IBM Capstone Project

Space-X

Miguel Angel Victoria

Outline

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

Executive Summary

Summary of Methodologies

- Data collection/Wrangling
- EDA with SQL
- EDA with Data Visualization
- Interactive Folium map & Plotly Dashboard
- Predictive Analysis (Classification)

Summary of all Results

- Data Analysis results
- Interactive analytics demos pictures
- Predictive Analysis results

Introduction

Project Background and context

In this project is presented the probabilities and factors that determine whether the first stage of the Falcon 9 will land successfully or not. With approximately $62 million of dollars being advertised on Space X website and $165 million of dollars from other provider for the Falcon 9 rocket launches, it is vital for Space X whether a Falcon 9 rocket lands successfully or not, because Space X can tremendously benefit from reusing its rockets. By Analyzing the probabilities and factors influencing the Falcon 9 landings, costs of launches can be determined.

Main constrains to be solved

- Factors, and their relationships, that influence Falcon 9 launches.
- Conditions to be accomplished to increase chances of successful landings.

Methodology

Methodology

- **Data collection methodology:**

  Web scraping from Wikipedia and SpaceX API.

- **Perform data wrangling:**

  One Hot Encoding for machine learning and dropping irrelevant columns and values

- **Perform exploratory data analysis (EDA) using visualization and SQL**

  Scatter Graphs and Bar Graphs to depict the relationships among various variables and their patterns.

- **Perform interactive visual analytics using Folium and Plotly Dash.**

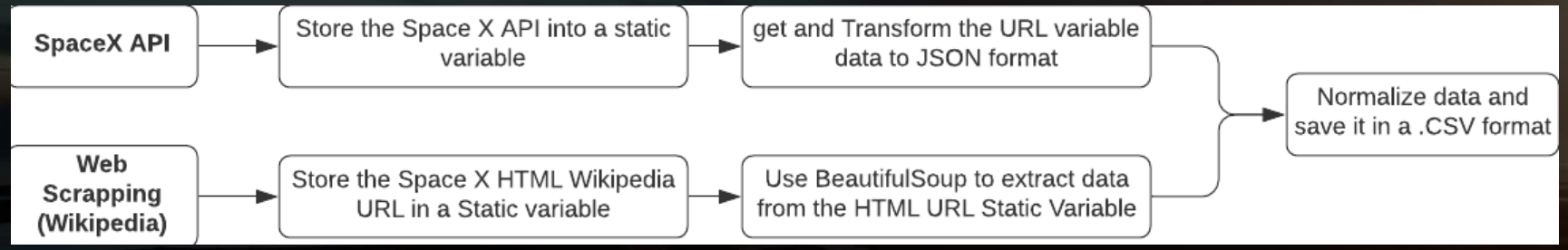
- **Perform predictive analysis using classification models:**

  How to build, tune, evaluate classification models.

Data collection

- Datasets were collected from SpaceX REST API (URL: api.spacedata.com/v4/rockets/) and from web scraping from Wikipedia.

- Datasets contained information about the SpaceX rocket launches such as rockets used, payload delivered, launch and landing specs and landing outcomes, among others. The information from these datasets were utilized to determine and examine the success/failure rate of the Falcon 9 landing.



Data Collection (SpaceX API)

GitHub Notebook

- Getting/Saving SpaceX API into a static variable & converting it into .json format

```
url="http://api.spacexdata.com/v4/launches/past"
responseL = requests.get(url).json()
dataL = pd.json_normalize(responseL)
```

- Apply Pre-determined functions to the established dataset

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

- Create a data frame from a dictionary from the dataset information

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}

df = pd.DataFrame(launch_dict)
```

- Filter out data by Falcon 9 information and export data to CSV format

```
indexnames=df.loc[df['BoosterVersion']!='Falcon 9'].index
df.to_csv('dataset_part_1.csv', index=False)
```

Data Collection (Web scrapping)

GitHub Notebook

- Saving HTML URL into a static variable & converting it into a Beautiful Object

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_
response_Static = requests.get(static_url)

soup = BeautifulSoup(response_Static.content, 'html5lib')
```

- Find tables in the HTML URL and fetching columns' names from tables

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

column_names = []
th_elements = first_launch_table.find_all('th')

for name in th_elements:
    name=extract_column_from_header(name)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

- Create a data frame by parsing the launch HTML table and appending data to keys

```
# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheade
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.
            launch_dict['Flight No.'].append(flight_number)

            #print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            #print(date)
```

- Convert the Dictionary to a data frame and convert it to CSV format

```
df2= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df2.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

GitHub Notebook

- Saving HTML URL into a static variable & identify the percentage of Missing values in each attribute and the type

```
df2=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/1
df2.isnull().sum()/df.count()*100
df2.dtypes
```

- Analyze how many Launch sites there are and how many type of orbits

```
df2['Orbit'].value_counts()
df2['LaunchSite'].value_counts()
```

- Examine the number and occurrences of mission depending the ype of orbit

```
df2['Outcome'].value_counts()
landing_outcomes =df2['Outcome'].value_counts()
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

- Assign Landing Class for bad and good landings outcomes, o and 1, respectively.

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

landing_class = []

#Assign to the "landing_class variable bad/good_outcomes 0 and ones, respectively.(1
landing_class=[0 if outcome in bad_outcomes else 1 for outcome in df2['Outcome']]
df2['Class']=landing_class
df2[['Class']].head(8)
```

- Determine Success rate average of the landing outcomes per class

