Winning Space Race with Data Science

Maria Filip
6/16/2023

Outline

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

Executive Summary

- Summary of methodologies
  - ✓ Data collection using Web Scraping a API
  - ✓ Data Wrangling
  - ✓ EDA using Data Visualization, SQL and Folium
  - ✓ Machine learning using SVM, Decision Tree and Logistic Regression
- Summary of all results
  - ✓ EDA results
  - ✓ Data visualization results
  - ✓ Predictive analysis results

Introduction

- Project background and context
  - Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch.
- Problems you want to find answers
  - ➢ Find the factors that will increase the success of a rocket landing rate
  - ➢ Find the most cost effective option base on features like rocket type, size, location
  - ➢ Predict if the first stage will land successfully

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Requested and parsed the data using Get request to the SpaceX API
- Perform data wrangling
  - Processed data using Exploratory Data Analysis techniques
  - Created landing outcomes labes
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Found best Hyperparameter for SVM, Classification Trees and Logistic Regression

Data Collection

- Data was gathered by making get request to the SpaceX API
  - We used basic data wrangling and formation to extract the more relevant information. We stored the information in a list that we used to create a data frame.
  - We use BeautifulSoup to get the data from the website
- We loaded the dataset into a corresponding table in a DB2 database.
  - We executed SQL queries to determine the const of a launch

Data Collection – SpaceX API

- We requested rocket launch data from SpaceX API with the following URL:
  spacex_url="https://api.spacexdata.com/v4/launches/past"

- GitHub URL of the completed SpaceX API calls notebook
  https://github.com/MariaProjects/IBM_Projects/blob/master/spacex-data-collection-api%20(2).ipynb

In [5]:
```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Now let's start requesting rocket launch data from SpaceX API with the following URL:

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

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

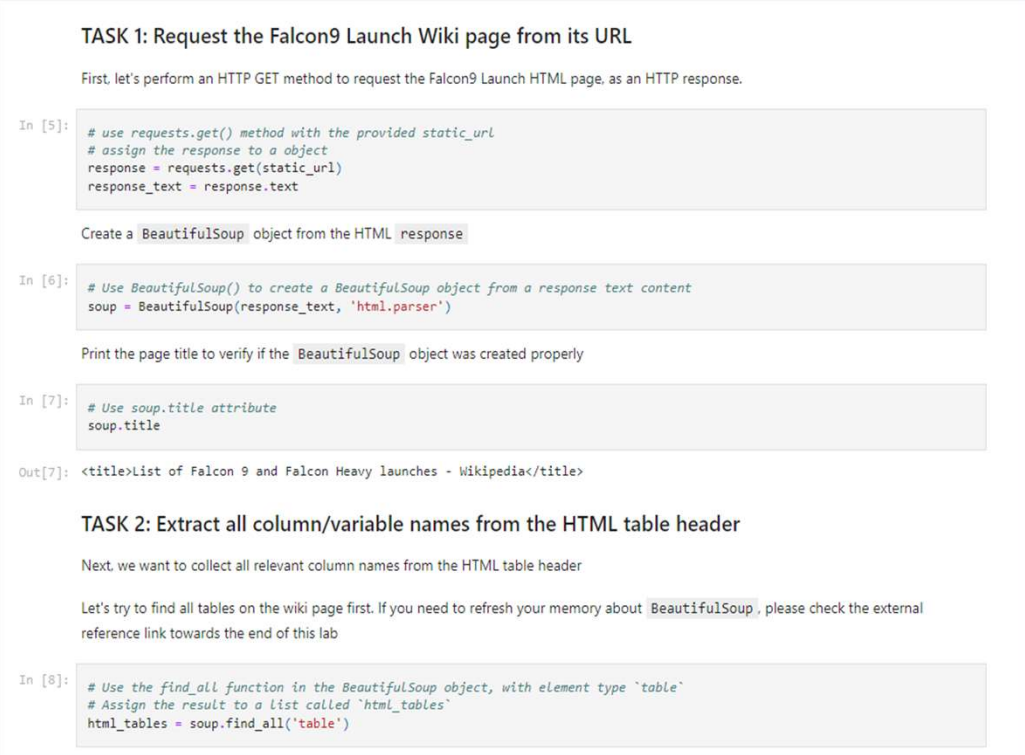
Check the content of the response

In [8]:
```
print(response.content)
```

Data Collection - Scraping

- We web scrap the data for the Falcon 9 launch records using BeautifulSoup. Then converted our data into DataFrame.
- GitHub URL of the completed web scraping notebook link:

https://github.com/MariaProjects/IBM_Projects/blob/master/web%20scraping%20(1).ipynb



TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
response_text = response.text
```

Create a `BeautifulSoup` object from the HTML `response`

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response_text, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

In [8]:
```
# 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')
```

Data Wrangling

- We looked at the data types and missing values
- We calculated the number of launches for each site and the number of occurrence and outcomes of each orbit
- We created an outcome column that labeled our data with success or failure
- We calculate the success rate
- https://github.com/MariaProjects/IBM_Projects/blob/master/jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

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 it to the variable `landing_class`:

In [15]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of eac one means the first stage landed Successfully

In [16]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

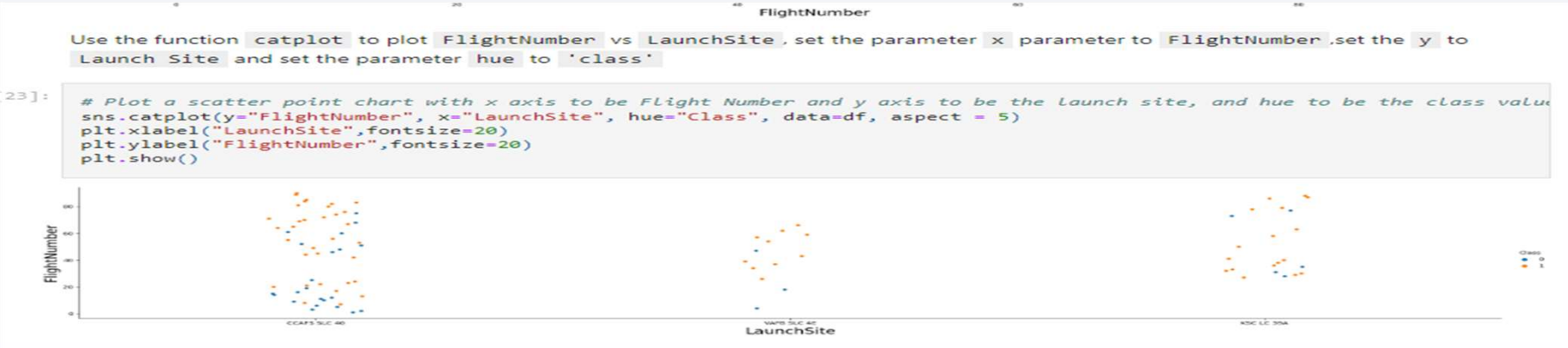
Out[16]:

