IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<VODOUMBO Phillippe Miracle Towanou>

<17 Juin 2023>

Outline

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

Executive Summary

The goal of this capstone project is to predict if the Falcon 9 first stage will land successfully. In did, SpaceX 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 SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch.

- Summary of methodologies

We have adopted the following methodology

- ✓ Collect data through API and Web scraping
- ✓ Transform data through data wrangling
- ✓ Conduct exploratory data analysis with SQL and data visuals
- ✓ Build an interactive map with folium to analyze launch site proximity

Executive Summary

- ✓ Build a dashboard to analyze launch records interactively with Plotly Dash
- ✓ Finally, build a predictive model to predict if the first stage of Falcon 9 will land successfully

- Summary of all results

The results of our analyses will then be presented in the form of :

- ✓ Data analysis results
- ✓ Data visuals, interactive dashboards
- ✓ Predictive model analysis results

Introduction

- Project background and context
  - ✓ Companies such as Virgin Galactic Rocket Lab Blue Origin SpaceX are making space travel affordable for everyone.
  - ✓ The most successful is SpaceX which sending spacecraft to the International Space Station.
  - ✓ Cost of 62 million dollars for SpaceX versus 165 million dollars each for other providers.
  - ✓ For which reasons? The rocket launches are relatively inexpensive, and that SpaceX can reuse the first stage.
- Problems you want to find answers
  - ✓ Determine if the first stage will land and then the cost of each launch. determine the price of each launch.
  - ✓ Impact of different parameters/variables on the landing outcomes (e.g., launch site, payload mass, booster version, etc.)

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - ✓ SpaceX API
  - ✓ Web scrap Falcon 9 and Falcon Heavy launch records from Wikipedia (link)
- Perform data wrangling
  - ✓ Determined labels for training the supervised models by converting mission outcomes in to training labels (0-unsuccessful, 1-successful)
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - ✓ Created a column for 'class'; standardized and transformed data; train/test split data; find best classification algorithm (Logistic regression, SVM, decision tree, & KNN) using test data

Data Collection

Data collection is the process of gathering data from available sources. This data can be structured, unstructured, or semi-structured. For this project, data was collected via SpaceX API and Web scrapping Wiki pages for relevant launch data.

- Used SpaceX REST API to gather data on rocket launches:

  https://api.spacexdata.com/v4

- Falcon 9 launch data was also collected via Webscraping Wikipedia using

BeautifulSoup from the page below:

https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

- We also use the API to get information about the launches using the IDs given for each launch. Specifically we have using columns rocket, payloads, launchpad, and cores.

Data Collection - SpaceX API

1-API request and read request into DataFrame → 2-Declare global variables → 3-Call helper functions with API calls to populate global vars → 4-Construct dataset → 5-Filter the dataframe by only keeping the Falcon 9 launches

1-API request and read request into DataFrame

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

response = requests.get(spacex_url)

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

2-Declare global variables

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

3-Call helper functions with API calls to populate global vars

getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data

4-Construct dataset

```
# Create a data from launch_dict
df_launch = pd.DataFrame(launch_dict)
```

5-Filter the dataframe by only keeping the Falcon 9 launches

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df_launch[df_launch['BoosterVersion']!= 'Falcon 1']
```

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

GITHUB

Data Collection - Scraping

1-Perform an HTTP GET method → 2-Create a BeautifulSoup → 3-Find all tables on the wiki page and extract column name → 4-Create an dictionary with keys from the extracted column names → 5-Call helper functions to fill up dict with launch records → 6-Create a dataframe from laucnh dict

1-Perform an HTTP GET method

```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url).text
```

2-Create a BeautifulSoup

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data,"html.parser")
```

3-Find all tables on the wiki page and extract column name

```
column_names = []

# Apply find_all() function with `th` element on fir
# Iterate each th element and apply the provided ext
# Append the Non-empty column name (`if name is not
colnames = first_launch_table.find_all('th')
for x in range(len(colnames)):
    name2 = extract_column_from_header(colnames[x])
    if (name2 is not None and len(name2) > 0):
        column_names.append(name2)
```

4-Create an dictionary with keys from the extracted column names

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each
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']=[]
```

5-Call helper functions to fill up dict with launch records

```
def date_time(table_cells):
def booster_version(table_cells):
def landing_status(table_cells):
def get_mass(table_cells):
```

6-Create a dataframe from laucnh dict

```
df=pd.DataFrame(launch_dict)
```

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

GITHUB

Data Wrangling

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident;

- True Ocean means the mission outcome was successfully landed to a specific region of the
- ocean
- False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean.
- True RTLS means the mission outcome was successfully landed to a ground pad
- False RTLS means the mission outcome was unsuccessfully landed to a ground pad.
- True ASDS means the mission outcome was successfully landed on a drone ship
- False ASDS means the mission outcome was unsuccessfully landed on a drone ship.

In the following slide, we'll take you through the various stages.

Data Wrangling

1. Load dataset in to Dataframe

```
resp = await fetch(URL)
dataset_part_1_csv = io.BytesIO((await resp.arrayBuffer()).to_py())
```

```
df=pd.read_csv(dataset_part_1_csv)
df.head(10)
```

3. Create landing outcome label

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

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

2. Find patterns in Data
   - Missing Values

```
df.isnull().sum()/df.shape[0]*100
```

   - Data Types

```
df.dtypes
```

   - Various launch sites, orbit and landing outcomes

```
# Apply value_counts() on column
df['LaunchSite'].value_counts()
```

```
# Apply value_counts on Orbit
df['Orbit'].value_counts()
```

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
```

   - Succes rate

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

GITHUB

EDA with Data Visualization

- As part of the Exploratory Data Analysis (EDA), following charts were plotted to gain further insights into the dataset:

1.Scatter plot:

- •Shows relationship or correlation between two variables making patterns easy to observe
- •Plotted following charts to visualize:
- •Relationship between Flight Number and Launch Site
- •Relationship between Payload and Launch Site
- •Relationship between Flight Number and Orbit Type
- •Relationship between Payload and Orbit Type

2.Bar Chart:

- •Commonly used to compare the values of a variable at a given point in time. Barcharts makes it easy to see which groups are highest/common and how other groups compare against each other. Length of each bar is proportional to the value of the items that it represents
- •Plotted following Bar chart to visualize:
- •Relationship between success rate of each orbit type

3.Line Chart:

- •Commonly used to track changes over a period of time. It helps depict trends over time.
- •Plotted following Line chart to observe:
- •Average launch success yearly trend

GITHUB

EDA with SQL

Used SQL Queries to an IBM DB2 instance to gain insight on the dataset.

Desired Insight:

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string 'CCA'
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome in ground pad was achieved
6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. List the total number of successful and failure mission outcomes
8. List the names of the booster versions which have carried the maximum payload mass
9. List the failed landing_outcomesin drone ship, their booster versions, and launch site names for in year 2015
10. 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.