```
df2["Class"].mean()
```

EDA with Data Visualization

Scatter Plot graphs and bar graphs were created to visually analyze and examine the relationship and correlation among various variables of the SpaceX Falcon 9.

Line graphs were created to analyze the success rate through the years, from 2010 to 2020.

EDA with SQL

- The following SQL queries were performed to better understand and analyze more in detail the collected/processed data:
  - The names of the unique launch sites in the space mission
  - 5 records where launch sites begin with the string 'CCA'
  - The total payload mass carried by boosters launched by NASA (CRS)
  - Average payload mass carried by booster version F9 v1.1
  - List the date when the first successful landing outcome in ground pad was achieved.
  - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - List the total number of successful and failure mission outcomes
  - List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
  - List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  - Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Build an Interactive Map with Folium

- Folium library was utilized to create interactive maps in where the launch sites and their outcomes could be easily located and depicted in the interactive Folium map.
- Marker clusters were created to color, differentiate and plot the Launch_Outcomes into the interactive Folium map. The Launch_Outcomes were assigned as 0 and 1 and Green and Red as failure of success, respectively.

- Once the the Marker clusters and the Launch_Outcomes were plotted on the interactive Folium map, the following questions could be addressed:
- Are launch sites in close proximity to railways?
- Are launch sites in close proximity to highways?
- Are launch sites in close proximity to coastline?
- Do launch sites keep certain distance away from cities?

Build a Dashboard with Plotly Dash

- A Plotly Dashboard was created to provide a dynamic report environment. The following graphs were created to dynamically depict information depending in the filtered data in the slider and the drop-down list in the Plotly Dashboard:

- A pie chart that depicts the total Success Launches for different Launches sites.
- A scatter plot graphs that depicts the correlation between the payload and the launch success depending the booster class type

Predictive Analysis (Classification)

- Model Building:
  - Load and save dataset into a data frame
  - Transform data