	Class
0	0
1	0
2	0
3	0
4	0

EDA with Data Visualization

- We use data visualization to explore the data and compare the relationship between different features like flite number, site launch, success rate, orbit type.

https://github.com/MariaProjects/IBM_Projects/blob/master/jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(1).ipynb



EDA with SQL

- We loaded the SpaceX dataset into DB2 databased.
- Used SQL language to execute Exploratory Data Analysis.

- https://github.com/MariaProjects/IBM_Projects/blob/master/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

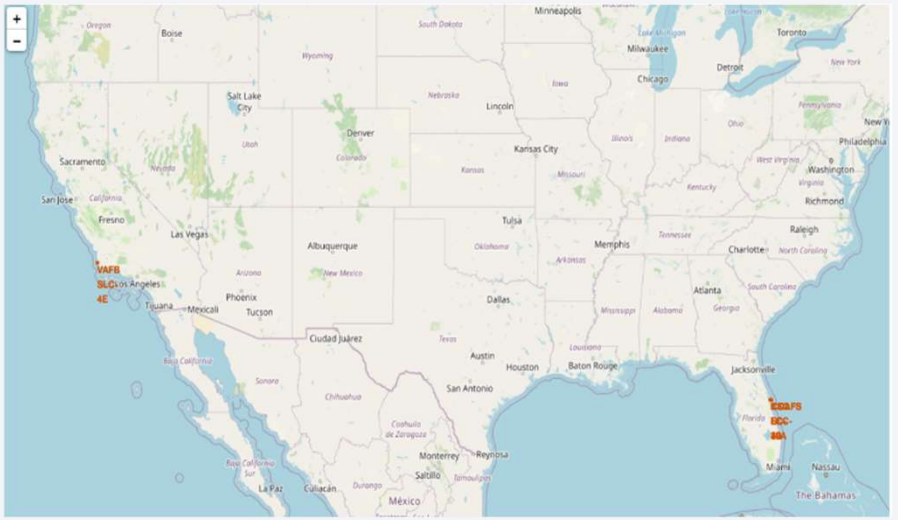
```
In [30]: %sql SELECT Landing_Outcome, COUNT(*) AS OUTCOME_COUNTS FROM SPACEXTBL WHERE DATE BETWEEN "04-06-2010" A
 * sqlite:///my_data1.db
Done.
```

Out[30]:

Landing_Outcome	OUTCOME_COUNTS
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	7
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1

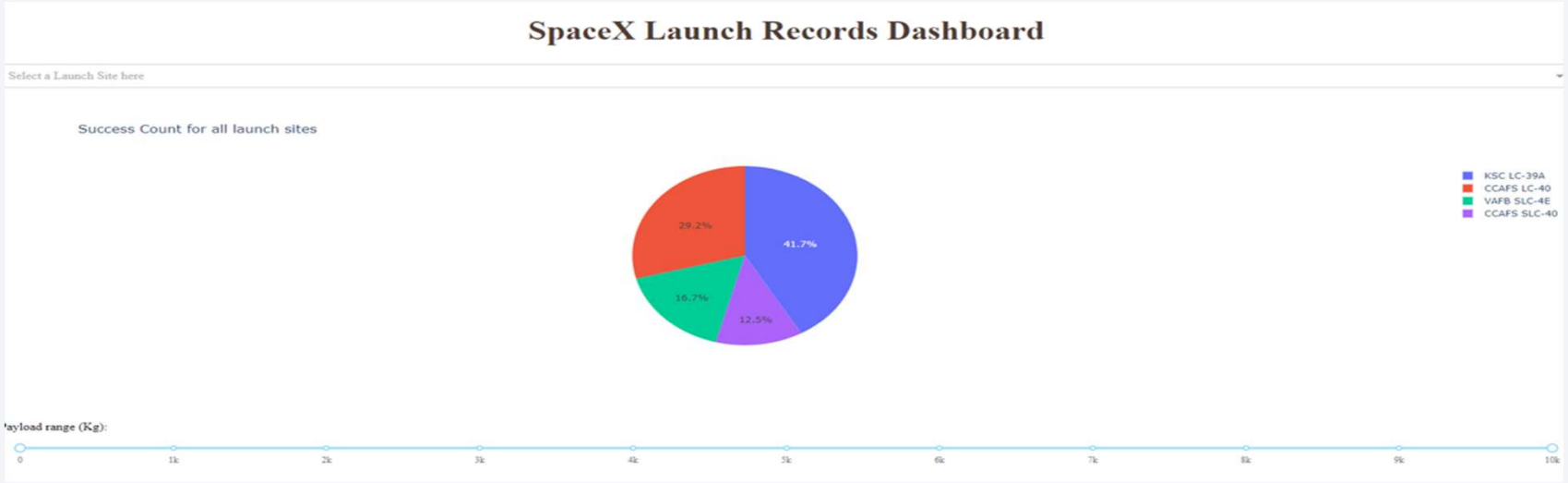
Build an Interactive Map with Folium

- We used Folium to mark all launch sites on a map, to mark the success/failed launches for a site on the map and to calculate the distance between a launch site and its proximities
- Having a good visualization of our data is very important and make it easier to understand
- https://github.com/MariaProjects/IBM_Projects/blob/master/jupyter_launch_site_location.jupyterlite%20(2).ipynb