GITHUB

Build an Interactive Map with Folium

Folium interactive map helps analyze geospatial data to perform more interactive visual analytics and better understand factors such location and proximity of launch sites that impact launch success rate. Following map object were created and added to the map:

- Mark all launch sites on the map. This allowed to visually see the launch sites on the map.
  - Added 'folium.circle' and 'folium.marker' to highlight circle area with a text label over each launch site.
- Added a 'MarkerCluster()' to show launch success (green) and failure (red) markers for each launch site.
- Calculated distances between a launch site to its proximities (e.g., coastline, railroad, highway, city)
  - Added 'MousePosition() to get coordinate for a mouse position over a point on the map
  - Added 'folium.Marker()' to display distance (in KM) on the point on the map (e.g., coastline, railroad, highway, city)
  - Added 'folium.Polyline()' to draw a line between the point on the map and the launch site
- Repeated steps above to add markers and draw lines between launch sites and proximities –coastline, railroad, highway, city)
- Building the Interactive Map with Folium helped answered following questions:
  - Are launch sites in close proximity torailways? YES
  - Are launch sites in close proximity tohighways? YES
  - Are launch sites in close proximity tocoastline? YES
  - Do launch sites keep certain distance away from cities? YES

GITHUB

Build a Dashboard with Plotly Dash

- Built PlotlyDashboard to make an interactive web app to visualize launch data
- Includes Pie Charts to visualize launch landing success broken down by Launch Site
  - If all sites are selected we get the proportion successful launch landings each site accounts for
  - If we select an individual site we see the proportion of all launches at that site which landed successfully
- Includes Scatter Plot of Payload Mass (kg) vs Landing Success Rating (0 for Failure, 1 for Success) color coded by booster version:
  - Selecting a single site removes points from other sites, Selecting All includes all data
  - Plot Payload Mass Range can be selected by a slider for Min and Max values

•Dashboard helped answer following questions:

- Which site has the largest successful launches? KSC LC-39A with 10
- Which site has the highest launch success rate? KSC LC-39A with 76.9% success
- Which payload range(s) has the highest launch success rate? 2000 –5000 kg
- Which payload range(s) has the lowest launch success rate? 0-2000 and 5500 -7000
- Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest launch success rate? FT

GITHUB

Predictive Analysis (Classification)

1-Create a NumPy array from the column Class in data → 2-Standardize the data in X → 3-Split the data X and Y into training and test data → 4-Create and Refine Models → 5-Find the best performing Model

1- Create a NumPy array from the column Class in data

```
Y = data['Class'].to_numpy()
```

2- Standardize the data in X

```
# students get this
X= preprocessing.StandardScaler().fit(X).transform(X)
```

3-Split the data X and Y into training and test data

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

4-Create and Refine Models

- Logistic Regression
- Support Vector Machine
- KNN
- Decision Tree

5-Find the best performing Model

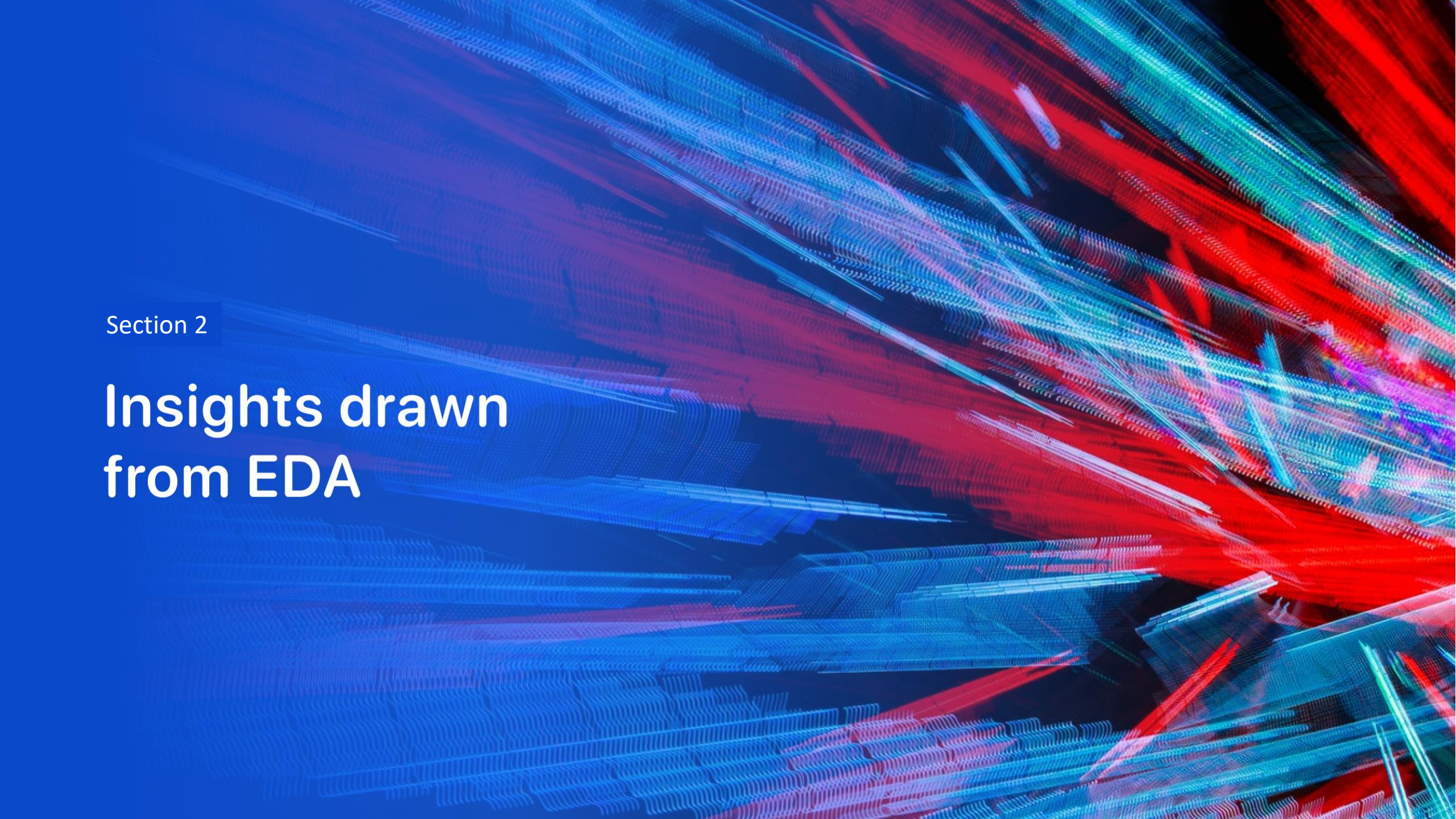
```
Model_Performance_df = pd.DataFrame({'Algo Type': ['Logistic Regression', 'SVM','Decision Tree','KNN'],
'Accuracy Score': [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_],
'Test Data Accuracy Score': [logreg_cv.score(X_test, Y_test), svm_cv.score(X_test, Y_test),
tree_cv.score(X_test, Y_test), knn_cv.score(X_test, Y_test)]})
Model_Performance_df.sort_values(['Accuracy Score'], ascending = False, inplace=True)
Model_Performance_df
```