  - Split data into Training and test data sets
  - Create GridSearchCV object to find the best parameters from a predetermined dictionary parameters for the Logistic Regression, support vector machine, decision tree classifier and K nearest neighbor's machine learning methods.
  - Calculate/plot Logistic regression, support vector machine, decision tree classifier and K nearest neighbors' accuracy and confusion matrix .

- Best Performing classification model:
  - Each previously mentioned machine learning methods are calculated their respective accuracy and confusion matrix to determine which machine learning is the best method.

Results

Exploratory data analysis results

Interactive analytics demo in screenshots

Predictive analysis results

Insights drawn from EDA

Flight Number vs Launch Site

The greater LaunchSite the greater the success rate. (Blue: Failure, Orange: Success).

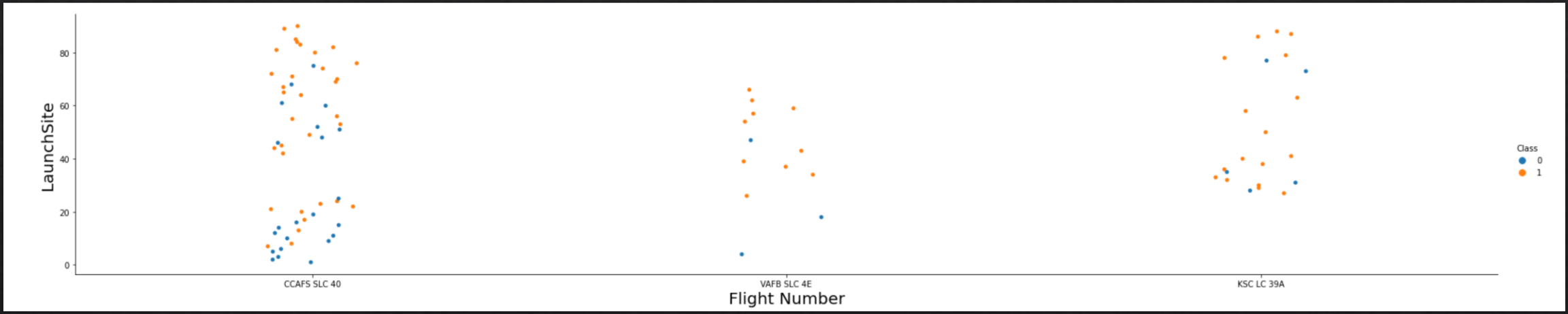


Figure 1

GitHub Notebook

Payload vs Launch Site

For each LaunchSite there is a range of Pay Load Mass in where the success rate is greater than the failure rate.

- CCAFS SLC 40: The success rate between the Pay Load Mass of 14,000 kg and 16,000 kg is approximately an 83.3%.
- VAFB SLC 4E: The success rate around 10,000 kg Pay Load Mass is 75%.
- KSC LC 39A: The success rate around 16,000 kg Pay Load Mass is 80%

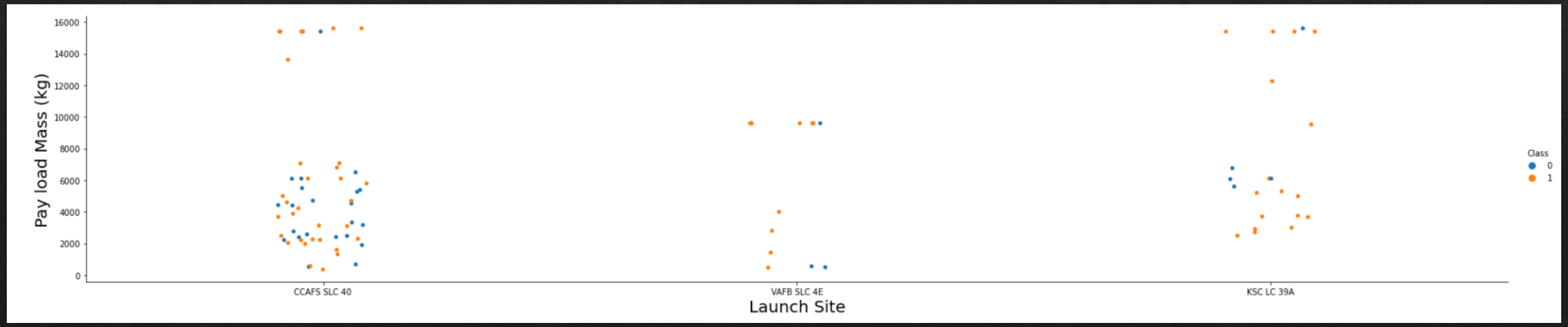
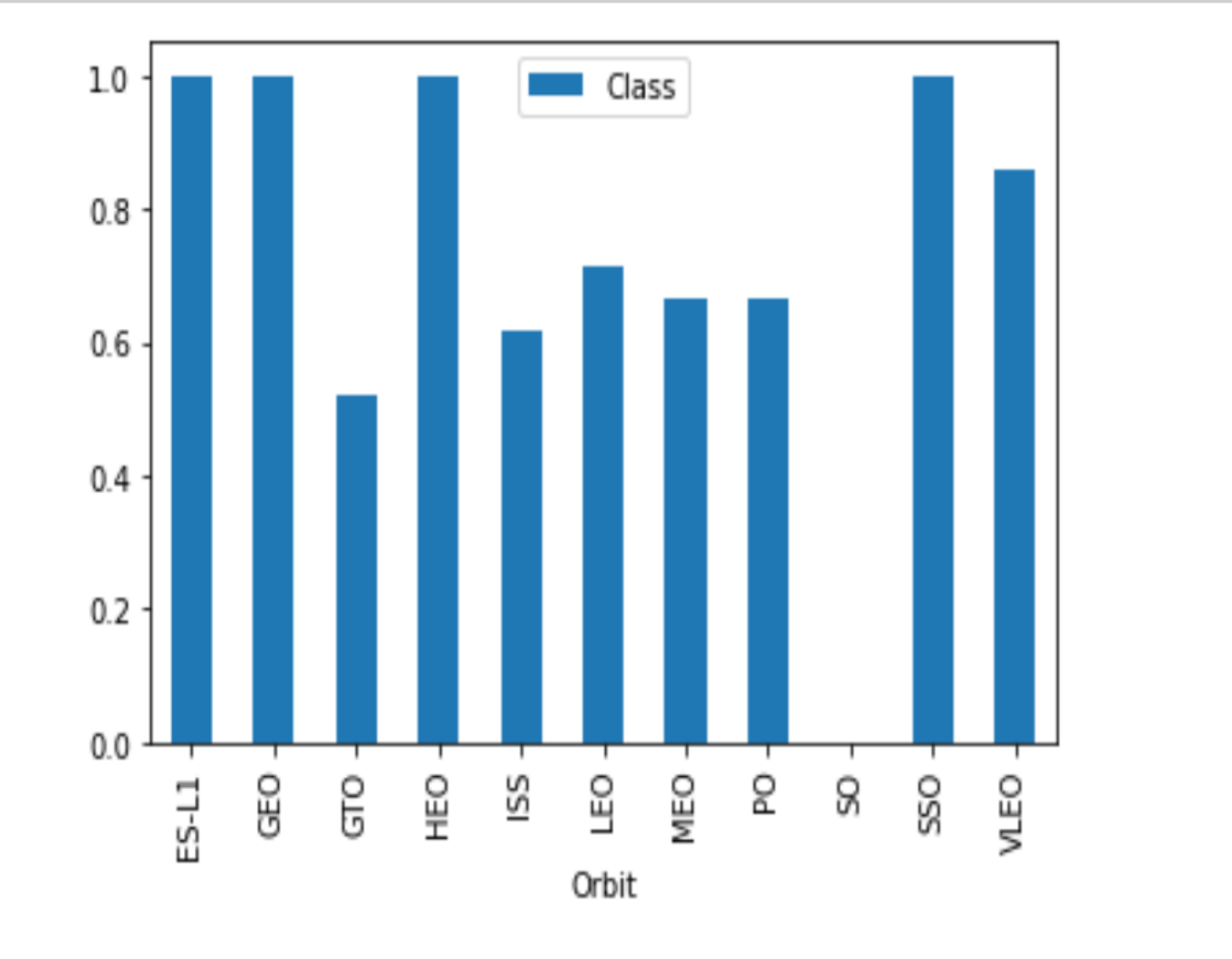


Figure 2

GitHub Notebook



Success Rate vs Orbit Type

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

Figure 3

[GitHub Notebook]

Flight Number vs Orbit Type

- There seems to be a relationship between the LEO and the VLEO Orbit to the Number of flights after certain range.
- LEO: Greater than 20 flights there seems to be a greater success rate.
- VLEO: Greater than 70 flights there seems to be a greater success rate.

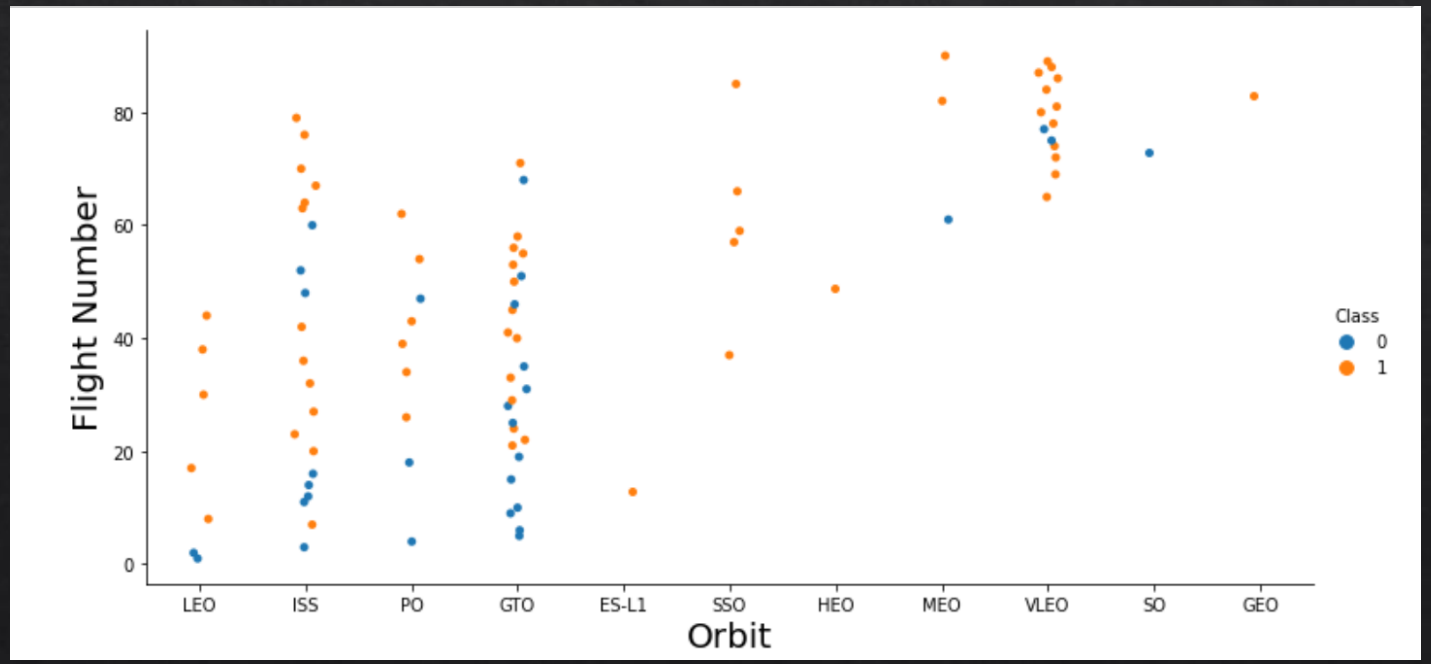


Figure 4

GitHub Notebook

Payload vs Orbit Type

- There seems to be an increase of success rate between Orbits LEO, and ISS to higher Pay Load Mass.
- There seems greater success rates the lighter the Pay Load Mass for the Orbit SOO

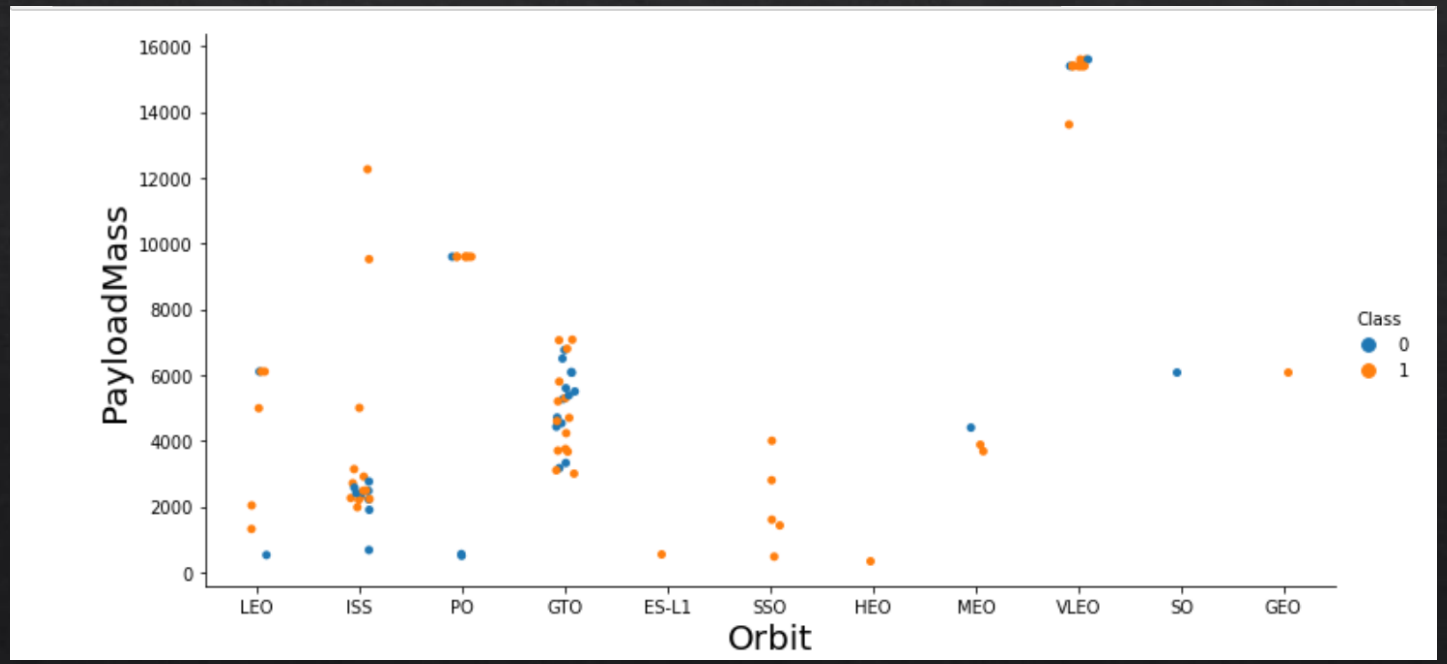


Figure 5

[GitHub Notebook]

Launch Success Yearly Trend

- It can clearly be observed that as time progressed from 2010 to 2020 the success rate was gradually increasing

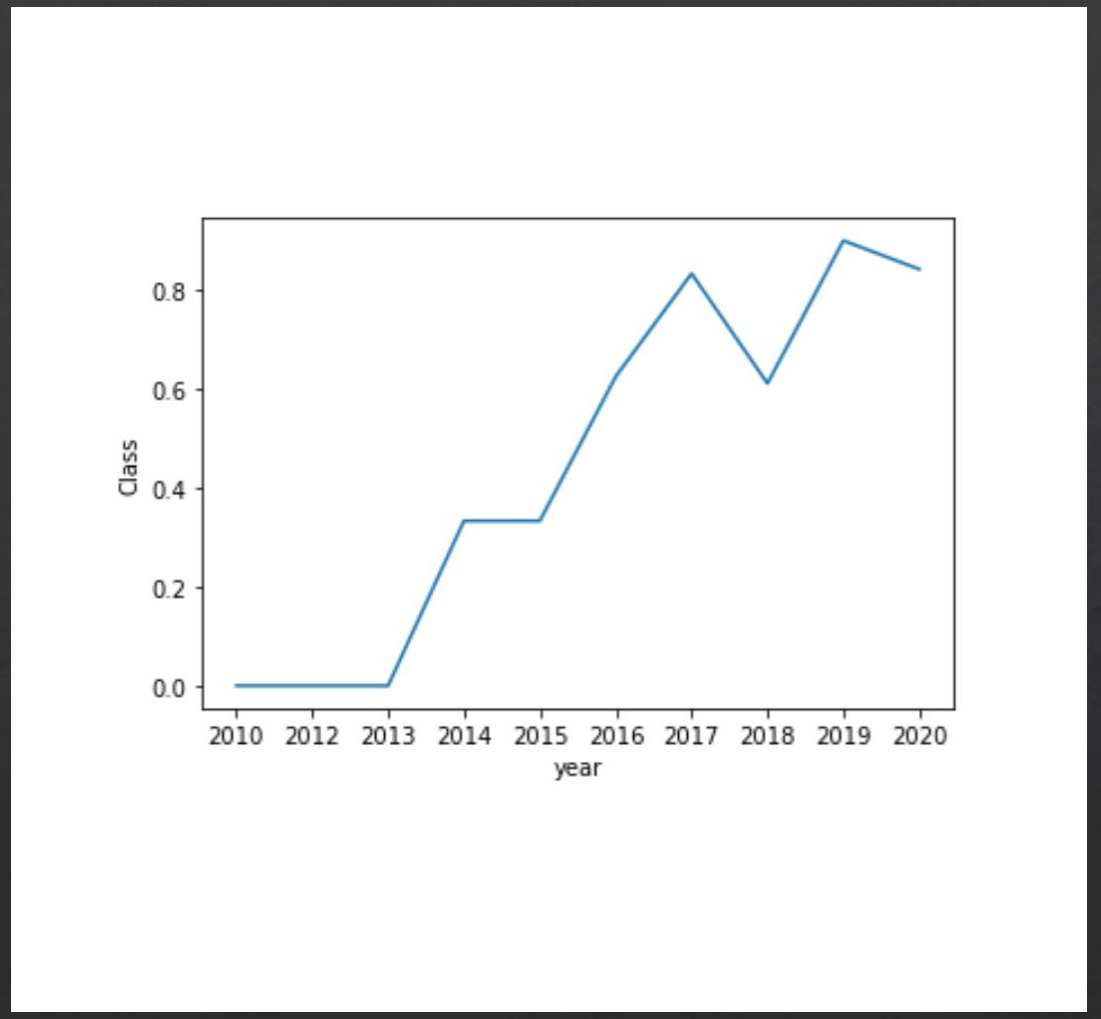


Figure 5

GitHub Notebook

EDA with SQL

All Launch Site Names

SQL QUERY