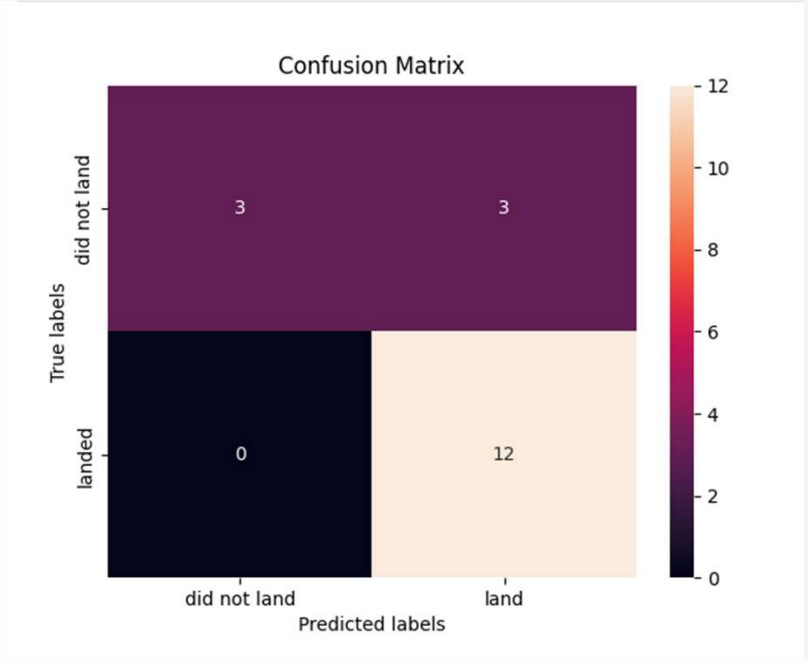
Build a Dashboard with Plotly Dash

- We build an interactive dashboard with Plotly dash and showed the relationship between Outcome and Paylaod
- https://github.com/MariaProjects/IBM_Projects/blob/master/spacex_dash_app.py



Predictive Analysis (Classification)

- First we standardize and split the data into train and test
- We built different machine learning model using GridSearchCV
- We improved the accuracy using the algorithm tuning
- We found the best performing model
- https://github.com/MariaProjects/IBM_Projects/blob/master/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20(1).ipynb