GITHUB

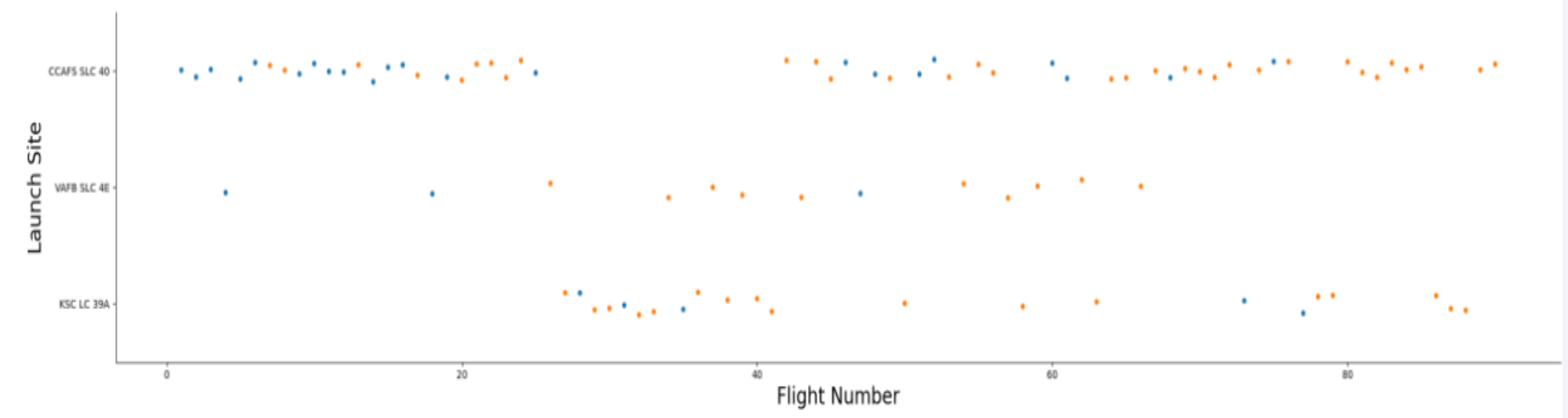
Results

Following sections and slides explain results for:

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


Section 2
Insights drawn from EDA

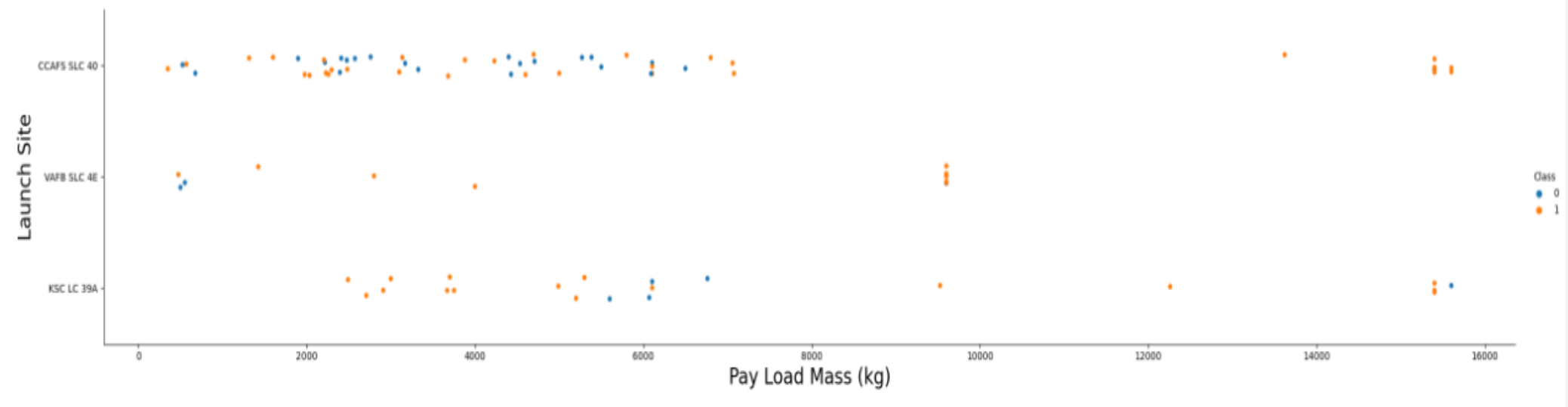
Flight Number vs. Launch Site



- Succes rate of Launch increase as the number of flight increase
- For launch site « KSC LC 39A », it takes at least around 25 launches before a first successful launch

GITHUB

Payload vs. Launch Site

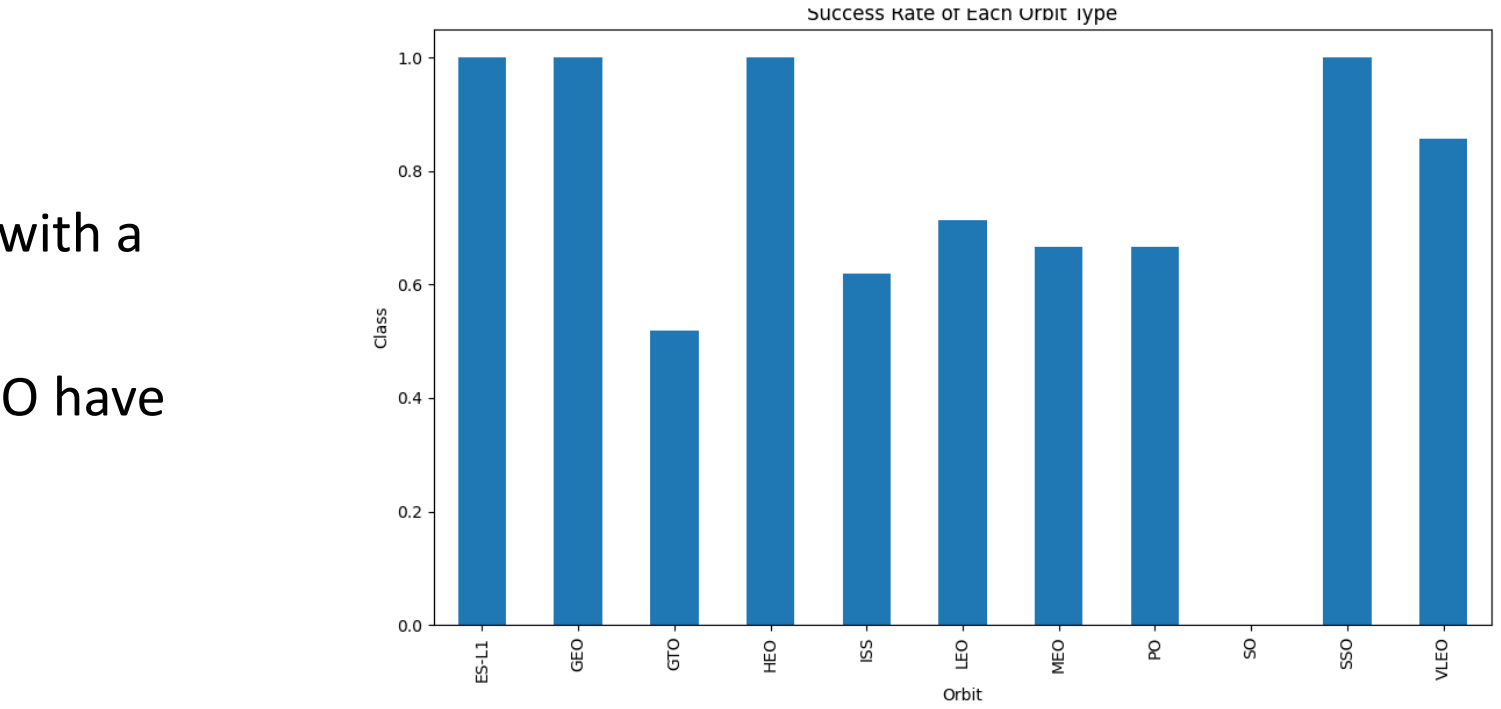


- Percentage of successful launch (Class=1) increases for launch site ‘VAFB SLC 4E’ as the payload mass increases
- For launch site ‘VAFB SLC 4E’, there are no rockets launched for payload greater than 10,000 kg