```
SELECT DISTINCT(Launch_Site) FROM SPACEXTBL
```

Distinct () keyword in SQL allows to call the unique values of a group, in this case the unique values of Launch_Site

SQL Results

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

Launch Site Names Begin with ‘CCA’

SQL QUERY

```
SELECT * FROM SPACEXTBL

WHERE Launch_Site LIKE 'CCA%';
```

SQL Results

DATE	TIME__UTC_	BOOSTER_VERSION	LAUNCH_SITE	PAYLOAD
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of B
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2

The keywords WHERE and LIKE allow to establish a condition to look up a specific valor. Moreover, the wildcard ‘%’ allows to look up values beginning or ending with specific letters, in this case Lauch Sites beginning with CCA

Total Payload Mass

SQL QUERY

```
SELECT SUM(PAYLOAD_MASS__KG_) AS NASA_CRS_TOTAL_PAYLOAD_MASS_KG  FROM SPACEXTBL

WHERE Customer = 'NASA (CRS)';
```

SQL Results

NASA_CRS_TOTAL_PAYLOAD_MASS_KG
45596

The keyword SUM() and the WHERE equals allow to add up and find the total sum of a specific attribute, in this case the total sum of PAYLOAD_MASS__KG_ of costumer type ‘NASA (CRS)’.

Average Payload Mass by F9 v1.1

SQL QUERY

```
SELECT avg(PAYLOAD_MASS__KG_) AS "Booster Version F9 v1.1_TOTAL_AVERAGE_PAYLOAD_MASS_KG"  FROM SPACEXTBL
WHERE Booster_Version = 'F9 v1.1';
```

SQL Results

Booster Version F9 v1.1_TOTAL_AVERAGE_PAYLOAD_MASS_KG
2928

The keyword avg() and the WHERE equals allow to find the average a specific attribute, in this case the average of Booster_Version F9 v1.1

First Successful Ground Landing Date

SQL QUERY

```
SELECT MIN (DATE) AS "Min Date" FROM SPACEXTBL

WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

The keyword min() and the WHERE equal allow to find the first date in where there was a successful Ground Landing

SQL Results

Min Date
2015-12-22

Successful Drone Ship Landing with Payload Between 4000 and 6000

SQL QUERY

```
SELECT Booster_Version FROM SPACEXTBL

WHERE LANDING__OUTCOME = 'Success (drone ship)' AND  PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000  ;
```

SQL Results

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

The keyword WHERE equal and AND keywords allow to establish two conditions and look up a certain value, in this case Successful Drone Ship Landing with Payload Between 4000 and 6000

Total Number of Successful and Failure Mission Outcomes

SQL QUERY

```
SELECT * FROM
(SELECT COUNT(Mission_Outcome) AS Successful_Mission_Outcomes FROM SPACEXTBL WHERE Mission_Outcome LIKE '%Succes%'),
(SELECT COUNT(Mission_Outcome) AS Failure_Mission_Outcomes FROM SPACEXTBL WHERE Mission_Outcome LIKE '%Failure%')
```

A Subquery was created to retrieve the number of successful and Failure Mission Outcomes

SQL Results

SUCCESSFUL_MISSION_OUTCOMES	FAILURE_MISSION_OUTCOMES
100	1

Booster Carried Maximum Payload

SQL QUERY

```
Select Booster_Version, max(PAYLOAD_MASS__KG_) as MAXBOOSTERS_PAYLOADMASS FROM SPACEXTBL
GROUP BY Booster_Version
ORDER BY MAXBOOSTERS_PAYLOADMASS DESC
```

SQL Results

BOOSTER_VERSION	MAXBOOSTERS_PAYLOADMASS
F9 B5 B1048.4	15600
F9 B5 B1048.5	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1049.7	15600

The MAX()keyword allows to find the maximum PAYLOAD_MASS__KG_ of every Booster version. Then, GROUP BY and ORDER BY DESC allow to group by Booster version and order in descent order the Pay Load Mass of the boosters.

2015 Launch Record

SQL QUERY

```
SELECT LANDING__OUTCOME,Booster_Version, Launch_Site, year(Date) as YEAR FROM SPACEXTBL

WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND year(Date) = 2015
```

A double cognitional was utilized to find LANDING__OUTCOME equal to Failure (drone ship) occurred in 2015

SQL Results

LANDING__OUTCOME	BOOSTER_VERSION	LAUNCH_SITE	YEAR
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015

Rank Landing Outcomes Between 2010-06-4 and 2017-03-20

SQL QUERY

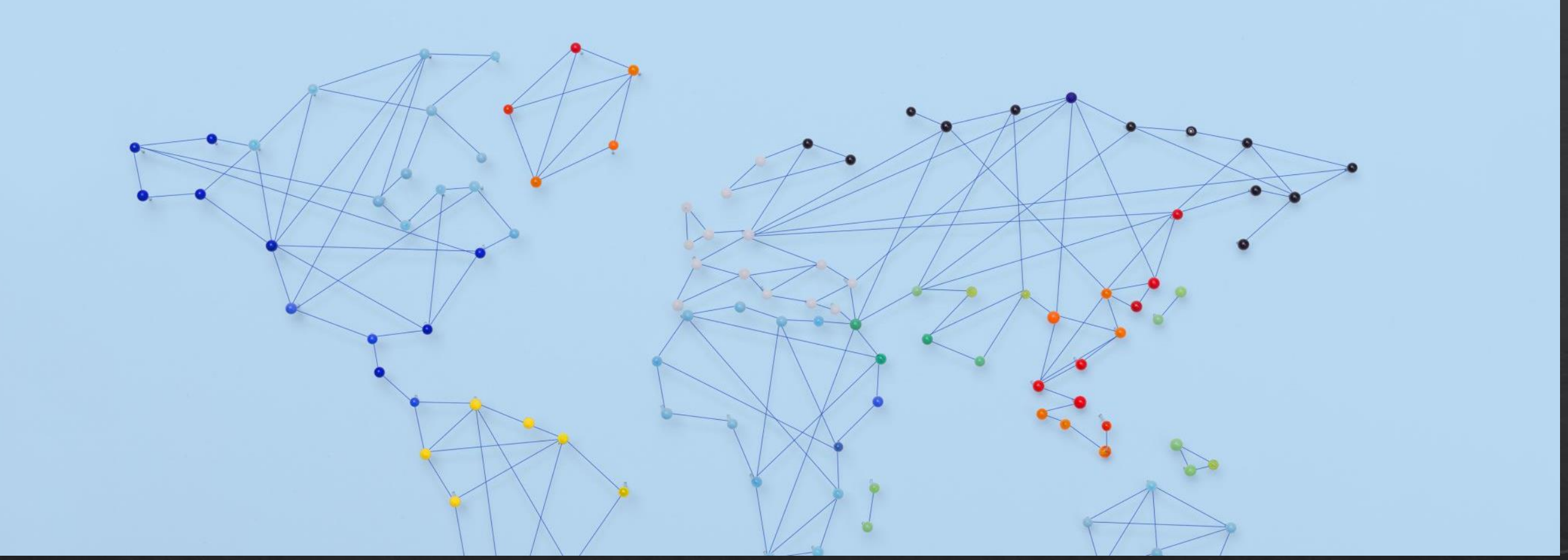
```
Select Landing__Outcome,  count(*) as Count  FROM SPACEXTBL
WHERE  DATE BETWEEN '2010-06-04' AND '2017-03-20'
AND ( Landing__Outcome = 'Failure (drone ship)' OR Landing__Outcome = 'Success (ground pad)')

GROUP BY Landing__Outcome
ORDER BY Count DESC
```

A double statement of the date between 2010/06/04 and 2017/03/20 and the type of LANDING_OUTCOME was set to be able to find the number of Landing outcomes during that date.

SQL Results

LANDING__OUTCOME	COUNT
Failure (drone ship)	5
Success (ground pad)	3



Launch Sites Proximities Analysis (Folium interactive map)

Folium map

- In the Folium map can be spotted the SpaceX Launch Sites, which are highlighted in red on the Folium map

