IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Harshal Naik

June 22, 2025

Outline

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

EXECUTIVE SUMMARY

- Summary of methodologies
  1. Data Collection API
  2. Data Collection with Web Scraping
  3. Data Wrangling
  4. EDA with SQL
  5. EDA with Visualization
  6. Interactive Visual Analytics with Folium
  7. Machine Learning Prediction

- Summary of all results
  1. Exploratory Data Analysis result
  2. Interactive analytics in screenshots
  3. Predictive Analytics result

Introduction

Project background and context

- We will predict if the Falcon 9 first stage will land successfully. 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. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- Problems you want to find answers
  1. Which factors influence the success of the first stage landing?
  2. What is the rate of successful landings?

Section 1

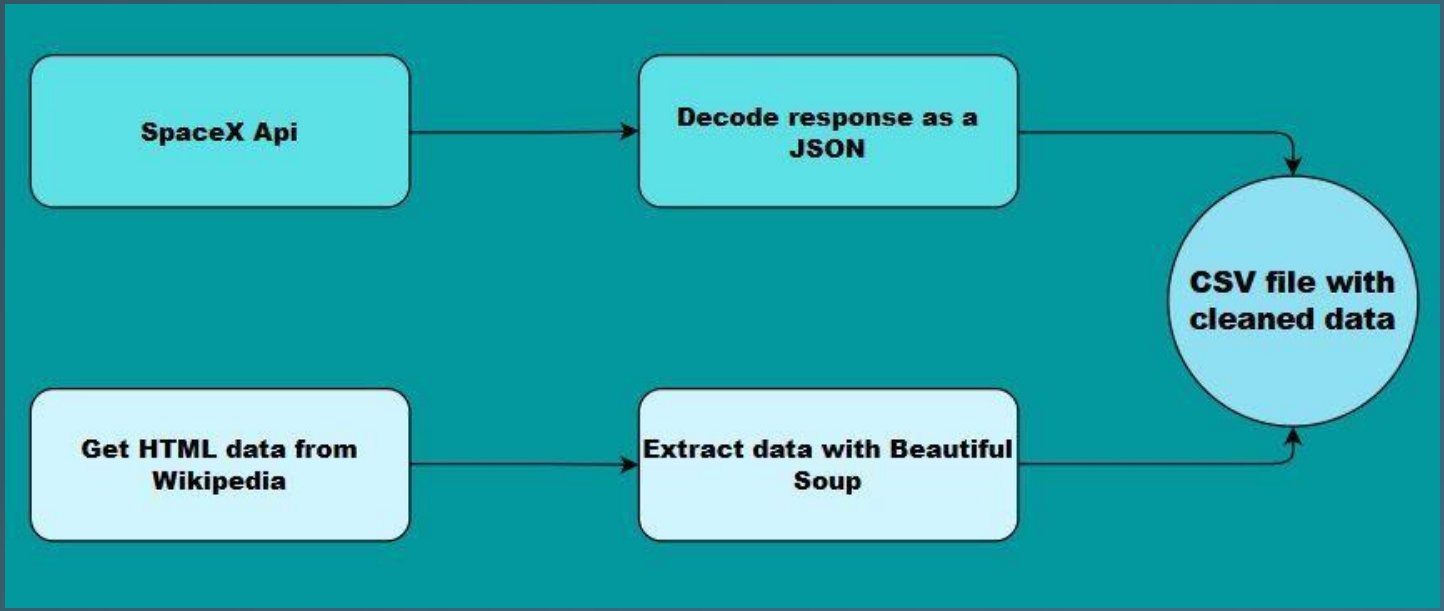
Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Collecting the data through the Space X API and web scraping from Wikipedia.
- Perform data wrangling
  - Application of One Hot encoding and data cleansing for use in ML models.
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Finding patterns and insights through data visualization techniques.
- Perform interactive visual analytics using Folium and Plotly Dash
  - Using Dash and Folium to perform interactive visualizations.
- Perform predictive analysis using classification models
  - Build and evaluate classification models.

Data Collection



- Data was gathered through the Space X API and Wikipedia web scraping, then extracted, decoded and cleaned and finally exported to a CSV file.

Data Collection - SpaceX API

1. First we request and parse the SpaceX launch data using the GET request which is then converted into a Pandas dataframe.
2. We filter the dataframe to only include Falcon 9 launches.
3. Then we perform some data wrangling to clean the data.

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

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

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

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

```
# Calculate the mean value of PayloadMass column
# Replace the np.nan values with its mean value
PayloadMass = pd.DataFrame(data_falcon9['PayloadMass'].values.tolist()).mean(1)
PayloadMass
```

GitHub URL of the completed SpaceX API calls notebook:
(It's a hyperlink, just click on it)

Data Collection - Scraping

1. First we request the Falcon9 Launch Wiki page from its URL.
2. We extract all column/variable names from the HTML table header.
3. Then we create a data frame by parsing the launch HTML tables.

GitHub URL of the completed Webscraping notebook:
(It's a hyperlink, just click on it)

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

```
200
```

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

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

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

```
# Use soup.title attribute
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

```
# 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')
```

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
launch_dict= dict.fromkeys(column_names)

# 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']=[]
```

Data Wrangling

1. We calculate the number of launches on each site.
2. We calculate the number and occurrence of each orbit.
3. We calculate the number and occurence of mission outcome of the orbits.
4. Finally we create a landing outcome label from Outcome column.

- GitHub URL of the completed Data Wrangling notebook:
  (It's a hyperlink, just click on it)

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

```
Launchsite
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: count, dtype: int64
```

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

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

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes`.Then assign it to a variable landing_outcomes.