GITHUB

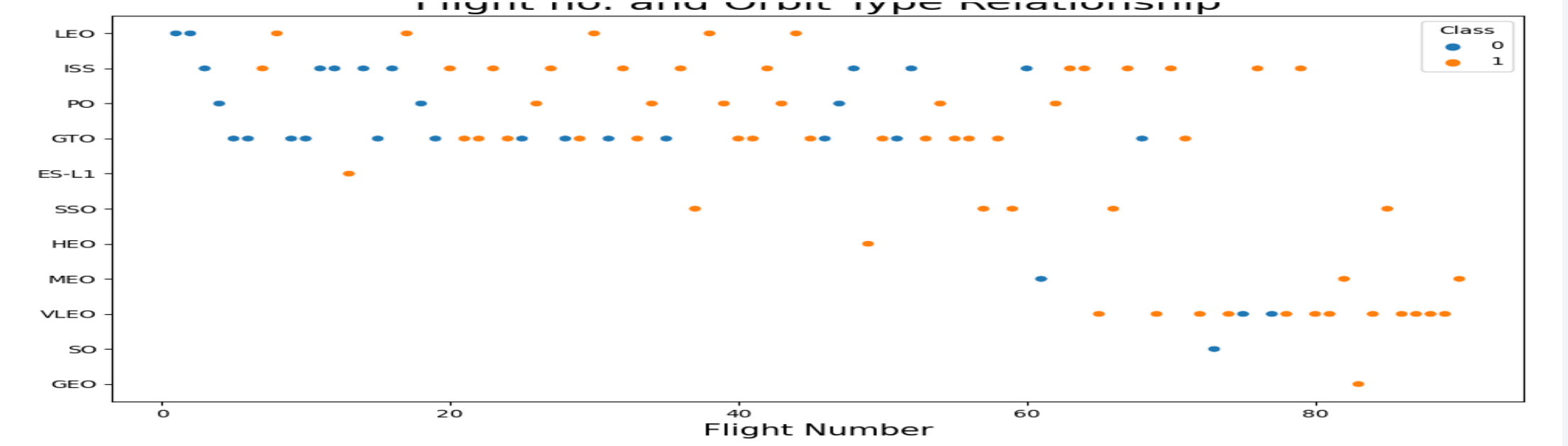
Success Rate vs. Orbit Type

- The GTO orbit is the only one with a success rate of less than 60%.
- Orbits ES LI, GEO, HEO, and SSO have never failed.



GITHUB

Flight Number vs. Orbit Type

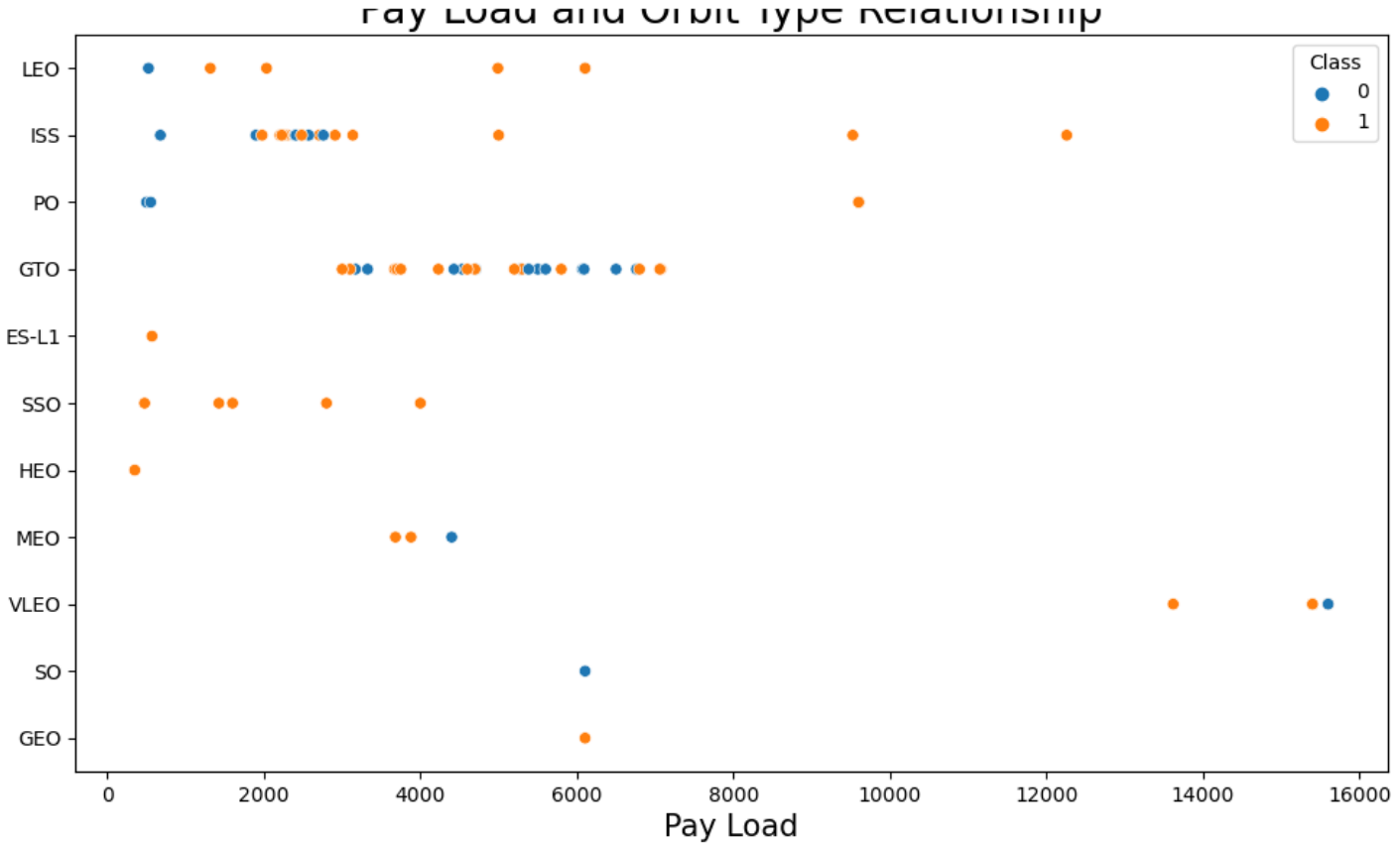


- For orbit VLEO, first successful landing (class=1) doesn't occur until 60+ number of flights
- For most orbits (LEO, ISS, PO, SSO, MEO, VLEO) successful landing rates appear to increase with flight numbers
- There is no relationship between flight number and orbit for GTO