Figure 6

GitHub Notebook

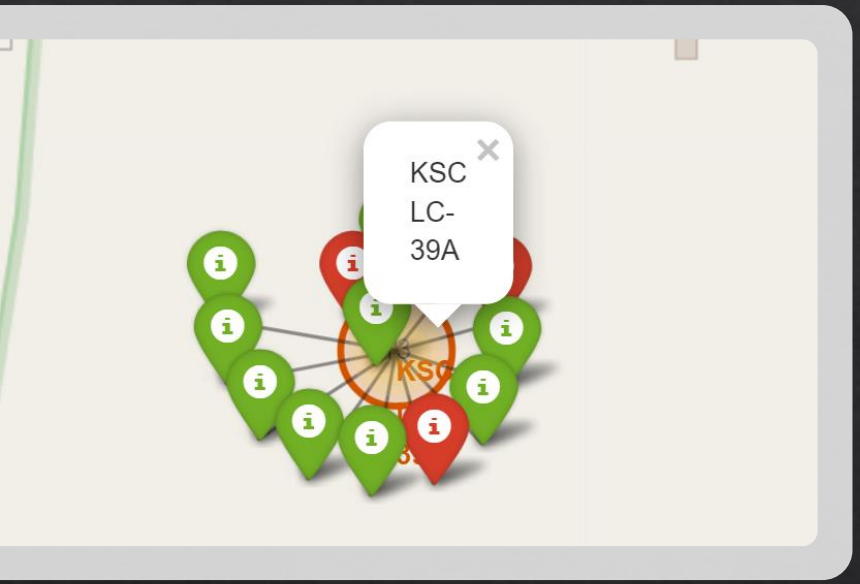


Figure 7

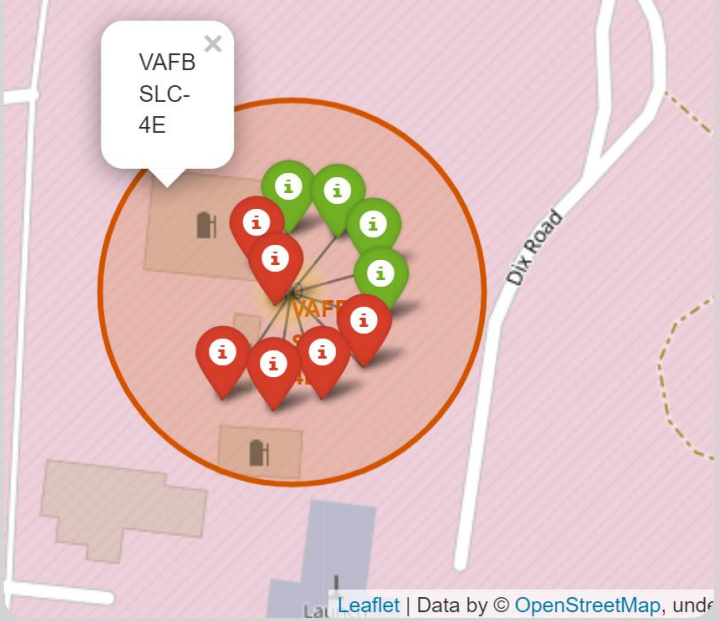


Figure 8

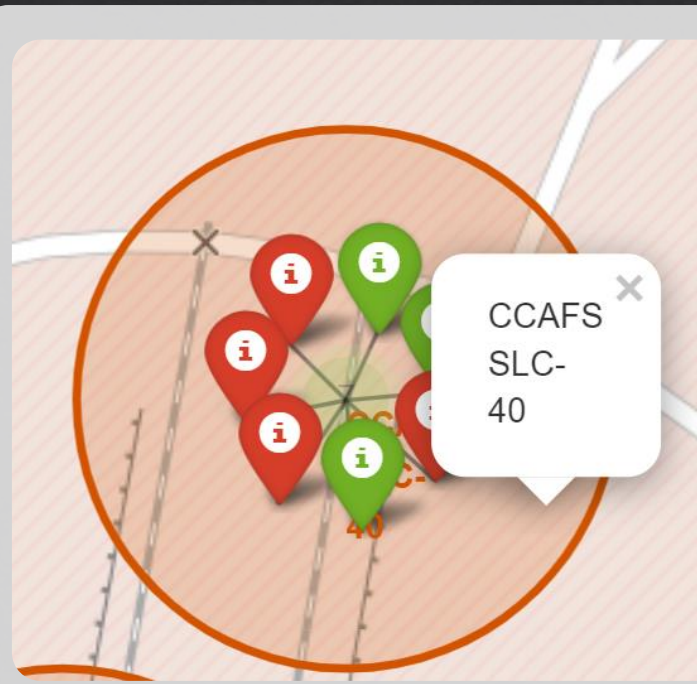


Figure 10

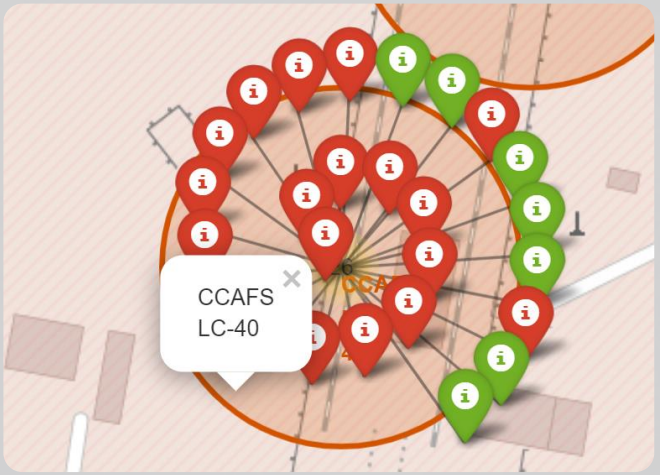


Figure 9

GitHub Notebook

Folium map (Successful/Failure Launches)

The green marks represent the Successful Launches and the red marks represent the Failure Launches

Figure 11

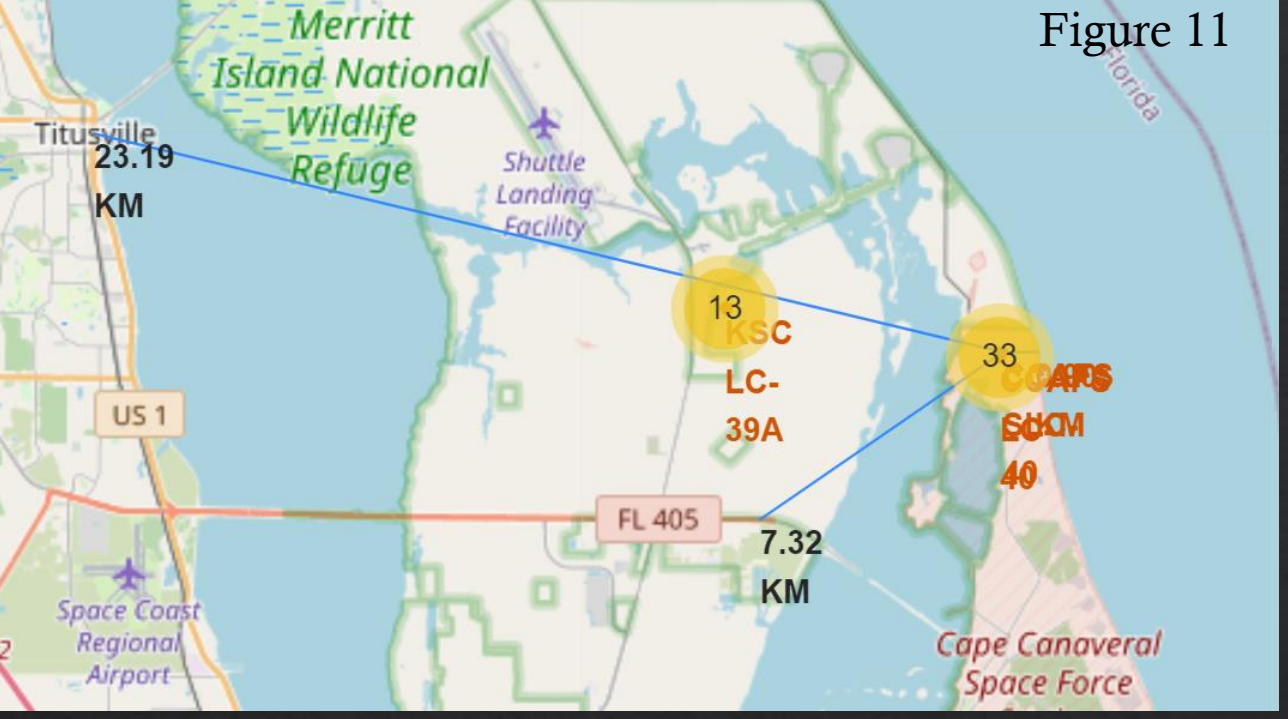
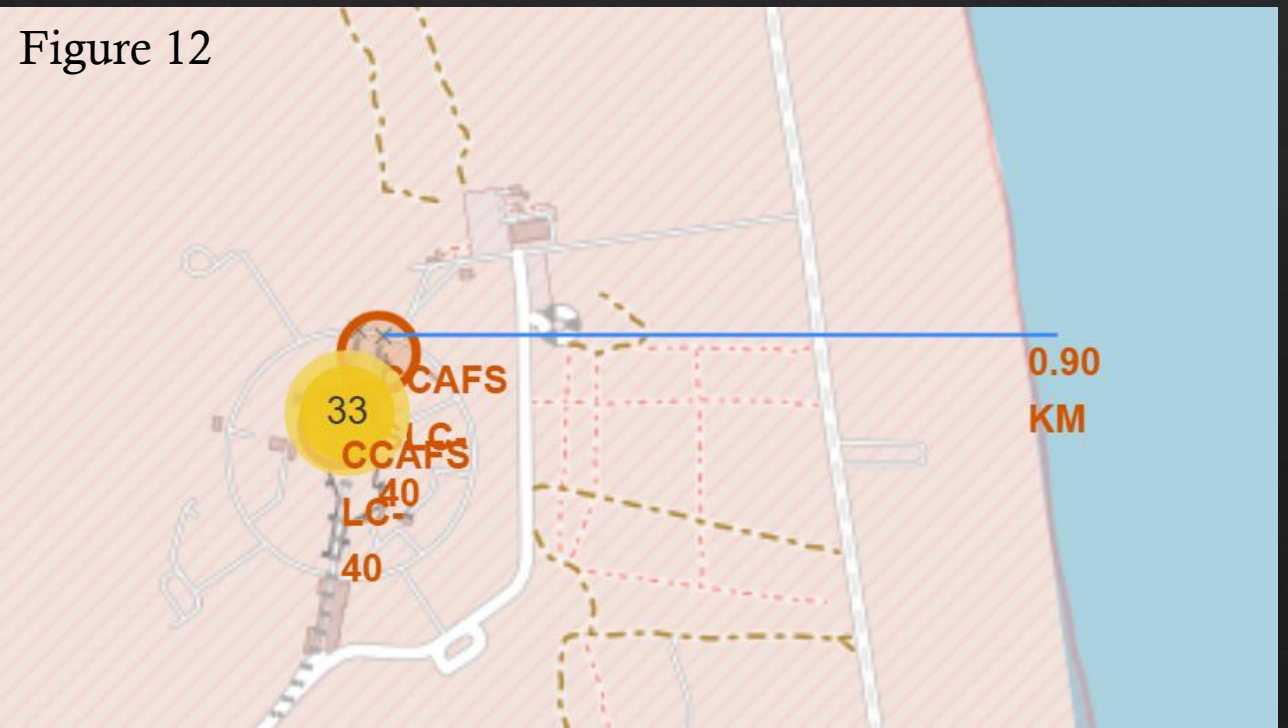