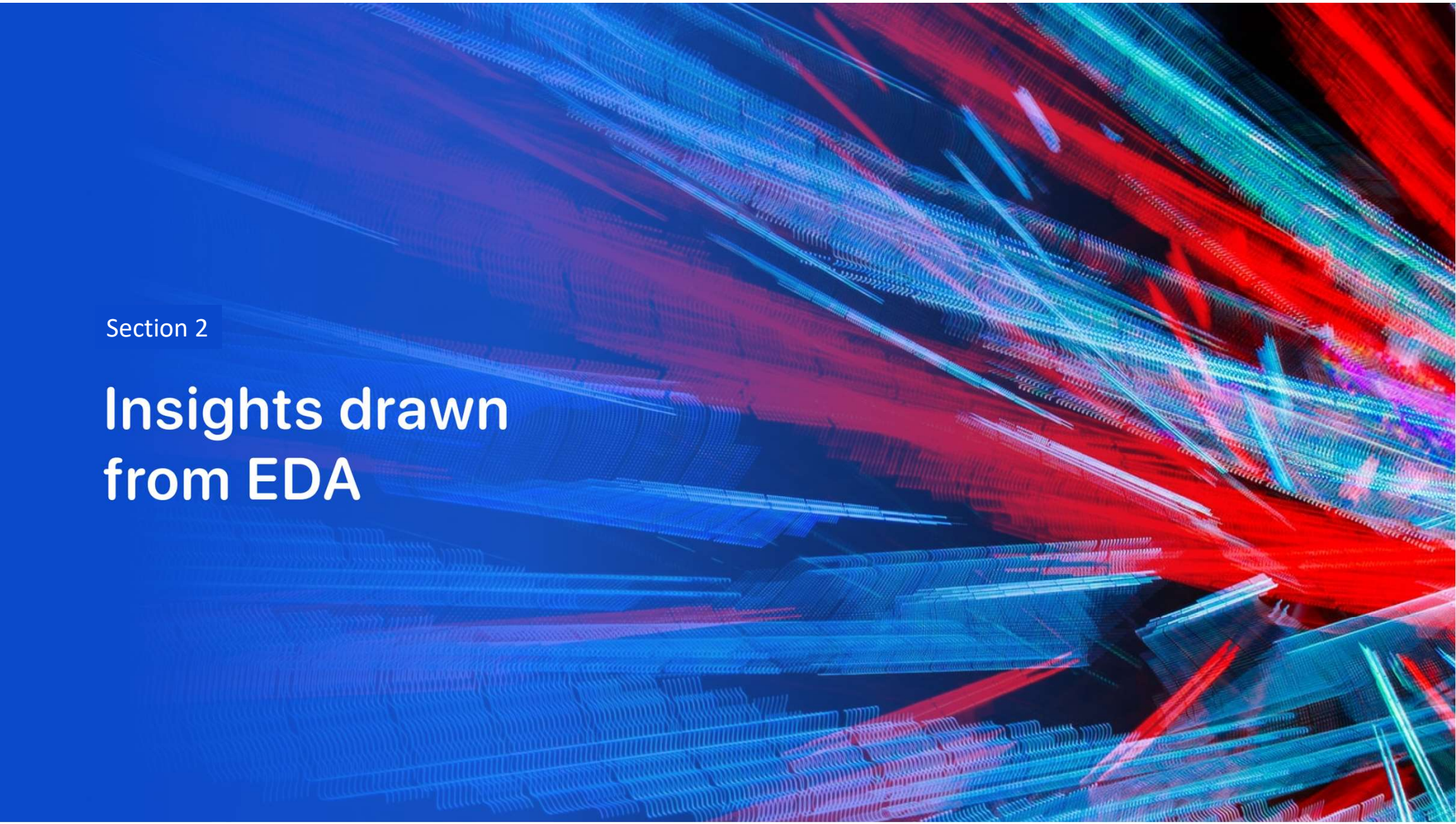


Results

EDA results

Data visualization results

Predictive analysis results

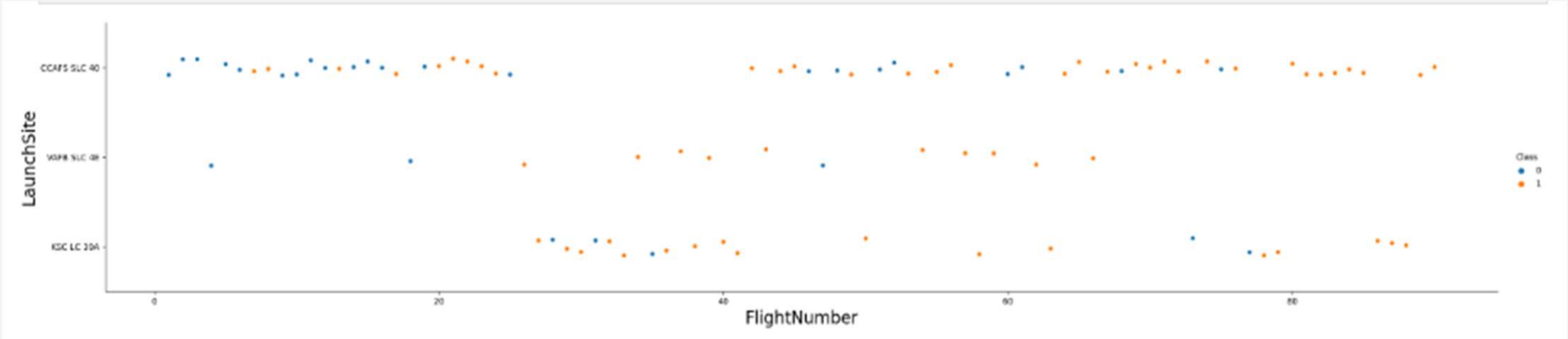


Section 2

Insights drawn from EDA

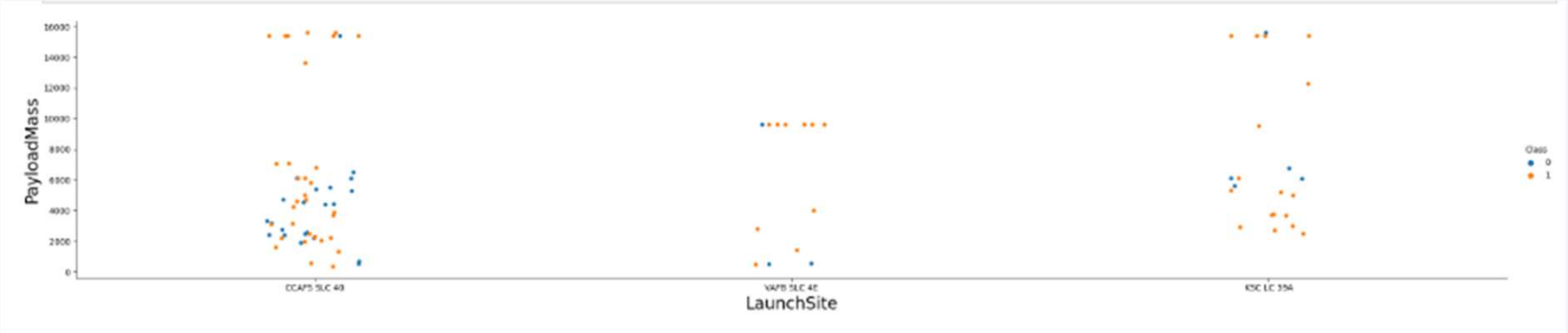
Flight Number vs. Launch Site

- The plot below shows the relationship between the number of flight for each site and the success rate



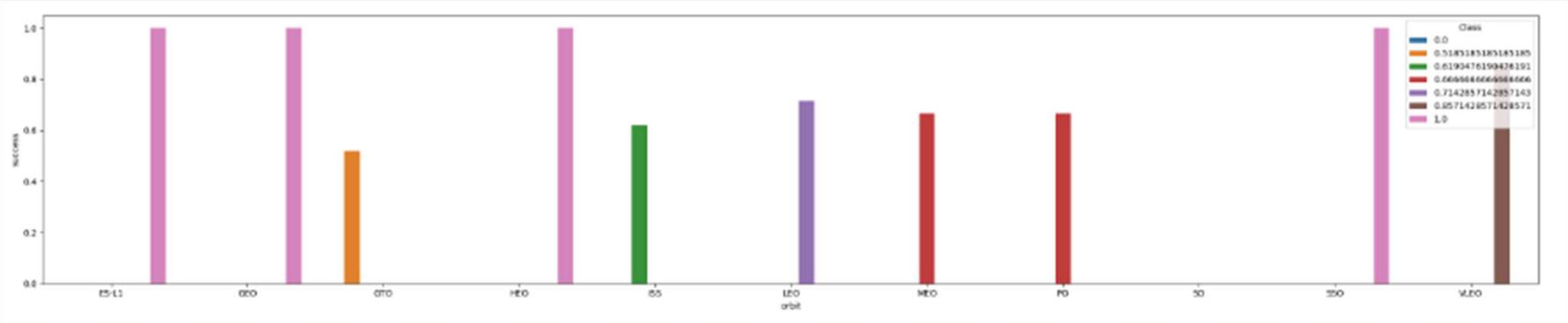
Payload vs. Launch Site

- The image bellow shows the relationship between the payload and the launch site with the success rate