GITHUB

Payload vs. Orbit Type

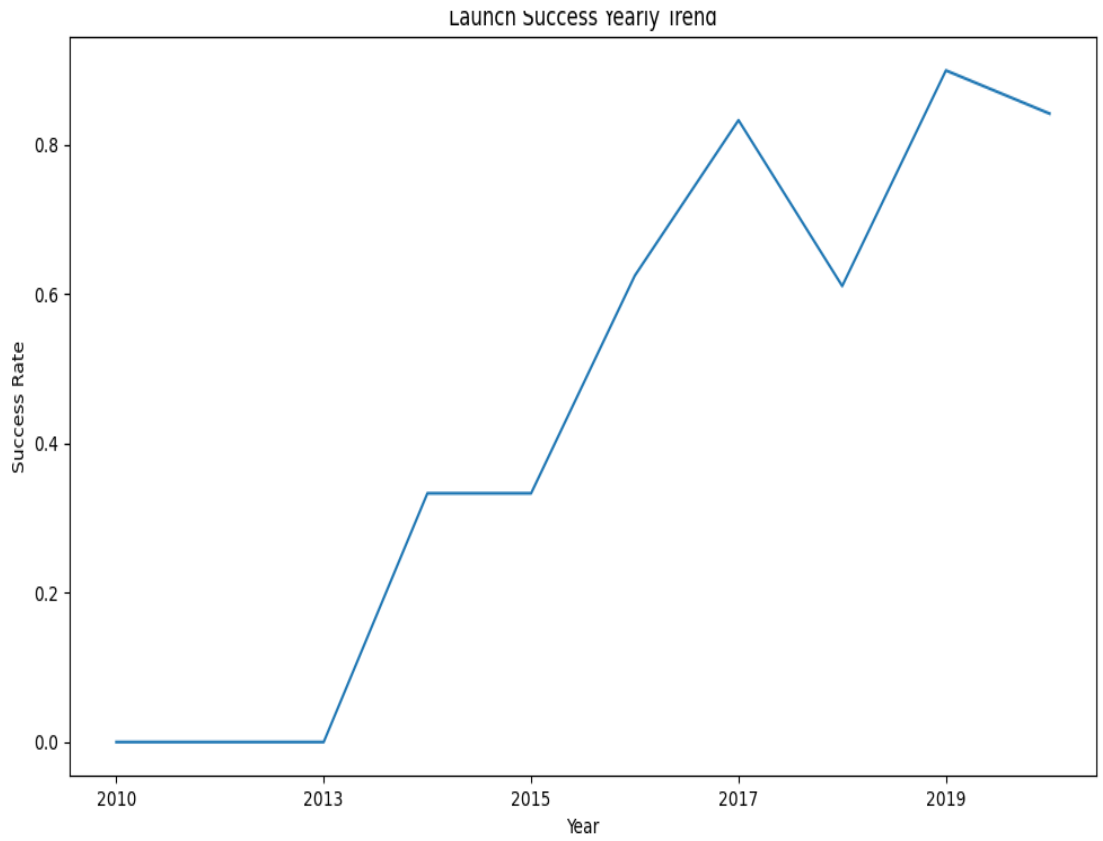
- The most of orbit have an payload mass less than 8000
- There are not clear relationship between PayLoad and Orbit Type



GITHUB

Launch Success Yearly Trend

- Prior to 2013, there were no successful launches
- Since 2013, the success rate has risen considerably, reaching over 80% in 2019.



GITHUB

All Launch Site Names

Query:

```
select distinct Launch_Site from spacextbl
```

Description:

- ”distinct” returns only unique values from the queries column (Launch_Site)
- There are 4 unique launch sites

Résultat:

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

GITHUB

Launch Site Names Begin with 'CCA'

Query:

```
select * from spacextbl where Launch_Site LIKE 'CCA%' limit 5;
```

Description:

- Using keyword ‘Like’ and format ‘CCA%’, returns records where ‘Launch_Site’ column starts with “CCA”.
- Limit 5, limits the number of returned records to 5

Resultats:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
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	Success	Failure (parachute)
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

GITHUB

Total Payload Mass

Query:

```
select sum(PAYLOAD_MASS__KG_) from spacextbl where Customer = 'NASA (CRS)'
```

Description:

- ‘sum’ adds column ‘PAYLOAD_MASS_KG’ and returns total payload mass for customers named ‘NASA (CRS)’

Resultats:

sum(PAYLOAD_MASS__KG_)
45596.0

GITHUB

Average Payload Mass by F9 v1.1

Query:

```
select avg(PAYLOAD_MASS__KG_) from spacextbl where Booster_Version LIKE 'F9 v1.1';
```

Description:

- 'avg' keyword returns the average of payload mass in 'PAYLOAD_MASS_KG' column where booster version is 'F9 v1.1'

Resultats:

avg(PAYLOAD_MASS__KG_)
2928.4

GITHUB

First Successful Ground Landing Date

Query:

```
select min(Date) as min_date from spacextbl where Landing_Outcome = 'Success (ground pad)';
```

Description:

- 'min(Date)' selects the first or the oldest date from the 'Date' column where first successful landing on group pad was achieved
- •Where clause defines the criteria to return date for scenarios where 'Landing_Outcome' value is equal to 'Success (ground pad)'

Resultats:

min_date
01/08/2018

GITHUB

Successful Drone Ship Landing with Payload between 4000 and 6000

Query:

```
select Booster_Version from spacextbl where (PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000)
and (Landing_Outcome = 'Success (drone ship)');
```

Description:

- The query finds the booster version where payload mass is greater than 4000 but less than 6000 and the landing outcome is success in drone ship
- The ‘and’ operator in the where clause returns booster versions where both conditions in the where clause are true

Resultats:

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

GITHUB

Total Number of Successful and Failure Mission Outcomes

Query:

```
select distinct Mission_Outcome, count(Mission_Outcome) as Total_Number from spacextbl group by Mission_Outcome;
```

Description:

- The ‘group by’ keyword arranges identical data in a column in to group
- In this case, number of mission outcomes by types of outcomes are grouped in column ‘counts’

Resultats:

Mission_Outcome	Total_Number
None	0
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

GITHUB

Boosters Carried Maximum Payload

Query:

```
select Booster_Version, PAYLOAD_MASS__KG_ from spacextbl where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacextbl);
```

Description:

- The sub query returns the maximum payload mass by using keywork ‘max’ on the pay load mass column
- The main query returns booster versions and respective payload mass where payload mass is maximum with value of 15600

Resultats:

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600.0
F9 B5 B1049.4	15600.0
F9 B5 B1051.3	15600.0
F9 B5 B1056.4	15600.0
F9 B5 B1048.5	15600.0
F9 B5 B1051.4	15600.0
F9 B5 B1049.5	15600.0
F9 B5 B1060.2	15600.0
F9 B5 B1058.3	15600.0
F9 B5 B1051.6	15600.0
F9 B5 B1060.3	15600.0
F9 B5 B1049.7	15600.0

GITHUB

2015 Launch Records

Query:

```
Select landing_outcome, booster_version, launch_site, substr(Date, 4, 2) as Month  from spacextbl
where landing_outcome = 'Failure (drone ship)' and substr(Date,7,4)='2015'
```

Description:

- The query lists landing outcome, booster version, the launch site and the month where landing outcome is failed in drone ship and the year is 2015
- The 'and' operator in the where clause returns booster versions where both conditions in the where clause are true
- The 'year' keywork extracts the year from column 'Date
- The results identify launch site as 'CCAFS LC-40' and booster version as F9 v1.1 B1012 and B1015 that had failed landing outcomes in drop ship in the year 2015

Resultats:

Landing_Outcome	Booster_Version	Launch_Site	Month
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	10
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	04

GITHUB

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

Query:

```
select Landing_Outcome, count(*) as LandingCounts from spacextbl where (Date between '04/06/2010' and '20/03/2017')
and (Landing_Outcome like 'Success%')
group by Landing_Outcome
order by count(*) desc;
```

Description:

- The ‘group by’ key word arranges data in column ‘Landing__Outcome’ into groups
- The ‘between’ and ‘and’ keywords return data that is between 2010-06-04 and 2017-03-20
- The ‘order by’ keyword arranges the counts column in descending order
- The result of the query is a ranked list of landing outcome counts per the specified date range

Resultats:

Landing_Outcome	LandingCounts
Success	20
Success (drone ship)	8
Success (ground pad)	7

GITHUB

Section 3

Launch Sites Proximities Analysis

SpaceX Falcon9 -Launch Sites Map

Figure 1:

Figure 2:

Figure 3

Figure 1 on left displays the Global map with Falcon 9 launch sites that are located inthe United States (in California and Florida). It is also evident that all launch sites are near the coast.
Figure 2 and Figure 3 zoom in to the launch sites to display 4 launch sites.