Figure 12



Closest city, coast and highway to the CCAFS SLC-40 Launch site

- Closest highway distance: 7.32 KM
- Closest City's (Titusville) distance:23.19 KM
- Closest coast distance: 0.90 KM

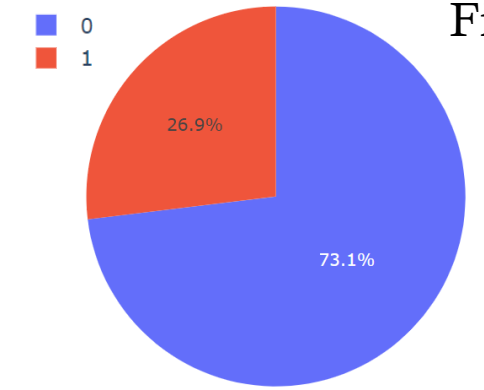
GitHub Notebook


Build a Dashboard with Plotly Dash

1

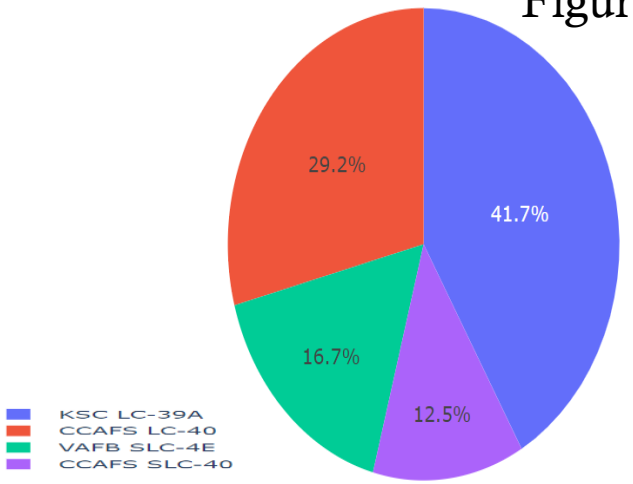
Launch Sites Pie Charts

Figure 13



Total Success Launches for site CCAFS LC-40

Figure 14



2

Figure 1. Figure 1 depicts the Success/Failure rate for the Launch site CCAFSLC-40 (1: Success Rate, 2: Failure Rate).

Figure 2. Figure 2 depicts the percentage of launches of all Launch Sites.

GitHub Notebook

Figure 15

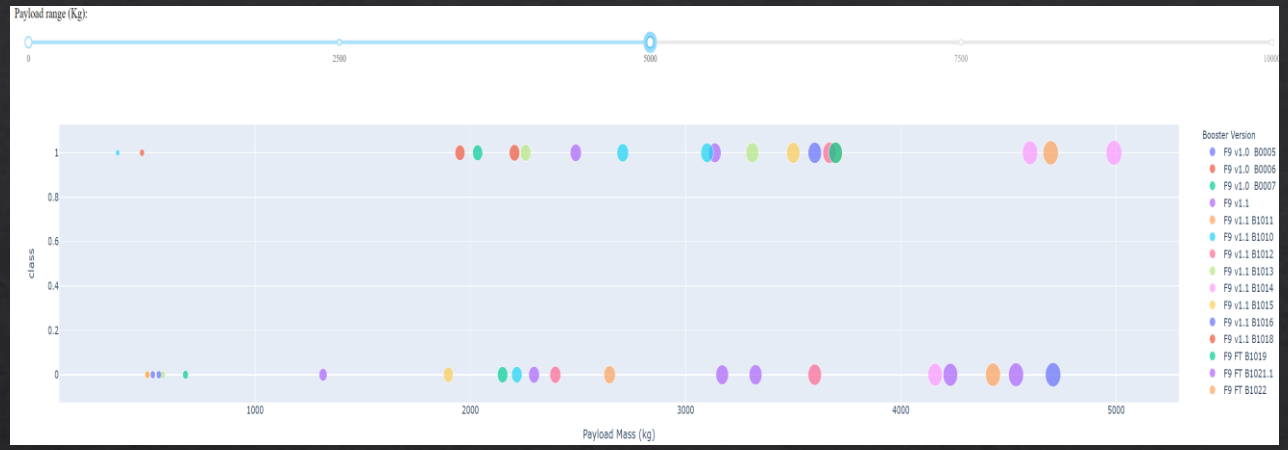
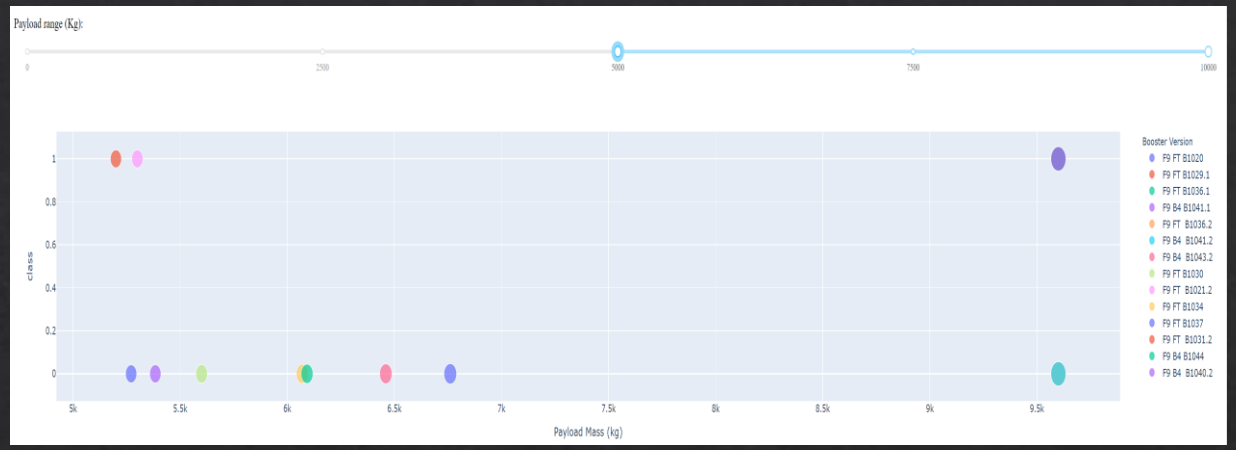