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

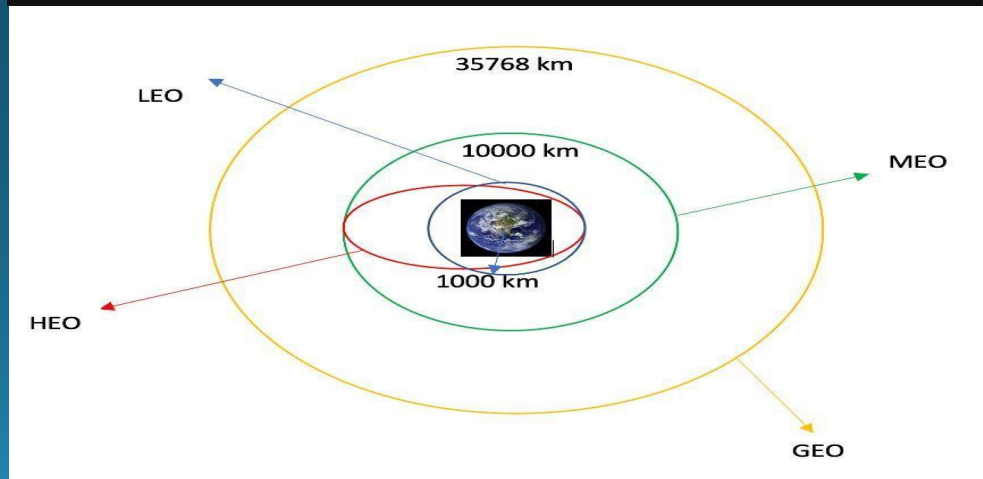
```
Outcome
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: count, dtype: int64
```

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []

for i, j in df["Outcome"].items():
    if j in bad_outcomes:
        outcome = 0
        landing_class.append(outcome)
    else:
        outcome = 1
        landing_class.append(outcome)
```

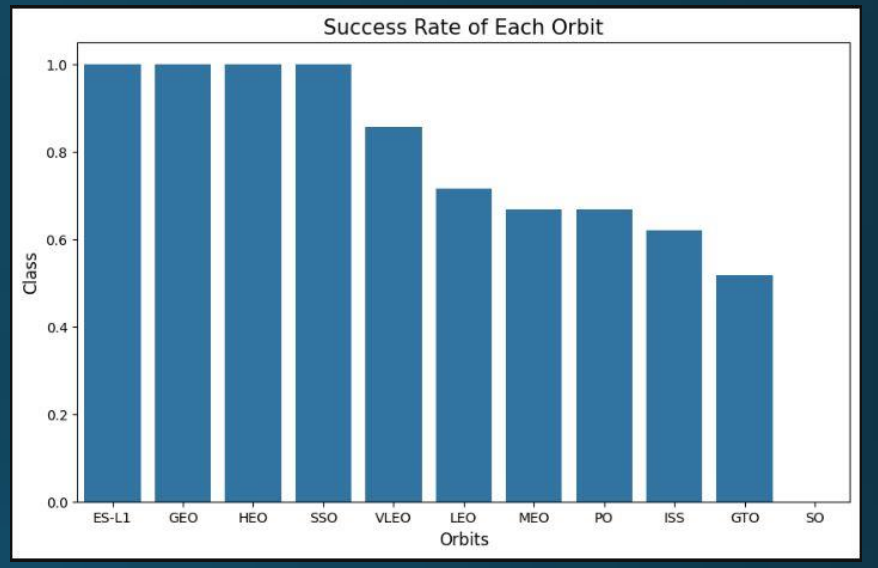
This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

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

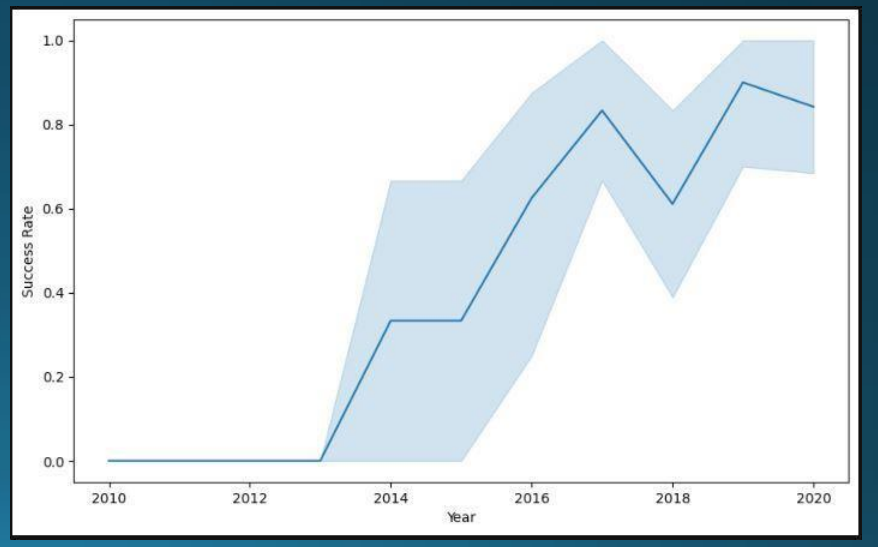


EDA with Data Visualization

- In the first image, we display the relationship between success rate of each orbit type using a bar chart.



- In the second image, we visualize the launch success yearly trend using a line chart.



- GitHub URL of the completed EDA with Data Visualization notebook: (It's a hyperlink, just click on it)

EDA with SQL

We applied EDA with SQL queries to obtain meaningful information on the data. Here are some examples:

- We display the names of the unique launch sites in the space mission:

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTBL;
```

- We display 5 records where launch sites begin with the string 'CCA':

```
%sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

- We display average payload mass carried by booster version F9 v1.1:

```
%sql SELECT AVG(payload_mass__kg_) AS Average_Payload_Mass_KG FROM SPACEXTBL WHERE booster_version = 'F9 v1.1';
```

- We list the total number of successful and failure mission outcomes:

```
%sql SELECT mission_outcome, COUNT(mission_outcome) AS Count \
     FROM SPACEXTBL GROUP BY mission_outcome;
```

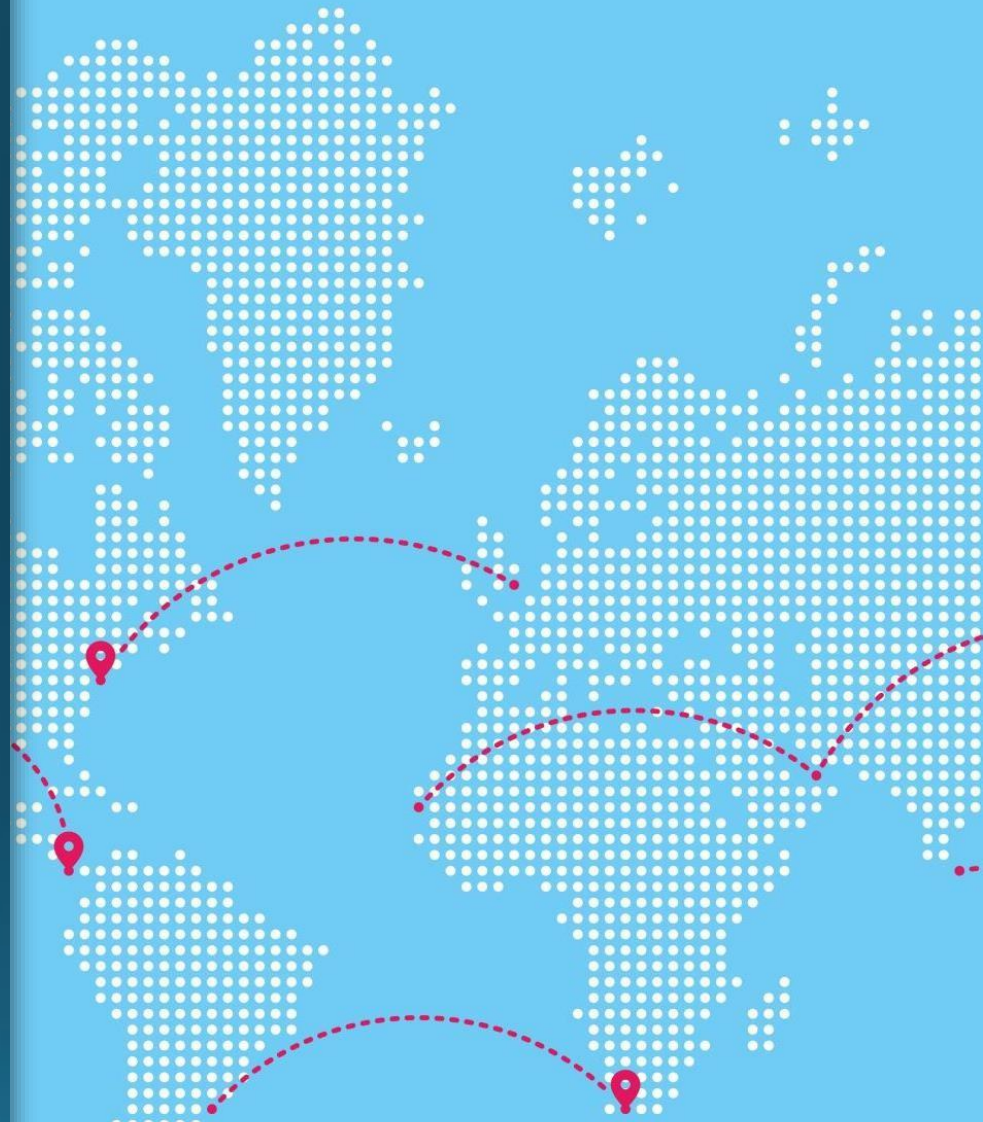
- GitHub URL of the completed EDA with SQL notebook:
  (It's a hyperlink, just click on it)

Build an Interactive Map with Folium

- We marked all launch sites on a map using site's latitude and longitude coordinates.
- We marked the success/failed launches for each site on the map.
- We calculated the distances between a launch site to its proximities.

Thanks to the insights we have gained about launch sites, we can conclude that:

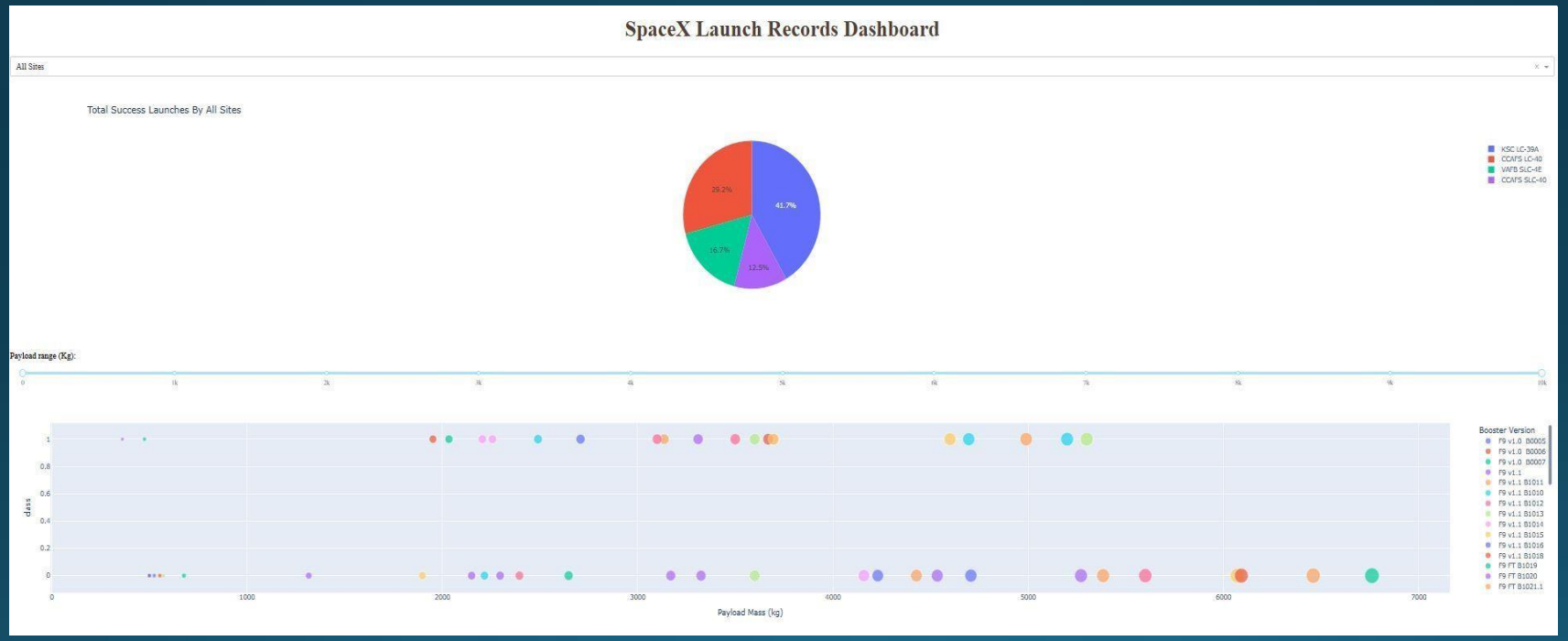
1. They are in close proximity to railways, which is good because it allows transporting the heavy components needed to assemble the rockets.
2. They are in close proximity to highways, which allows proper communication so that personnel can get there easily.
3. They are in close proximity to coastline to ensure that no debris falls on populated areas in case of failures.
4. They keep certain distance away from cities in order to mitigate risks in the event of a serious rocket failure.



- GitHub URL of the completed Launch Sites Location Analysis notebook:
(It's a hyperlink, just click on it)

Build a Dashboard with Plotly Dash

- We created a dashboard with Plotly Dash, pie charts showing the total launches for a given site and a scatter plot showing the relationship with the outcome and the payload mass for the different versions of the booster.



- GitHub URL of the completed SpaceX Dash App: (It's a hyperlink, just click on it)

Predictive Analysis (Classification)

- We used numpy and pandas, transformed the data and then split it for training and testing.

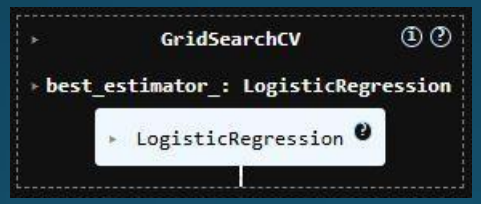
```
Y = data['Class'].to_numpy()
Y
```