GITHUB

SpaceX Falon9 –Success/Failed Launch Map for all Launch Sites

- Figure 1,2, 3, and 4 zoom in to each site and displays the success/fail markers with green as success and red as failed
- By looking at each site map, KSC LC-39A Launch Site has the greatest number of successful launches

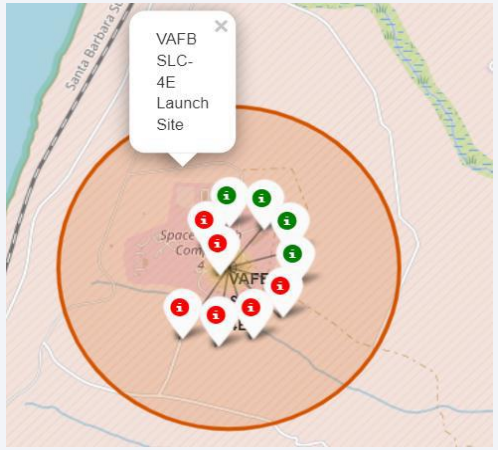


Figure 1 : VAFB

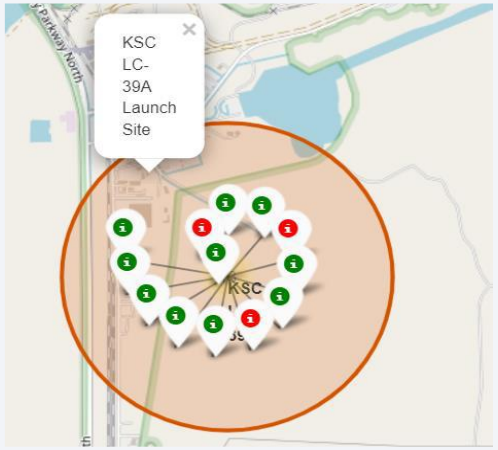


Figure 2: KSC LC-39A

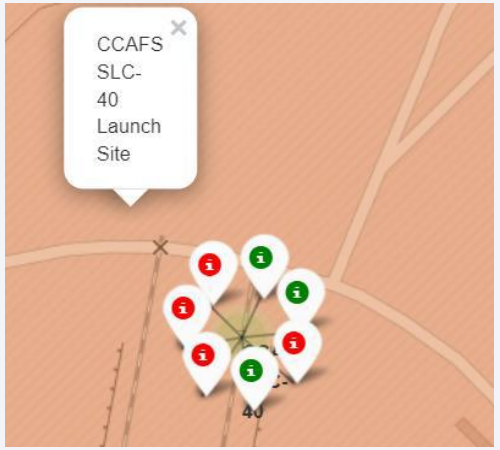


Figure 3: CCAFS SLC-40

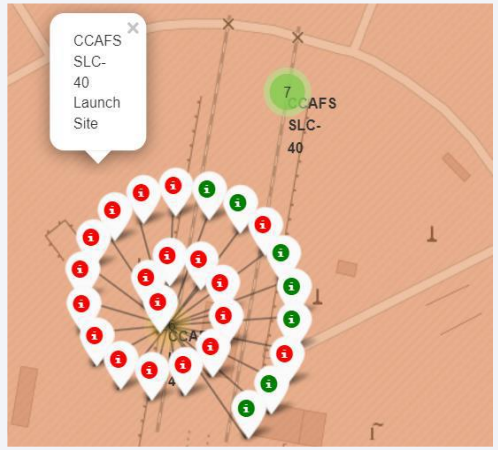
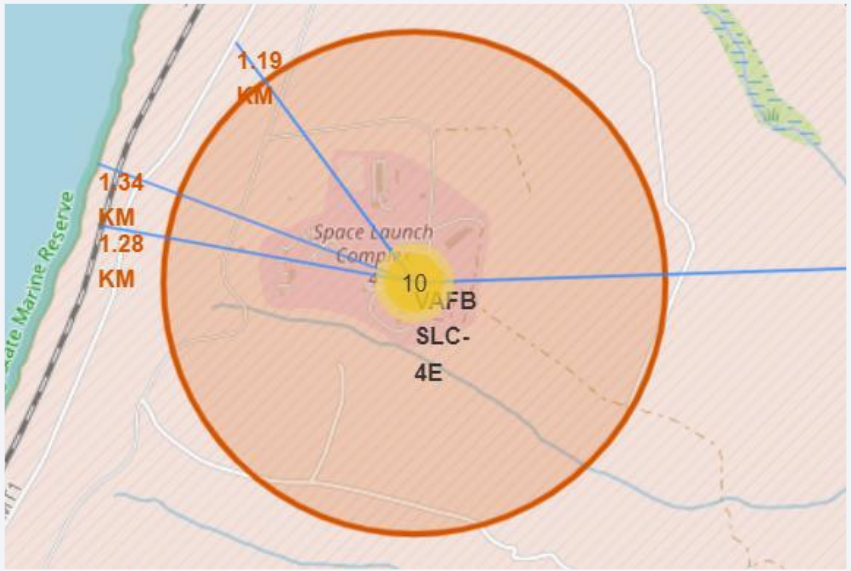