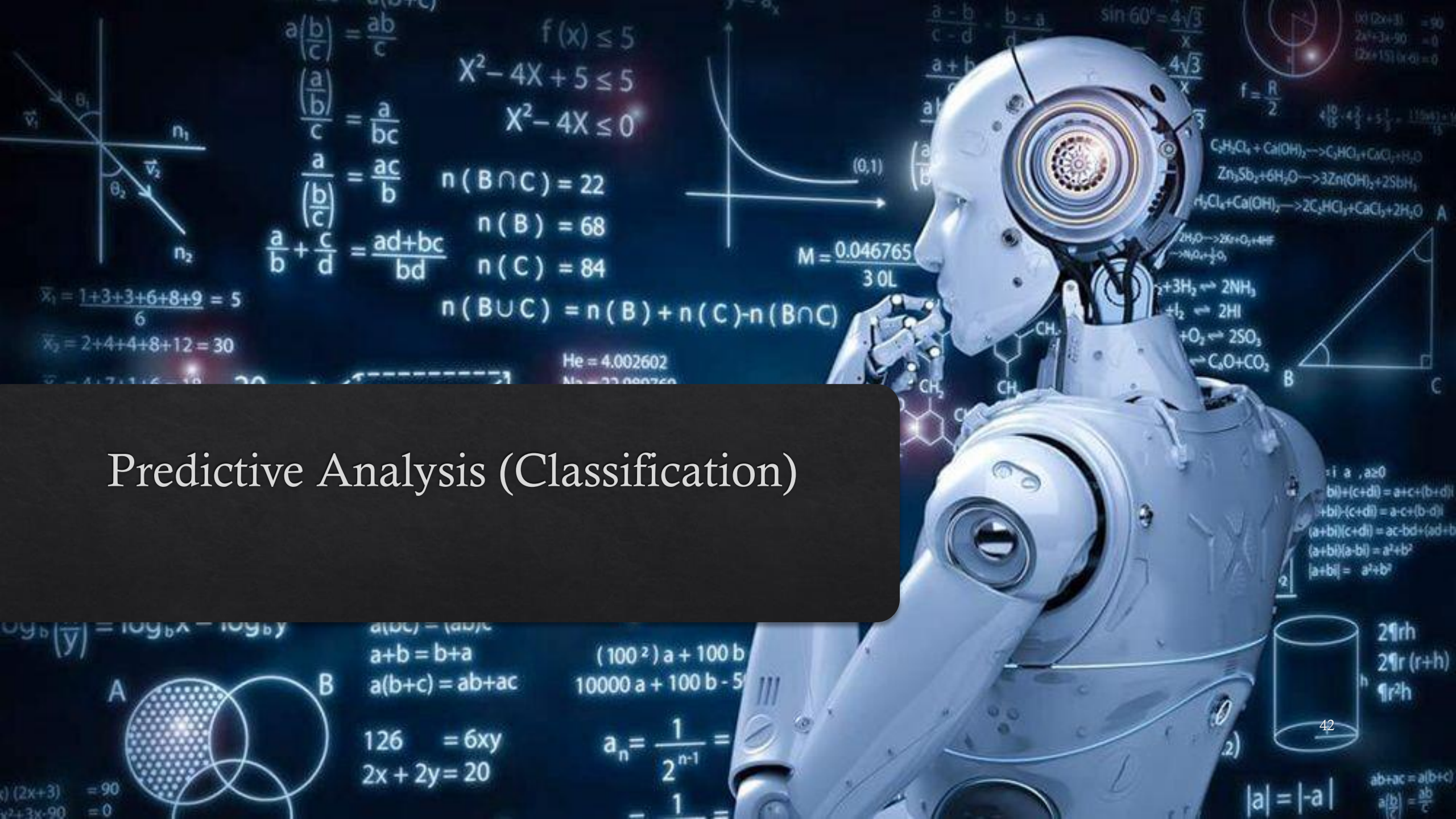
Figure 16



Success Rate vs Pay Load Mass Scatter plot and Slider

- Lighter Payload Mass (Figure 15) shows a higher success rate than heavier Load Mass (Figure 16).

GitHub Notebook



Predictive Analysis (Classification)

Classification Accuracy

- In Figure 5, the highest accuracy machine learning method to determine the likelihood of the success/failure Falcon 9 landings was the decision tree method (Figure 6).

- -Furthermore, Decision tree Test Data accuracy was approximately 78%.

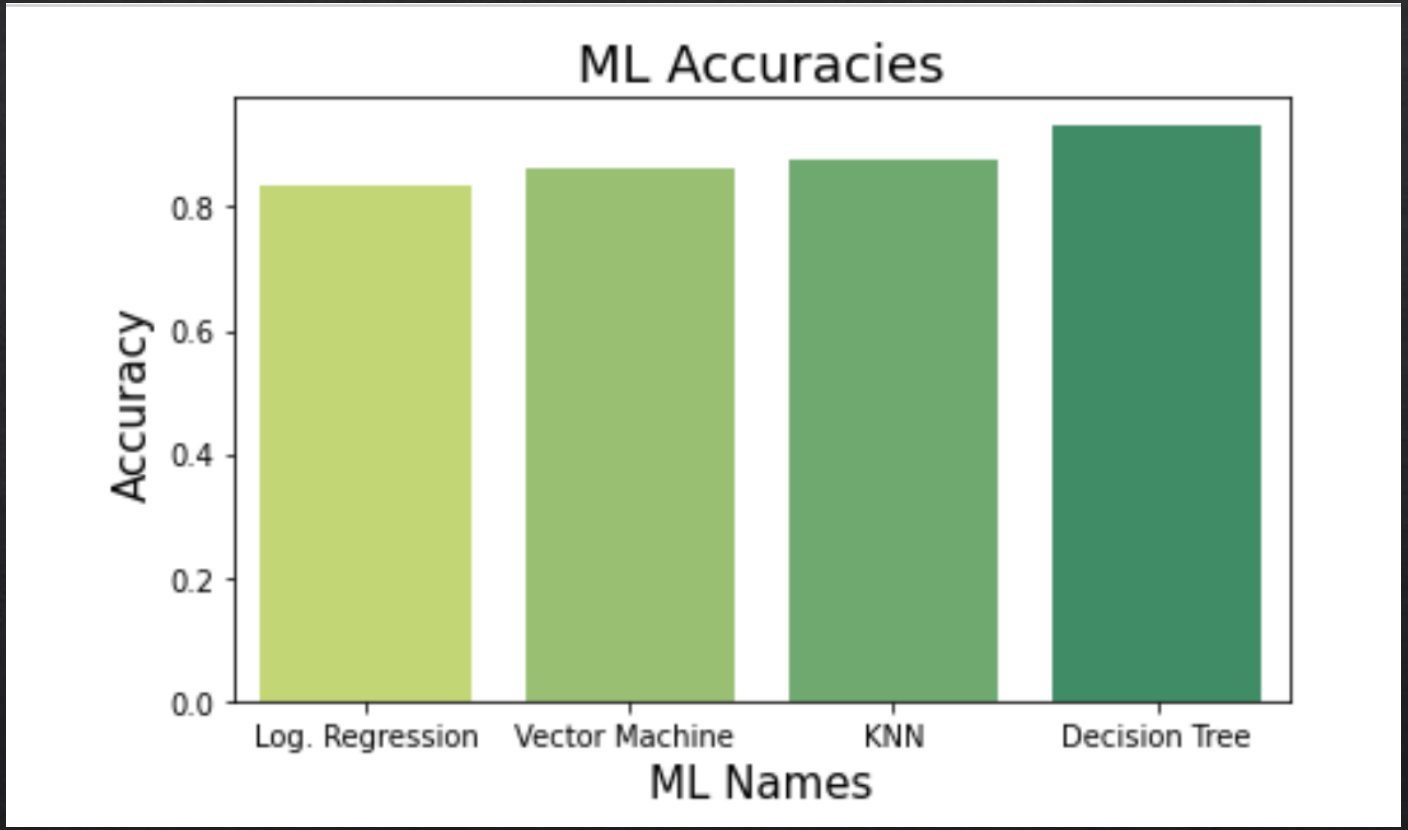


Figure 17

GitHub Notebook

Confusion Matrix (Decision Tree)