```
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
print (f"Train set: {X_train.shape}  {Y_train.shape}")
print (f"Test set: {X_test.shape}  {Y_test.shape}")
```

```
Train set: (72, 83)  (72,)
Test set: (18, 83)  (18,)
```

- We have created different machine learning models using GridSearchCV.



- Finally we found the best performing method.

```
Best model is DecisionTree with a score of 0.8714285714285713
Best params is: {'criterion': 'gini', 'max_depth': 8, 'max_features': None, 'min_samples_leaf': 4, 'min_samples_split': 10, 'splitter': 'best'}
```

- GitHub URL of the completed Machine Learning Prediction notebook:
  (It's a hyperlink, just click on it)

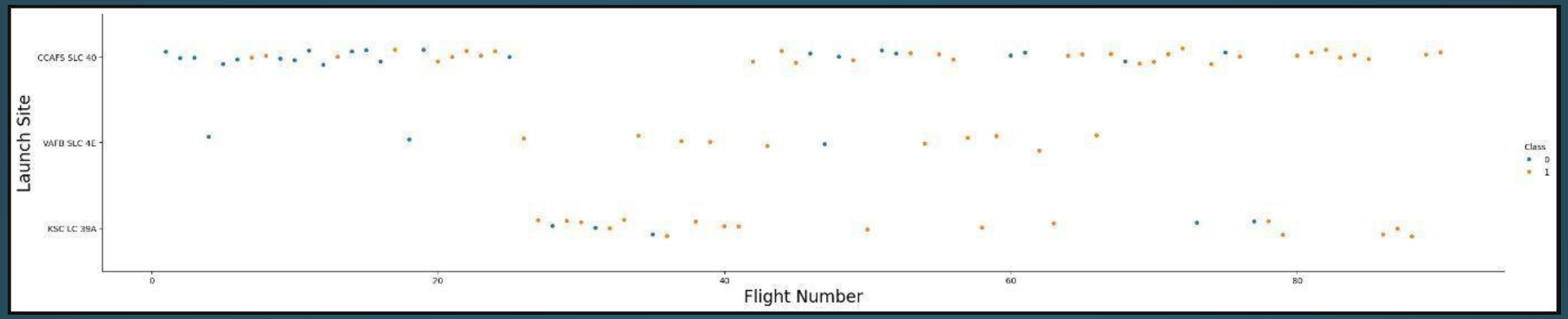
Results

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

Section 2

Insights drawn from EDA

