrix for the decision tree shows that the classifier can correctly classify cases with successful landing. The problem is recognizing cases with no successful landing – half of them is wrongly classified as successful.



Conclusions

The factors used in this attempt to predict success of the first stage landing were: payload mass, date of the launch, booster version (only Falcon 9), orbit of destination, launch site, number of flights, if there were grid fins, if the hardware was reused, if there were legs, landing pad, block, reused count and serial number. Those are postulated as factors determining if the first stage will land successfully.

This exercise leads the student through predefined path to a result matching the answer key. The result is the tree classifier algorithm recognizing positive landing outcomes with great precision but unable to predict failure (chance of the right classification of 'failed landing' is 50%).

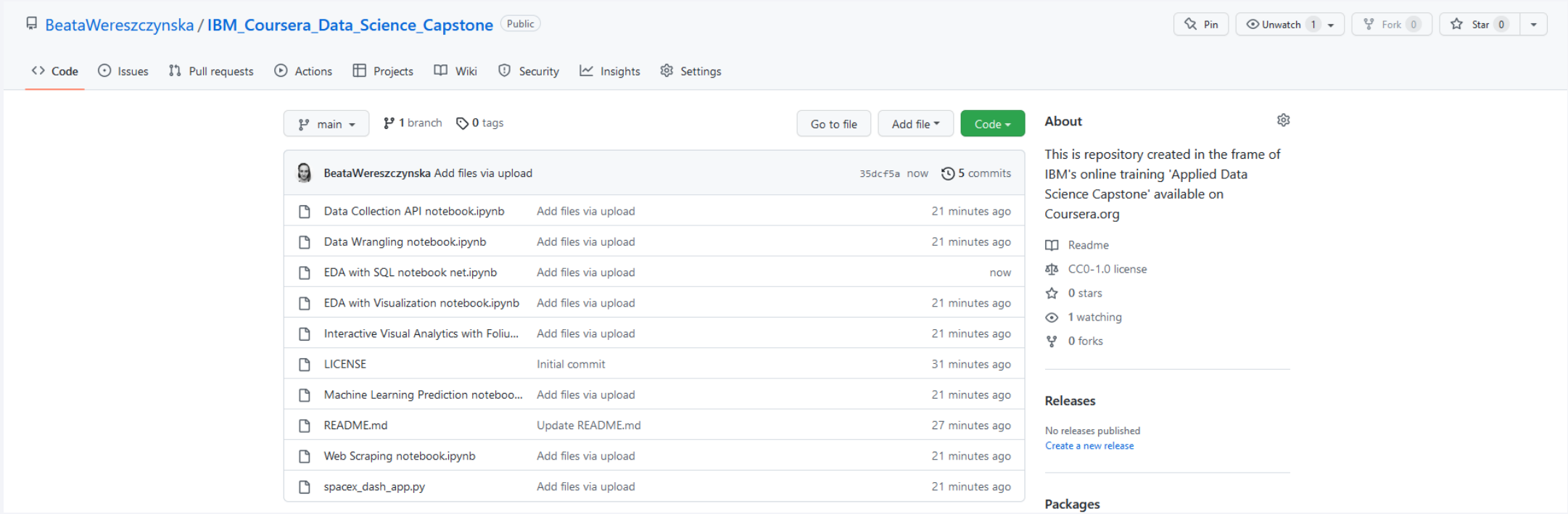
The reason for the tree classifier not recognizing failure is loosing important information on the stage of data collection. The package of information containing if there was landing attempt, if the landing was successful and what kind of landing was it, was reduced to the variable 'Outcome' containing only information if there was successful landing and what type. The information if the landing was planned to happen is lost at this point. Some missions are planned to lose first stage due to optimization of the overall mission success. In the data wrangling notebook template we can read IBM's commentary: *None ASDS and None None these represent a failure to land*. This is not exactly correct as *None None* could mean no landing attempt. There is no surprise in the algorithm being unable to distinguish failure when learning on this type of misleading mixed data.

Classification accuracy can also be improved by different way of dealing with missing values. In this exercise missing values of payload mass were substituted with mean payload mass calculated from the dataset. It can be worth checking if removing missing data instead would elevate classification accuracy due to lack of potentially untrue (the actual payload masses could be at the extreme) and thus, misleading data.

The best Machine Learning model created in the frame of this exercise was not a success but building a model able to predict the first stage landing outcome from publicly available data does not seem imposible, after addressing the above issues.

Appendix

All the notebooks and the Dash app created for this project are available on GitHub repository (https://github.com/BeataWereszczynska/IBM_Coursera_Data_Science_Capstone).



Thank you!