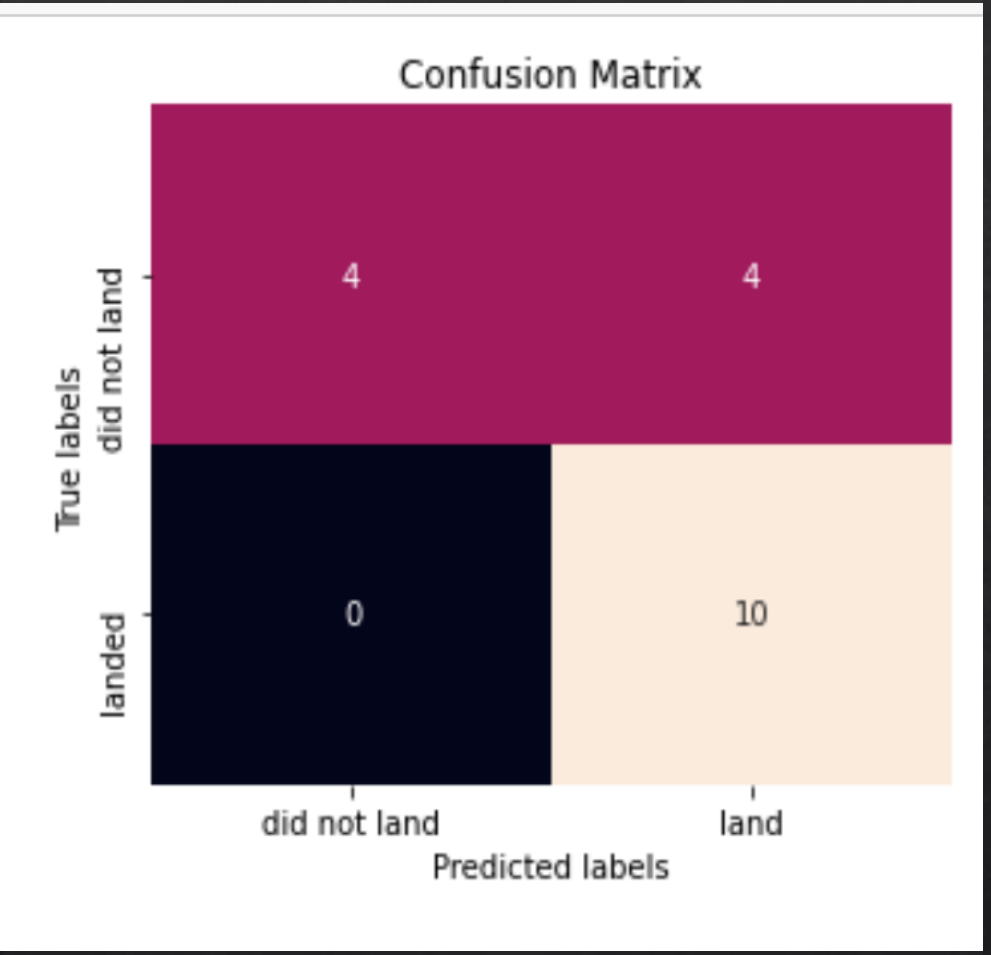
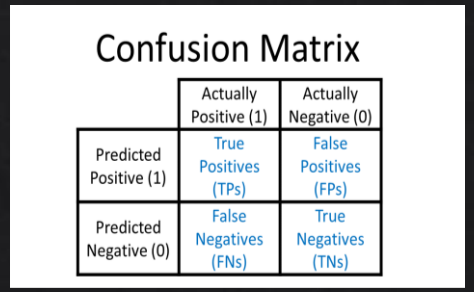
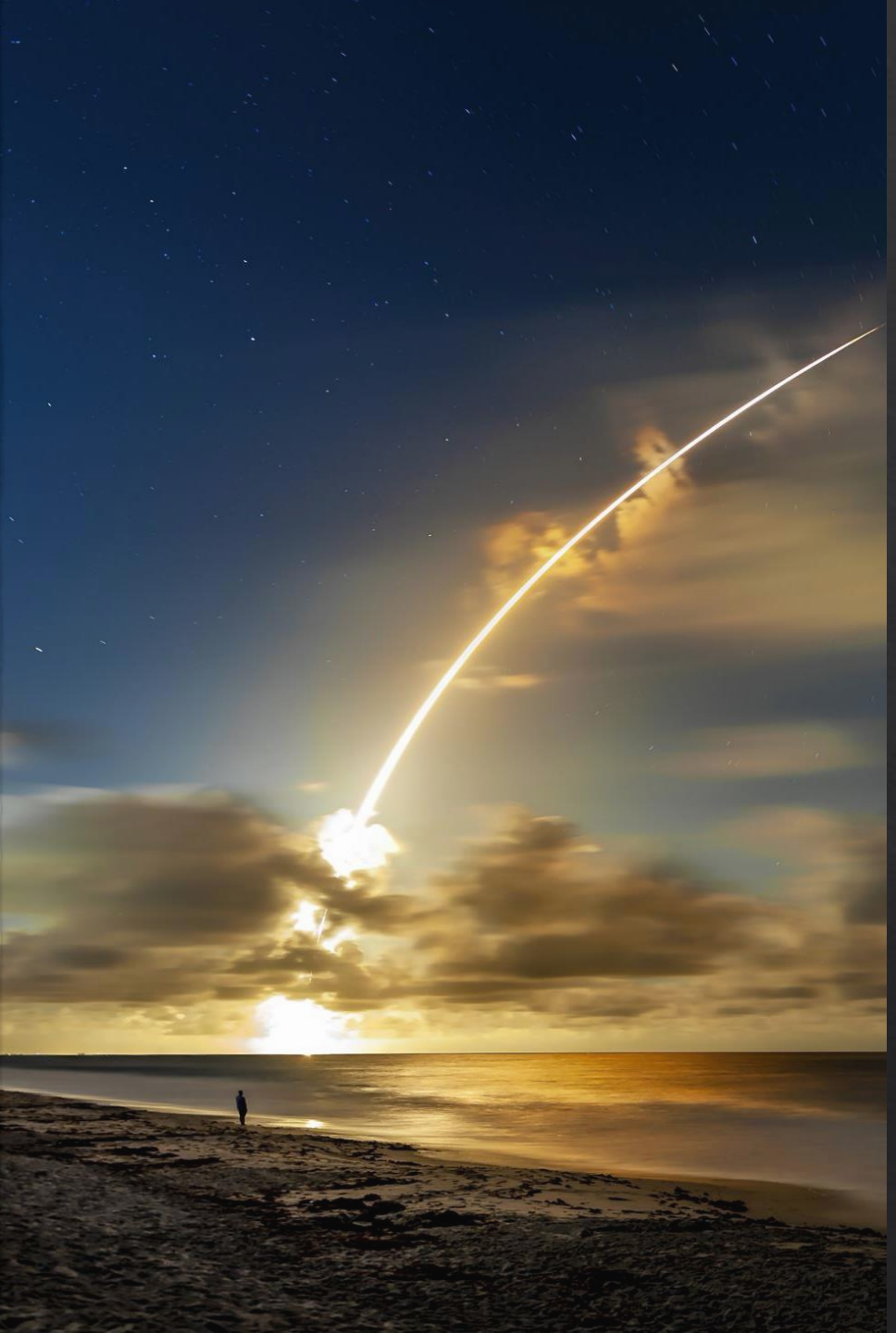


Figure 18

GitHub Notebook

- The Decision Tree Confusion Matrix depicts 10 predicted landings that actually landed, 4 that predicted landings that did not land but landed and 4 that did not (Figure 7).





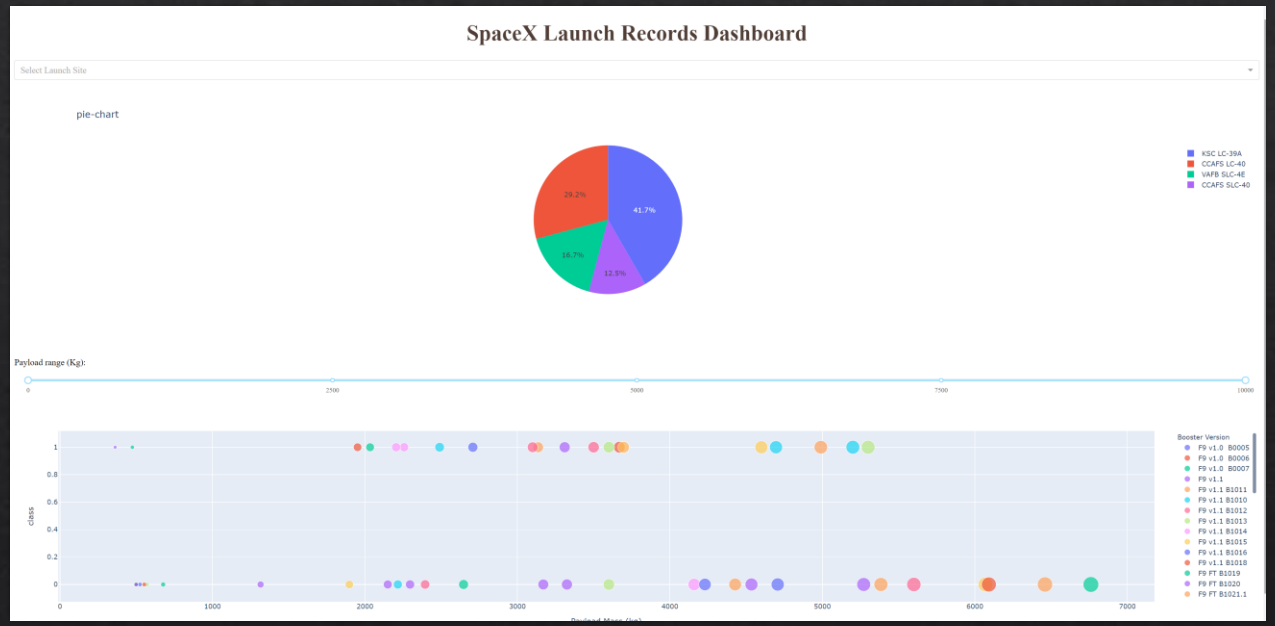
Conclusions

- Certainly, the Mass Payload is one of the factors with the most influence over the success/failure rate of the Falcon 9 landings. In the different Launch sites it can be observed that the Mass Payload played an important role for the success of the rocket landings. For instance, the rocket Mass payload might have been one of the main factors that may have caused such a low success rate of landings in the CCAFS LC-40 Launch Site, and in contrast, KSC LC-39A launch Site high Mass Payload may have boosted the success rate of landings.

- The best ML method to predict a successful /failure landing rockets is decision tree with almost a 93% of accuracy.

Appendix

Space X Plotly Dashboard



[GitHub Notebook]

Calculate_distance function between Launch site and a specific coordinate in the map (Folium Map)

```python
from math import sin, cos, sqrt, atan2, radians

def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance
```

Distance from nearest city (TITUSVILLE) to CCAFS SLC-40 Launch site (Python code, Folium Map)

```python
#Distance to TITUSVILLE

coordinates = [
    [28.56342, -80.57674],
    [28.6122, -80.8076]]

lines=folium.PolyLine(locations=coordinates, weight=1)
site_map.add_child(lines)
distance = calculate_distance(coordinates[0][0], coordinates[0][1], coordinates[1][0], coordinates[1][1])
distance_circle = folium.Marker(
    [28.6122, -80.8076],
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#252526;"><b>%s</b></div>' % "{:10.2f} KM".format(distance),
        )
    )
site_map.add_child(distance_circle)
site_map
```

GitHub Notebook

Thank you