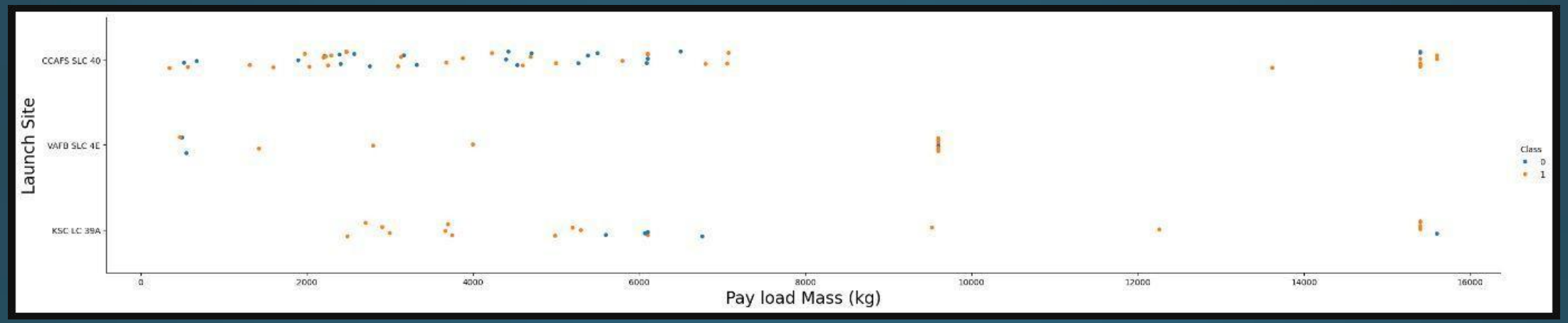
Flight Number vs. Launch Site



INSIGHTS:

1. CCAFS SLC 40 and KSC LC 39A have a higher number of flights compared to VAFB SLC 4E.
2. In general, the higher the number of flights, the higher the success rate.

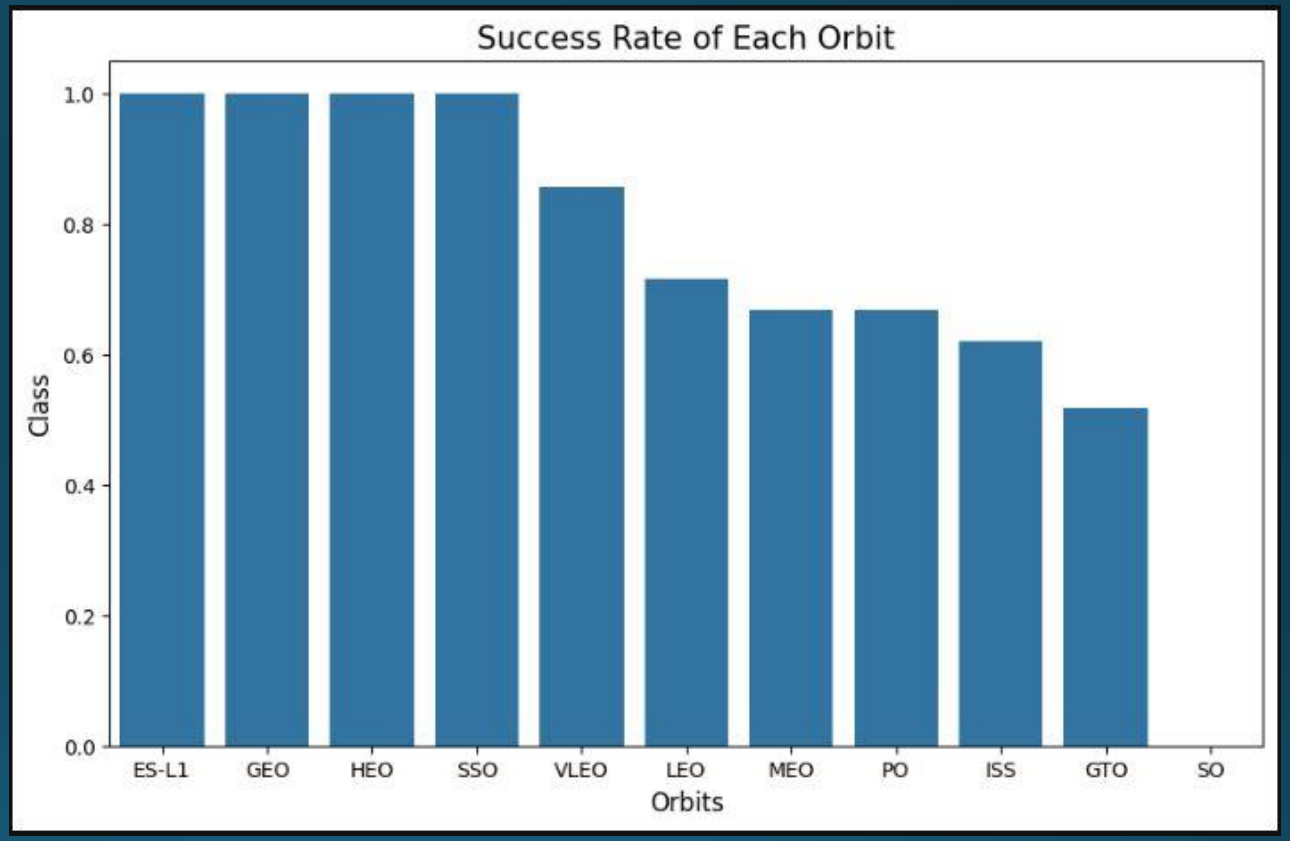
Payload vs. Launch Site



INSIGHTS:

1. The highest failure rate is recorded for payload masses between 4,000 kg and 6,000 kg.
2. Payload masses over 8,000 kg have a higher success rate.
3. In general, the higher the payload, the higher the success rate.

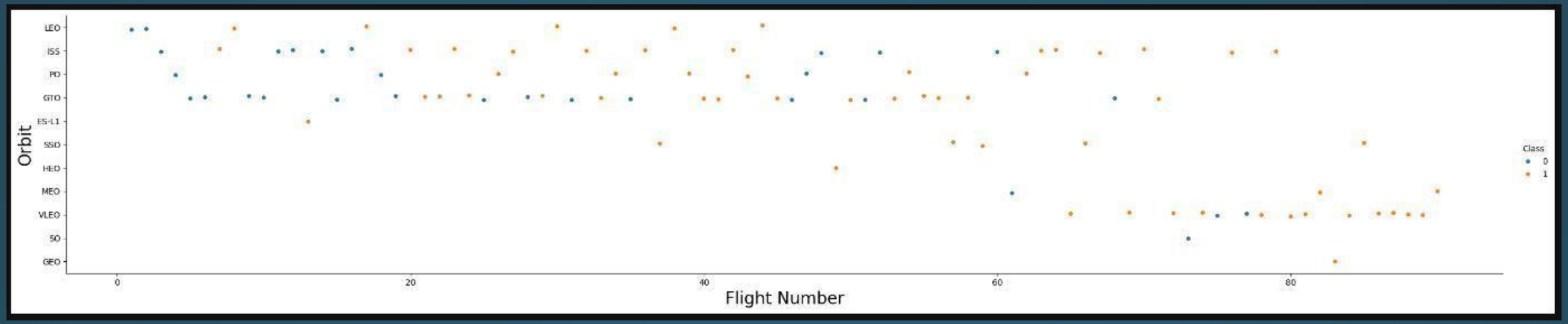
Success Rate vs. Orbit Type



INSIGHTS:

1. The top five orbits are ES-L1, GEO, HEO, SSO and VLEO.
2. SO orbit has no success rate.

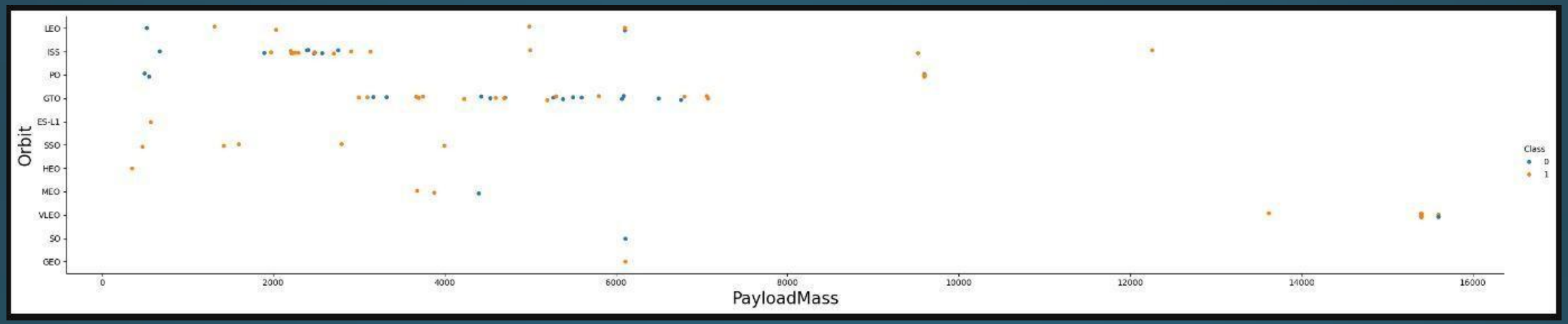
Flight Number vs. Orbit Type