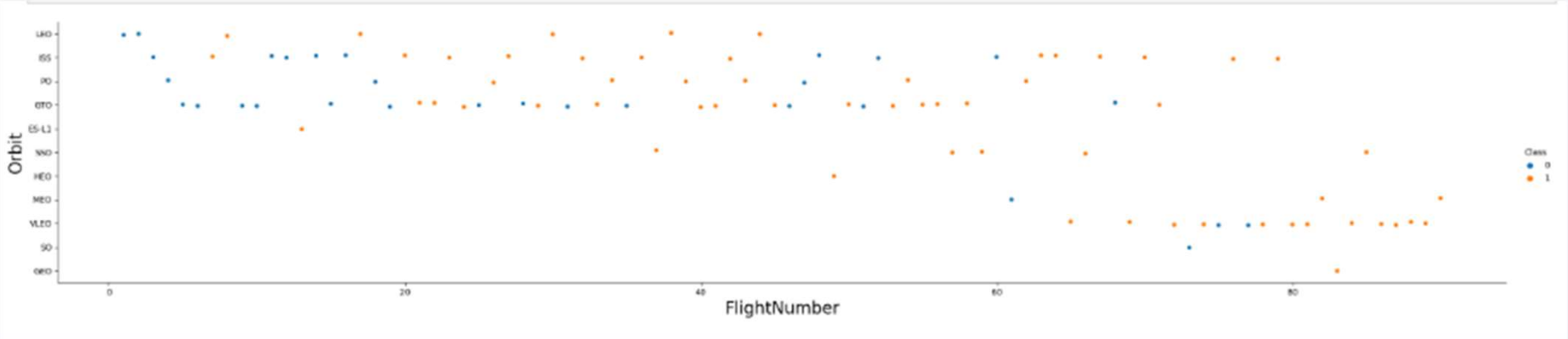
Success Rate vs. Orbit Type

- The image bellow shows the success rate of each orbit type



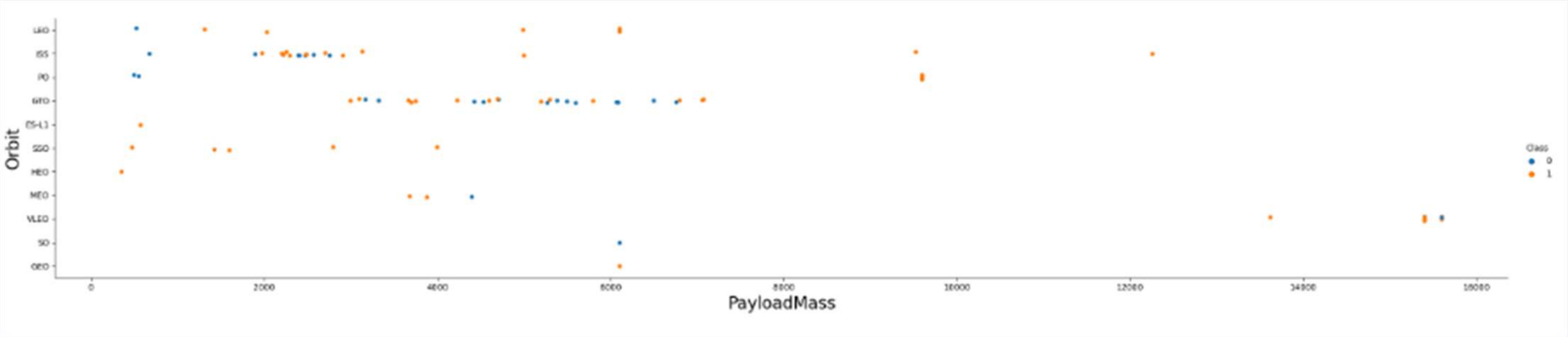
Flight Number vs. Orbit Type

- The mage bellow shows the relationship between the number of Flight vs. Orbit type



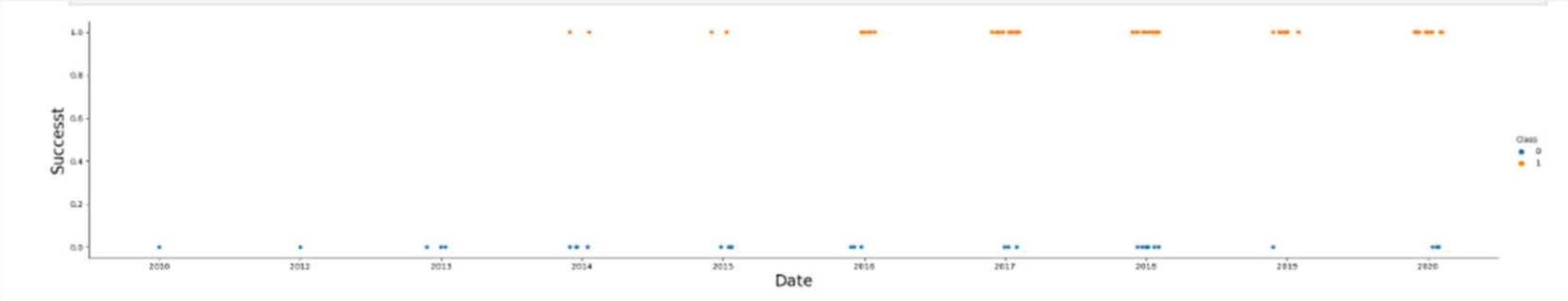
Payload vs. Orbit Type

- The graph bellow shows the relationship between the payload vs. orbit type



Launch Success Yearly Trend

- Yearly average success rate



All Launch Site Names

- We used the DISTINCT key word to get a list with all distinct launch sites
- Query: SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL

```
In [12]: %sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
 * sqlite:///my_data1.db
Done.
Out[12]:
```

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
None

Launch Site Names Begin with 'CCA'

- We used the Like key word to get five records the words that start with CCA
- Query: SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE ‘CCA%’ LIMIT 5;

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

```
In [11]: %sql select * from SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
 * sqlite:///my_data1.db
Done.
```

Out[11]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)

Total Payload Mass

- We use the SUM function to calculate the total payload carried by boosters from NASA
- Query: SELECT SUM(PAYLOAD_MASS_KG) GROM SPACEXTBL WHERE CUSTOMER =='NASA (CRS)';

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

```
In [16]: %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER == "NASA (CRS)";
 * sqlite:///my_data1.db
Done.
Out[16]: SUM(PAYLOAD_MASS__KG_)
                          45596.0
```

Average Payload Mass by F9 v1.1

- We use the AVF function to calculate the average payload mass carried by booster version F9 v1.1
- Query: SELECT AVG(PAYLOAD_MAS_KG_) FROM SPACEXTBL BOOSTER_VERSION = ‘F9 v1.1’;

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

```
In [17]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version = "F9 v1.1";
 * sqlite:///my_data1.db
Done.
Out[17]: AVG(PAYLOAD_MASS__KG_)
         2928.4
```

First Successful Ground Landing Date

- We used the MIN function find the dates of the first successful landing outcome on ground pad
- Query: SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING_OUTCOME == 'SUCCESS (GROUND PAD);

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

Hint:Use min function

In [19]:
```
%sql SELECT MIN(DATE) FROM SPACEXTBL WHERE Landing_Outcome == "Success (ground pad)";
```

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

Out[19]:

MIN(DATE)
01/08/2018

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the DISTINCT function to list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
- Query: SELECT DISTINCT BOOSTER_VERSION FORM SPACEXTBL WHERE ALNDING_OUTCOME == 'SUCCESS (DRONE SHIP) AND PAYLOAD_MASS_KG_ BETWEEN 4000 AND 6000;

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

In [20]:
```
%sql SELECT DISTINCT Booster_Version FROM SPACEXTBL WHERE Landing_Outcome == "Success (drone ship)" and PAYLOAD_MASS__KG_ b
```

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

Out[20]:

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

Total Number of Successful and Failure Mission Outcomes

- We used the Count function to calculate the total number of successful and failure mission outcomes
- Query: SEELCT MISSION_OUTCOME, COUNT(*) AS OUTCOME_TOTAL FORM SPACEXTBL GROUP BY MISSION_OUTCOME;