Figure 4 : CCAFS SLC 40

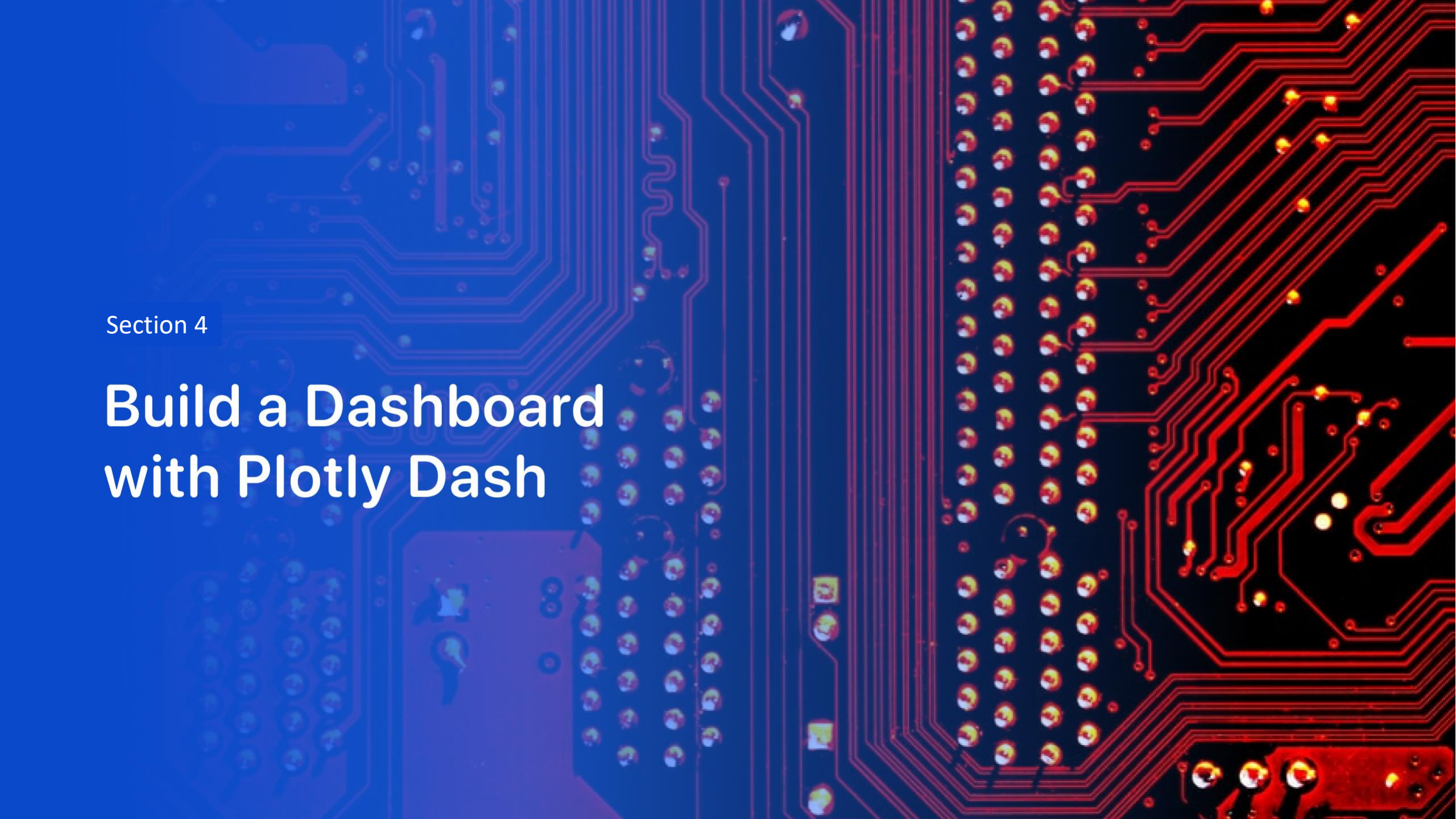
GITHUB

SpaceX Falcon9 –Launch Site to proximity Distance Map

Those figures displays all the proximity sites marked on the map for Launch Site VAFB SLC-4E. City Lompoc is located further away from Launch Site compared to other proximities such as coastline, railroad, highway, etc. The map also displays a marker with city distance from the Launch Site (14.09 km) In general, cities are located away from the Launch Sites to minimize impacts of any accidental impactsto the general public and infrastructure. Launch Sites are strategically located near the coastline, railroad, and highways to provide easy access to resources.



GITHUB

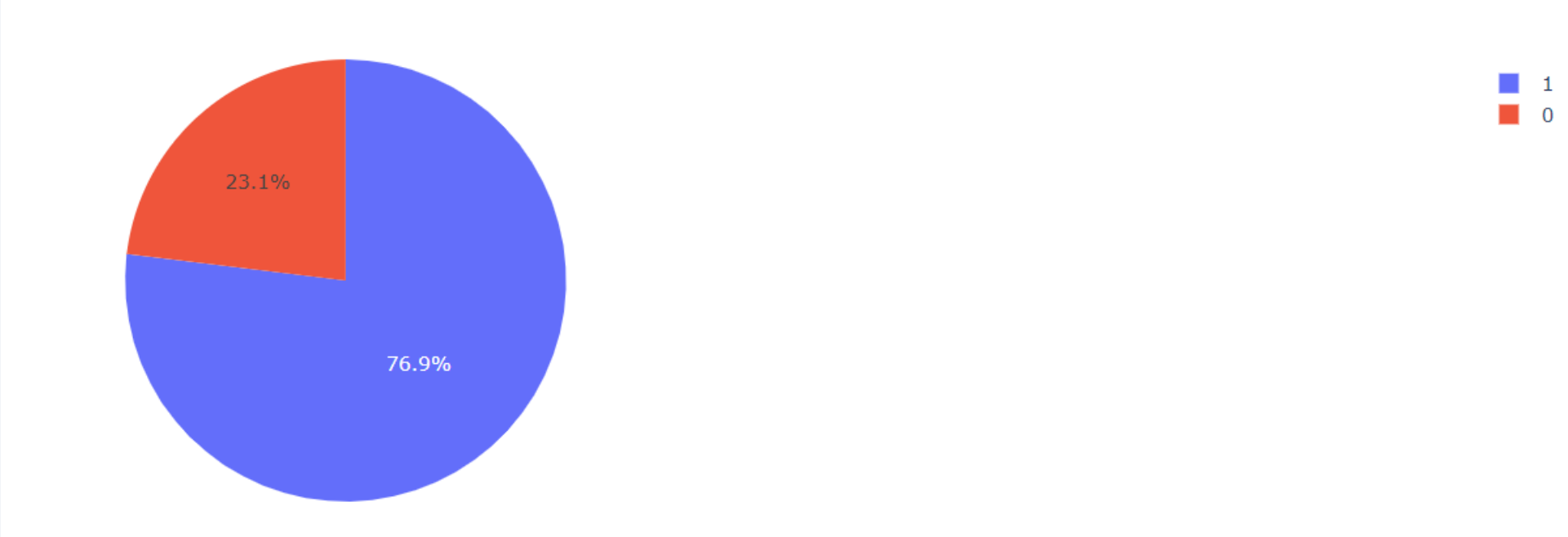

Section 4
Build a Dashboard with Plotly Dash

Launch Success Counts For All Sites

- Launch site “KSC LC39A” has the highest success rate
- Launch site “CCAFS SLC-40” has the lowest success rate, more than 3 times less than “KSC LC39A”

GITHUB

Launch Site with Highest Launch Success Ratio



To be precise, the KSC launch site has :