INSIGHTS:

1. LEO orbit success rate is related to the number of flights, as the number of flights increases the success rate increases.
2. SO orbit has no success rate.
3. There is an increasing trend of launches from VLEO orbit in recent years.

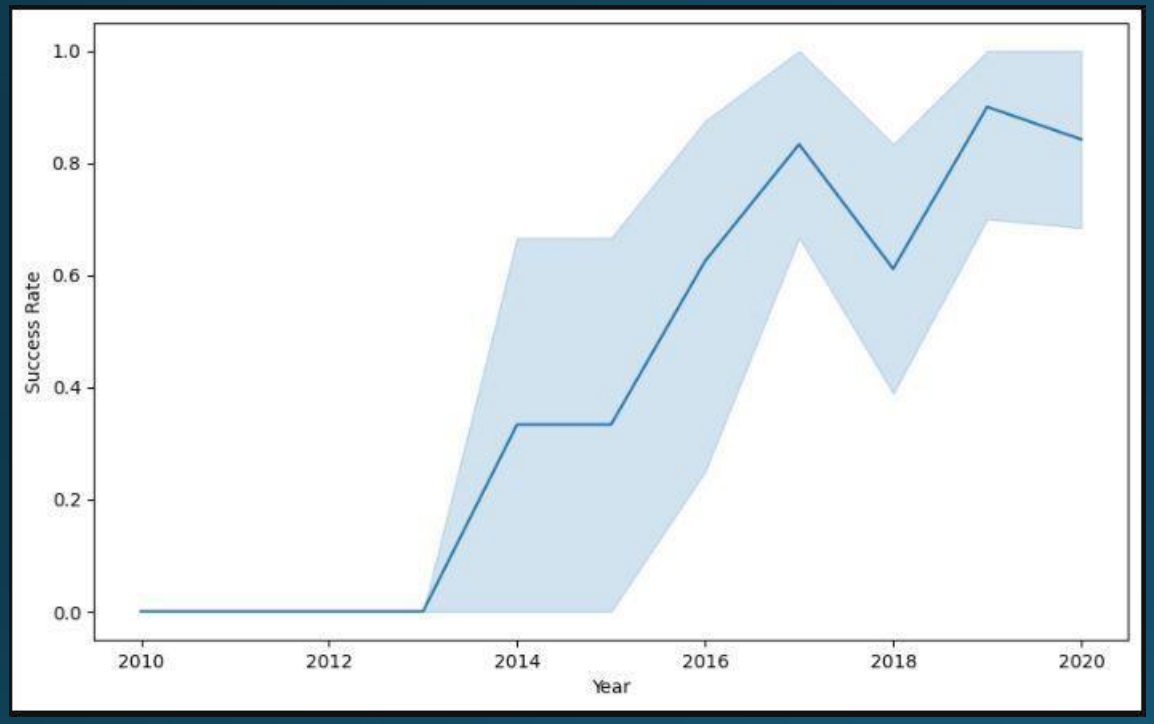
Payload vs. Orbit Type



INSIGHTS:

1. Payload mass between 3000 and 8000 is affecting GTO orbit.
2. With heavy payloads, the success rate is higher in ISS, LEO and PO orbits.

Launch Success Yearly Trend



- Since 2013 the success rate had a massive increase until 2018 which decreased a bit, but from 2019 onwards it continued to increase like never before.

All Launch Site Names

- We can get the unique launch site names by using the DISTINCT keyword.

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTBL;
```