List the total number of successful and failure mission outcomes

In [21]:
```
%sql SELECT Mission_Outcome, COUNT(*) AS Outcome_Total from SPACEXTBL GROUP BY Mission_Outcome;
```

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

Out[21]:

Mission_Outcome	Outcome_Total
None	898
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- We used the MAX function to list the names of the booster which have carried the maximum payload mass
- Query: SELECT BOUSTER_VERSION FORM SPACEXTBL WHERE PAYLAOD_MASS_KG_ = (SELECT MAX(PAYLOAD_MASS_KG_) FROM SPACEXTBL);

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

```
In [23]: %sql SELECT Booster_Version FROM SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
 * sqlite:///my_data1.db
Done.
```

Out[23]:

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

- We use the substr method to list the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Query: SELECT SUBSTR(DATE, 4, 2) AS MONTH, DATE, BOOSTER_VERSION, LAUNCH_SITE, [LANDING_OUTCOME] FORM SPACEXTBL WHERE [LANDING_OUTCOME]='FAILURE (DRONE SHIP)' AND SUBSTR(DATE,7,4)='2015';

```
In [26]: %sql SELECT substr(DATE,4,2) AS MONTH, DATE, BOOSTER_VERSION, LAUNCH_SITE, [Landing_outcome] from spacextbl where [Landing_
 * sqlite:///my_data1.db
Done.
```

Out[26]:

MONTH	Date	Booster_Version	Launch_Site	Landing_Outcome
10	01/10/2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	14/04/2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- We used the COUNT function to 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
- Query: SELECT LANDING_OUTCOME, COUNT(*) AS OUTCOME_COUNTS FORM SPACEXTBL WHERE DATE BEWTEEN ’04-06-2010’ AND ‘20-03-2017’ GROUPBY OUTCOME_COUNTS DESC;

Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

In [30]:
```
%sql SELECT Landing_Outcome, COUNT(*) AS OUTCOME_COUNTS FROM SPACEXTBL WHERE DATE BETWEEN "04-06-2010" AND "20-03-2017" GRO
```

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

Out[30]:

Landing_Outcome	OUTCOME_COUNTS
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	7
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1

Section 3

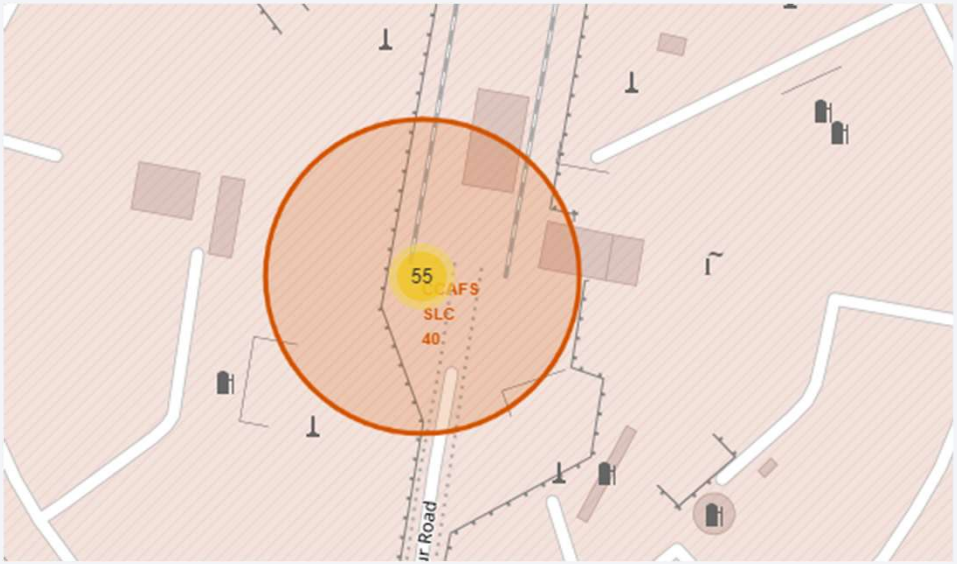
Launch Sites Proximities Analysis

Launch site location on map

- Launch location are located on the east and west costs.