- 76.9% of success rate
- 23.1% of failure rate

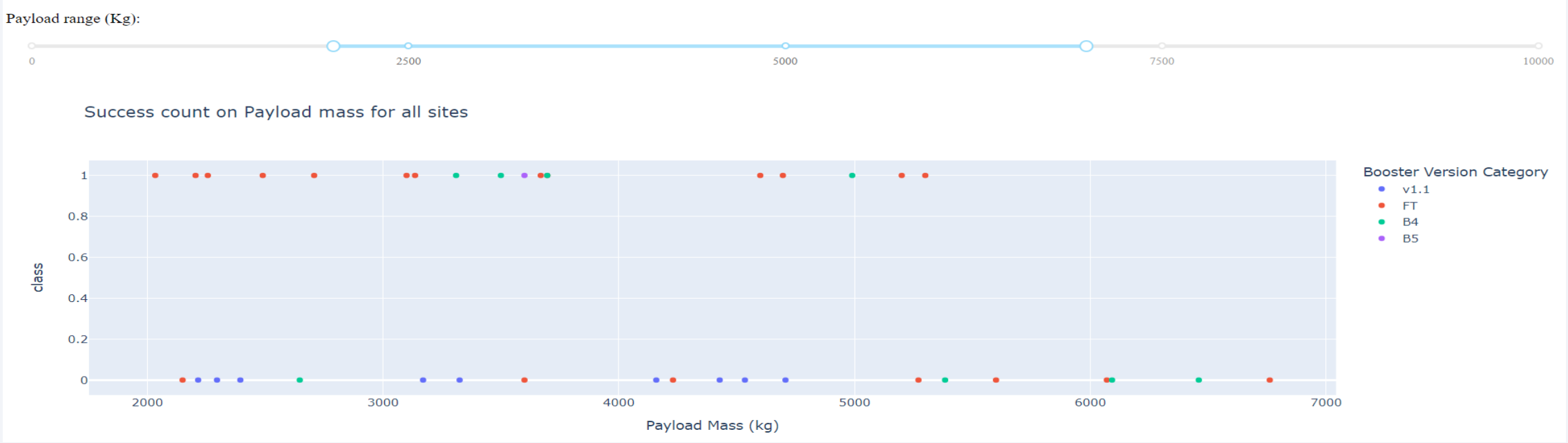
GITHUB

Payload vs. Launch Outcome Scatter Plot for All Sites

- Most falcon 9s have a payload mass between 2000 Kg and 6000 Kg
- Booster version category ‘v1.0 and v1.1’ has respectively “no success launches and lowest success rate”
- Booster version category ‘FT’ has the highest success launches rate,

GITHUB

Payload vs. Launch Outcome Scatter Plot for All Sites



- No falcon 9 from the booster version category “v1.0” has a payload mass between 2000 and 6000
- The falcon 9 from the booster version category “v1.1” with a payload mass between 2000 and 6000 have all failed.

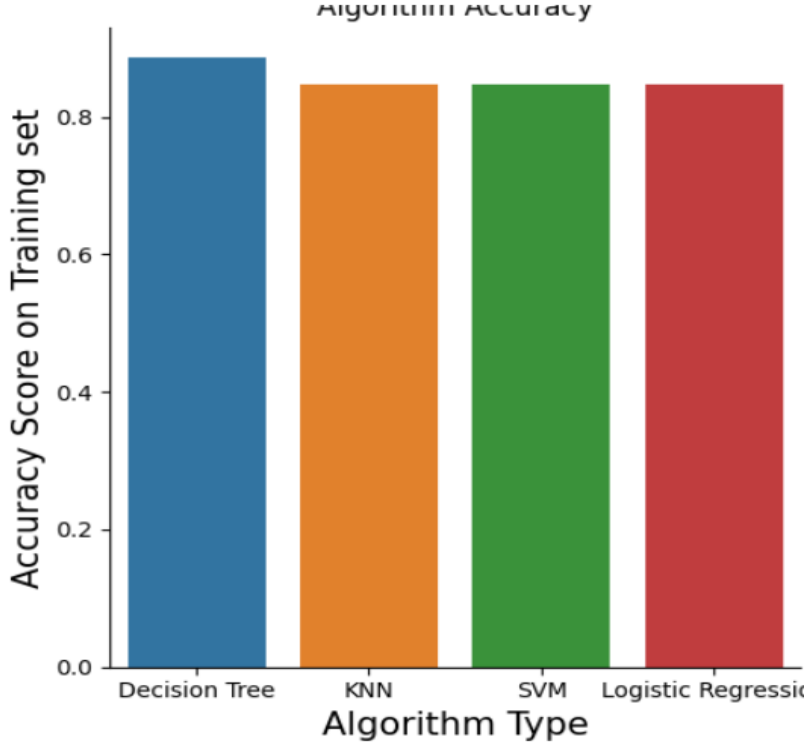
GITHUB

Section 5

Predictive Analysis (Classification)

Classification Accuracy

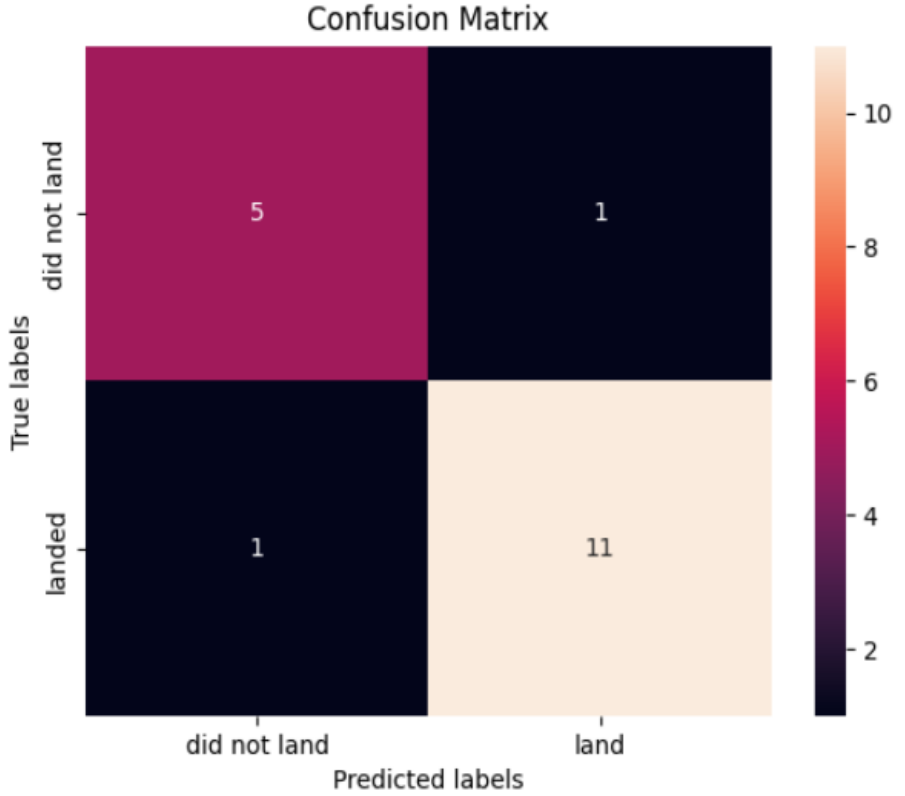
- Based on the Accuracy scores and as also evident from the bar chart, Decision Tree algorithm has the highest classification score with a value of .8750



GITHUB

Confusion Matrix

- The total samples in the positive class are 12. Similarly, the number of samples in the negative class is 6. The total number of samples evaluated is 18.
- The correct classifications are 11 for the positive class and 5 for the negative class.
- Now, 1 samples (top-right box) that was expected to be of the positive class were classified as the negative class by the model. So it is called “False Negatives”,
- Similarly, 1 samples (bottom-left box) were expected to be of negative class but were classified as “positive” by the model. They are thus called “False Positives.”



GITHUB

Conclusions

- Since 2013, the Falcon 9 launches success rate has considerably, reaching over 80% in 2019.
- First successful Ground landing was registered on 01/08/2018
- Success rate of falcon 9 launches increase as the number of flight increase.
- Launch site “KSC LC39A” has the highest success rate
- From launch site “VAFB SLC4E”, there are no rockets launched for payload mass greater than 10000Kg
- The best performing Machine Learning Classfication Model is the Decision Tree with an accuracy of about 87.5%. When the models were scored on the test data,theaccuracy score was around 83% for all models.

Appendix

- Haversine's Formula used to calculate distances on Folium Maps:

```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
```

GITHUB

Thank you!