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

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

Launch Site Names Begin with 'CCA'

- We can get 5 launch sites starting with the string 'CCA' by using the keyword "LIMIT".

```
%sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- We can get the total payload mass (45596 Kg) by using the "SUM" keyword.

```
%sql SELECT SUM(payload_mass__kg_) AS Total_Payload_Mass_KG \
     FROM SPACEXTBL WHERE customer = 'NASA (CRS)';
```

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

Total_Payload_Mass_KG
45596

Average Payload Mass by F9 v1.1

- We can get the average payload mass carried by booster version F9 v1.1 (2928.4 Kg) by using the "AVG" keyword.

```
%sql SELECT AVG(payload_mass__kg_) AS Average_Payload_Mass_KG FROM SPACEXTBL WHERE booster_version = 'F9 v1.1';
```

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

Average_Payload_Mass_KG
2928.4

First Successful Ground Landing Date

- We can get the first successful ground landing date (2015-12-22) by using the "MIN" keyword because the first date can be considered the minimum date.

```
%sql SELECT MIN(DATE) AS First_Successful_Landing \
     FROM SPACEXTBL WHERE landing_outcome = 'Success (ground pad)';
```

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

First_Successful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We can get the successful drone ship landing with payload between 4000 and 6000 (2015-12-22) by setting as the landing outcome "Success (drone ship)"

```
%sql SELECT PAYLOAD \
     FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' \
     AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

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

Payload
JCSAT-14
JCSAT-16
SES-10
SES-11 / EchoStar 105

Total Number of Successful and Failure Mission Outcomes

- We can get the total of successful and failure mission outcomes by using the "COUNT" keyword.

```
%sql SELECT mission_outcome, COUNT(mission_outcome) AS Count \
     FROM SPACEXTBL GROUP BY mission_outcome;
```

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

Mission_Outcome	Count
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- We can get the boosters carried maximum payload (15600 Kg) by using the "MAX" keyword.

```
%sql SELECT booster_version, payload_mass__kg_ \
     FROM SPACEXTBL \
     WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXTBL);
```

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

- We can get the 2015 launch records by using the "SUBSTR" keyword and in the "WHERE" function setting the year value to "2015".

```
%sql SELECT SUBSTR(Date,6,2) AS Month, DATE,BOOSTER_VERSION, LAUNCH_SITE, [Landing_Outcome] \
     FROM SPACEXTBL \
     WHERE [Landing_Outcome] = 'Failure (drone ship)' AND SUBSTR(Date,0,5) = '2015';
```

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

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

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

- We can order the value in descending order by using the "DESC" keyword and display its count by using the "COUNT" keyword.

```
%sql SELECT [Landing_Outcome], COUNT(*) AS Count \
    FROM SPACEXTBL \
    WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY [Landing_Outcome] \
    ORDER BY Count DESC;
```

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

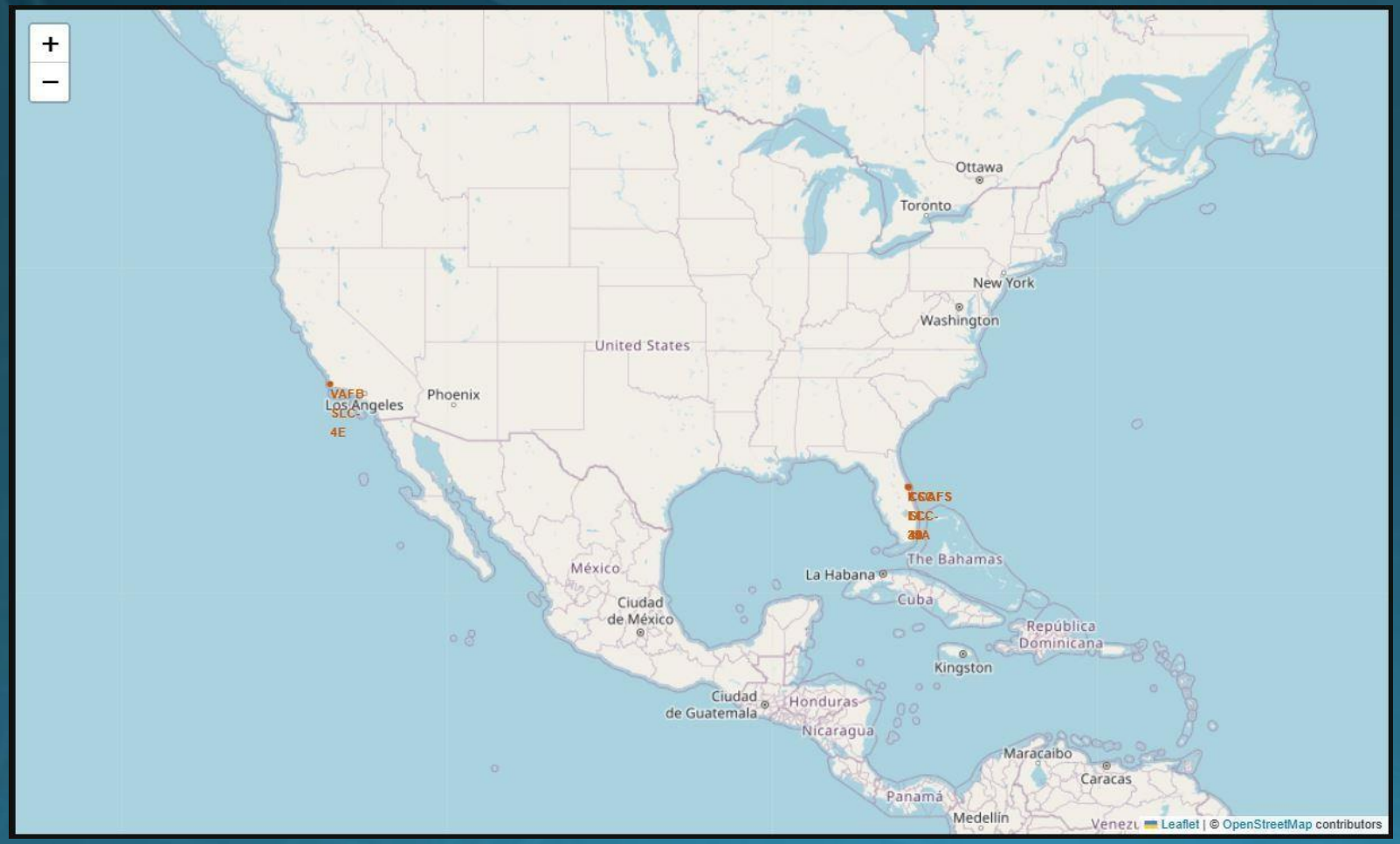
Landing_Outcome	Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