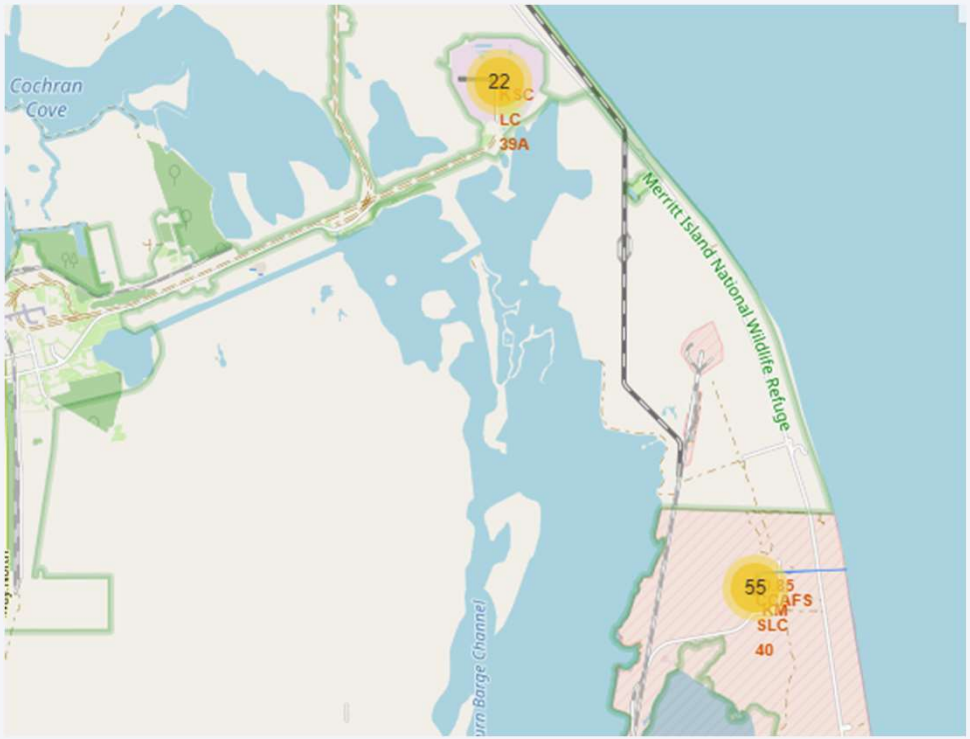
CCAFS SLC 40

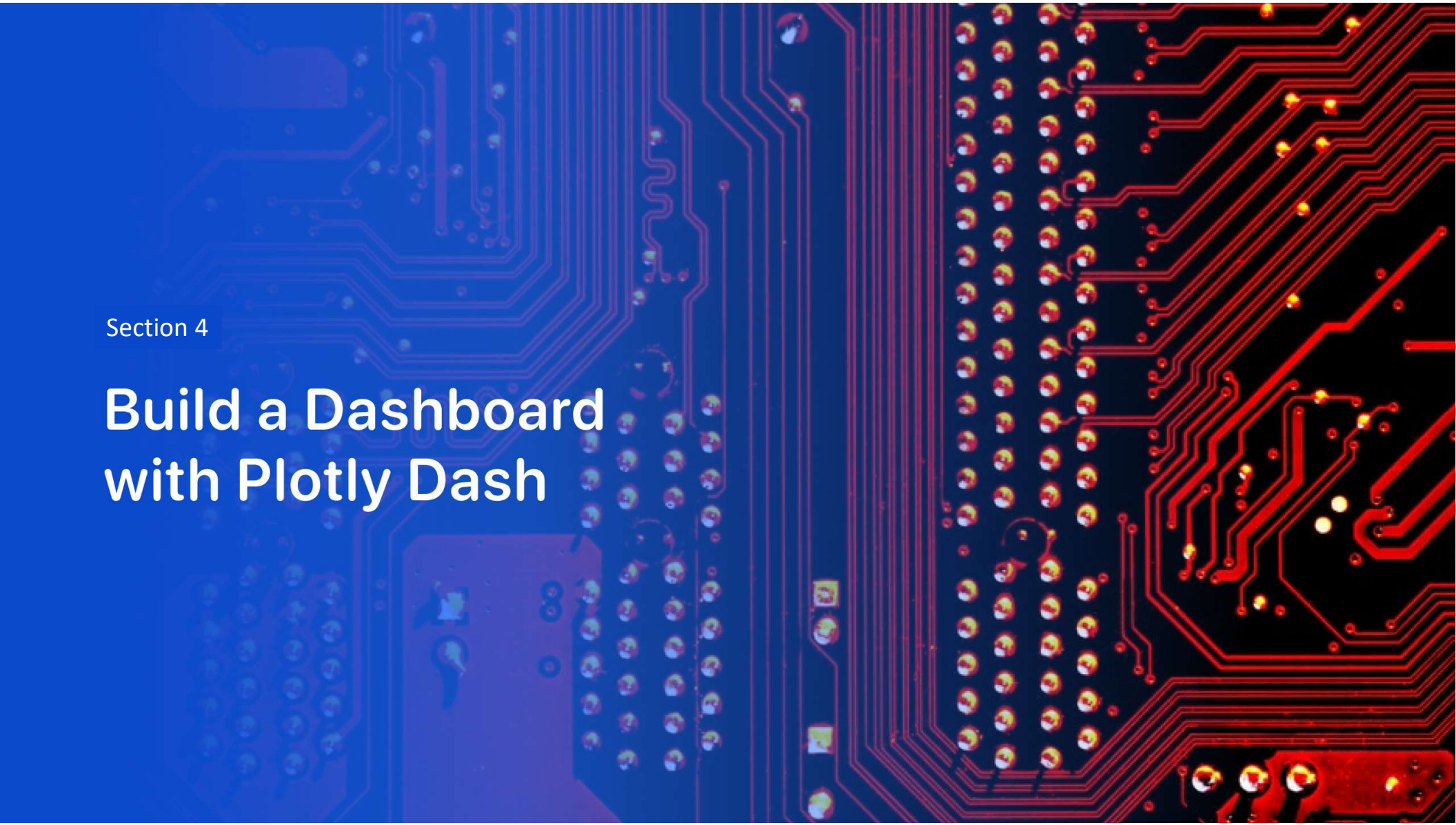
- CCAFS SLC 40 Location



Launch site distance

- Launch site distance to landmarks



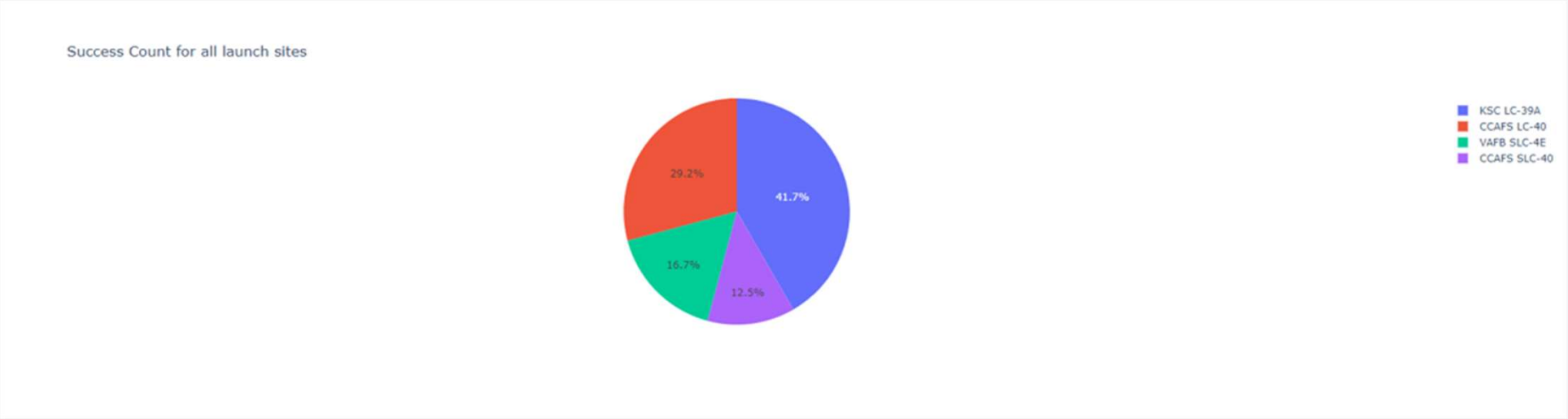


Section 4

Build a Dashboard with Plotly Dash

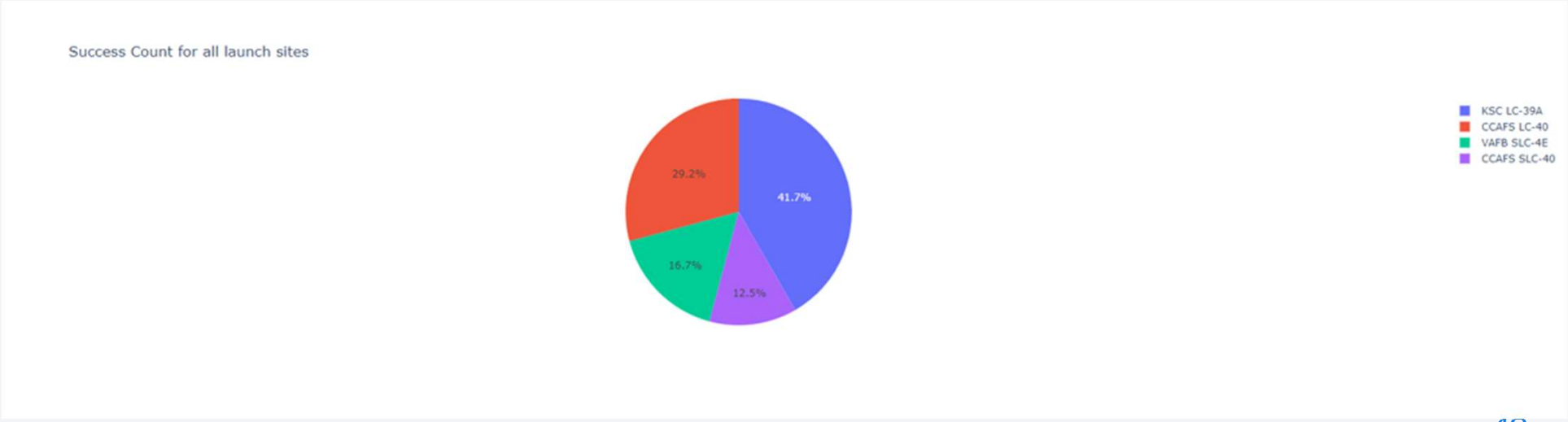
SPACEX LAUNCH RECORD DASHBOARD

- The chart bellow shows the success rate for all sites



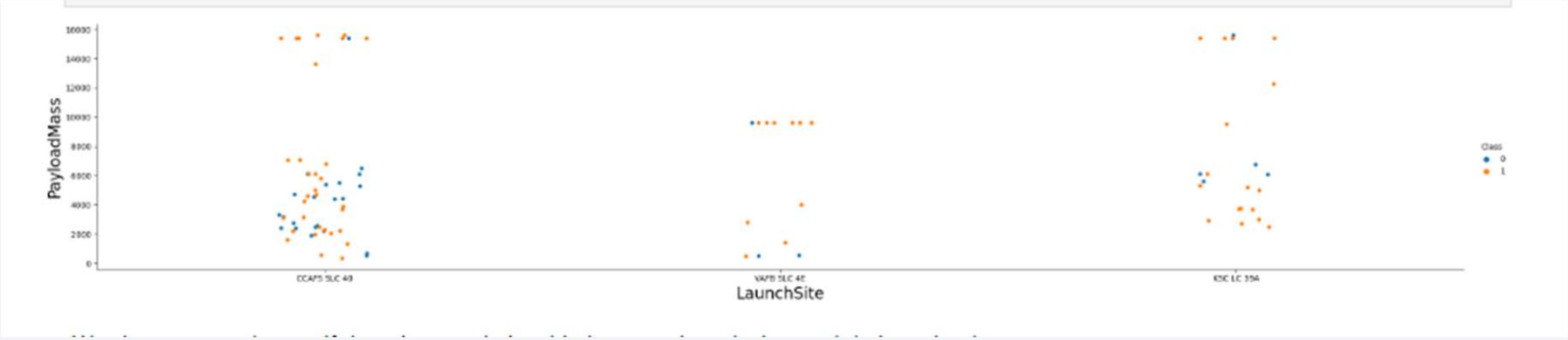
Highest launch success ration

- KSC LC-39A has the highest launch success ratio



Payload and success rate for each site

- Highest payload have the largest success rate for each site



Section 5

Predictive Analysis (Classification)

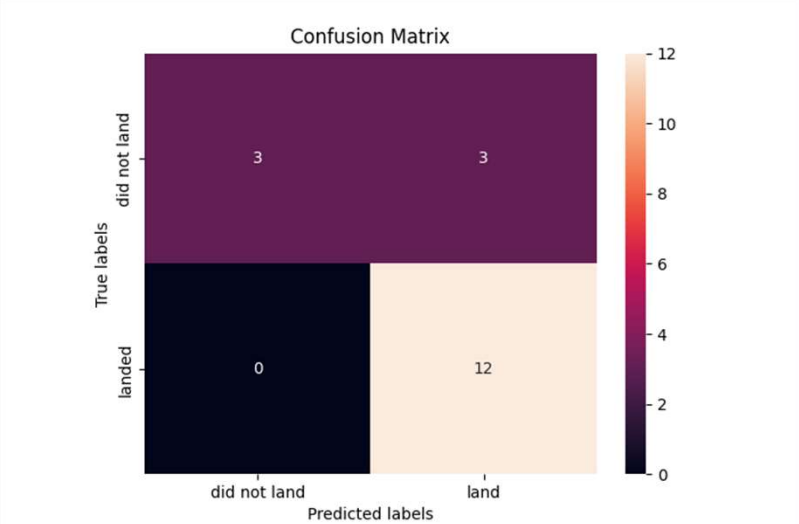
Classification Accuracy

- The method that perform best is Decision Tree

```
n [ ]: import numpy as np
       import matplotlib.pyplot as plt

       plt.barh(method, accuracy)
       plt.xlabel('Accuracy')
       plt.ylabel('Method')
       plt.show()

       results_df = {'method': method,
        'accuracy': accuracy}

       frame = pd.DataFrame(results_df)
       frame
```

Confusion Matrix

- The confusion matrix 3+12 is way bigger then 0+3 showing a good accuracy



Conclusions

- Decision Tree is the best machine learning algorithm
- KSC LC-39A has the highest launch success ratio
- Launch success rates start to increases in 2013
- Launch site are situated on costal side East or East
- If the payload is higher then the success rate is highter

Appendix

- Link to my Git Hub:

- https://github.com/MariaProjects/IBM_Projects

Thank you!