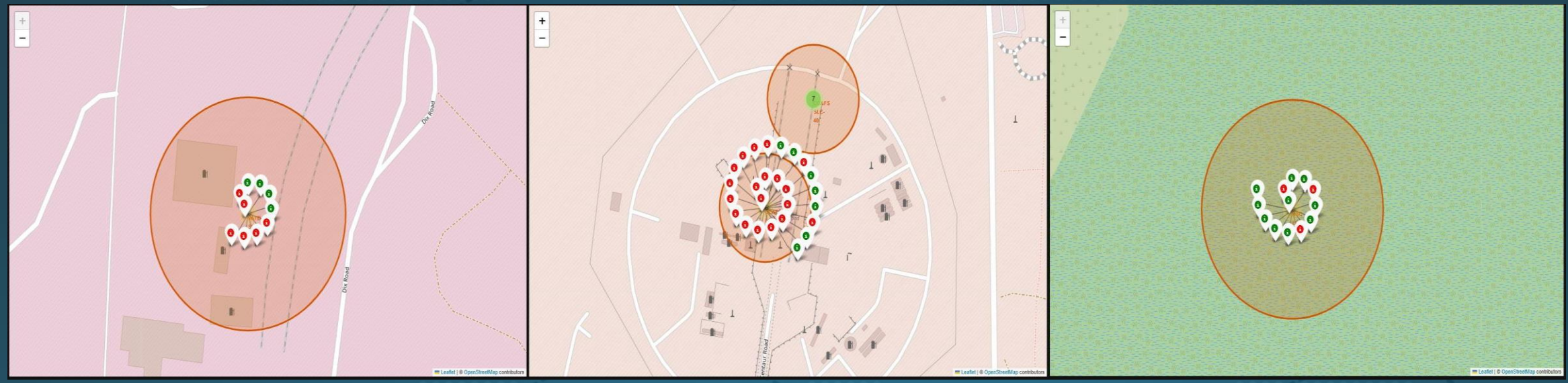
SpaceX Worldwide Launch Sites

- SpaceX launch sites are located on the U.S. coasts of California and Florida.



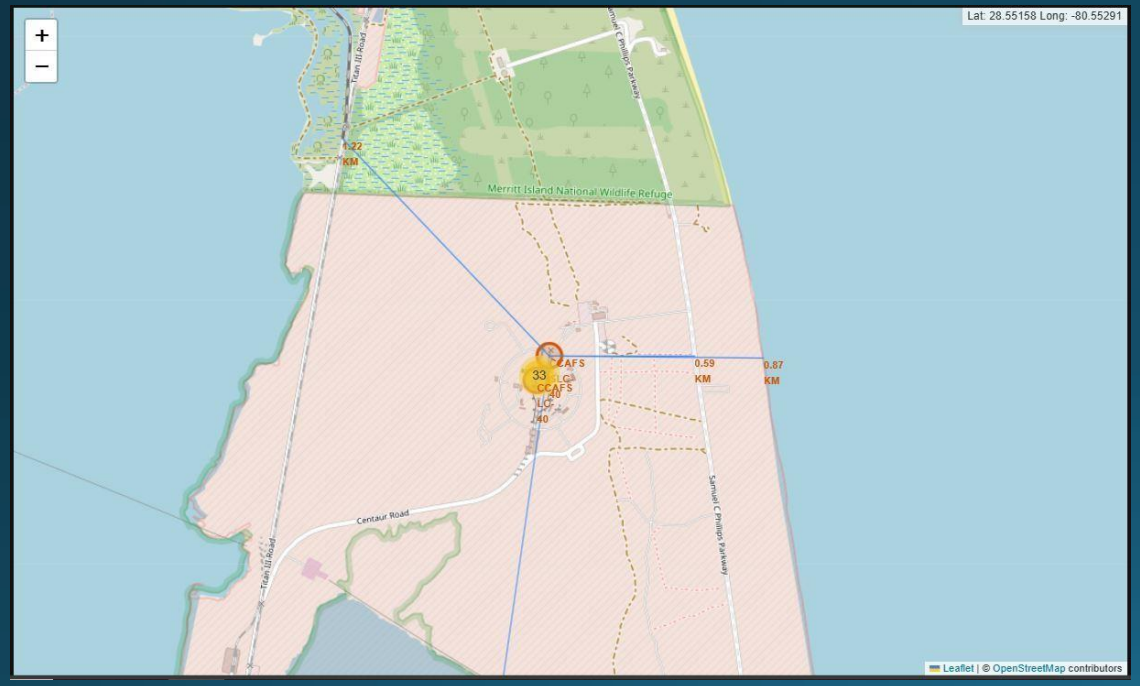
Launch Sites Map Markers

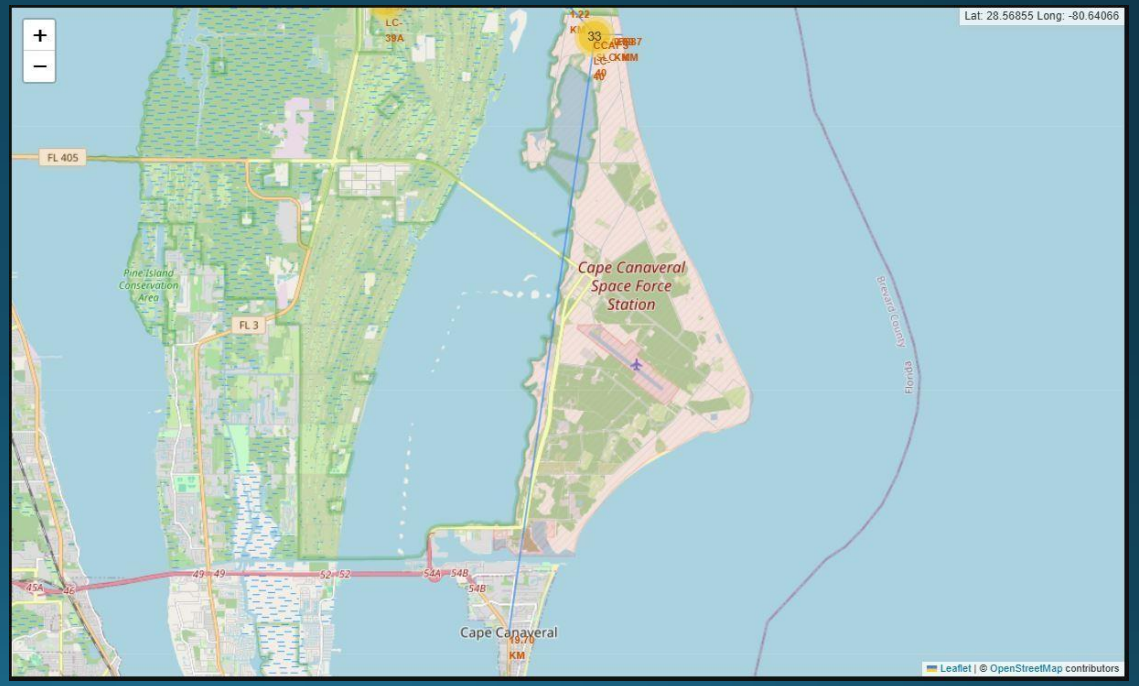
- Map markers for the California and Florida launch sites are shown below.

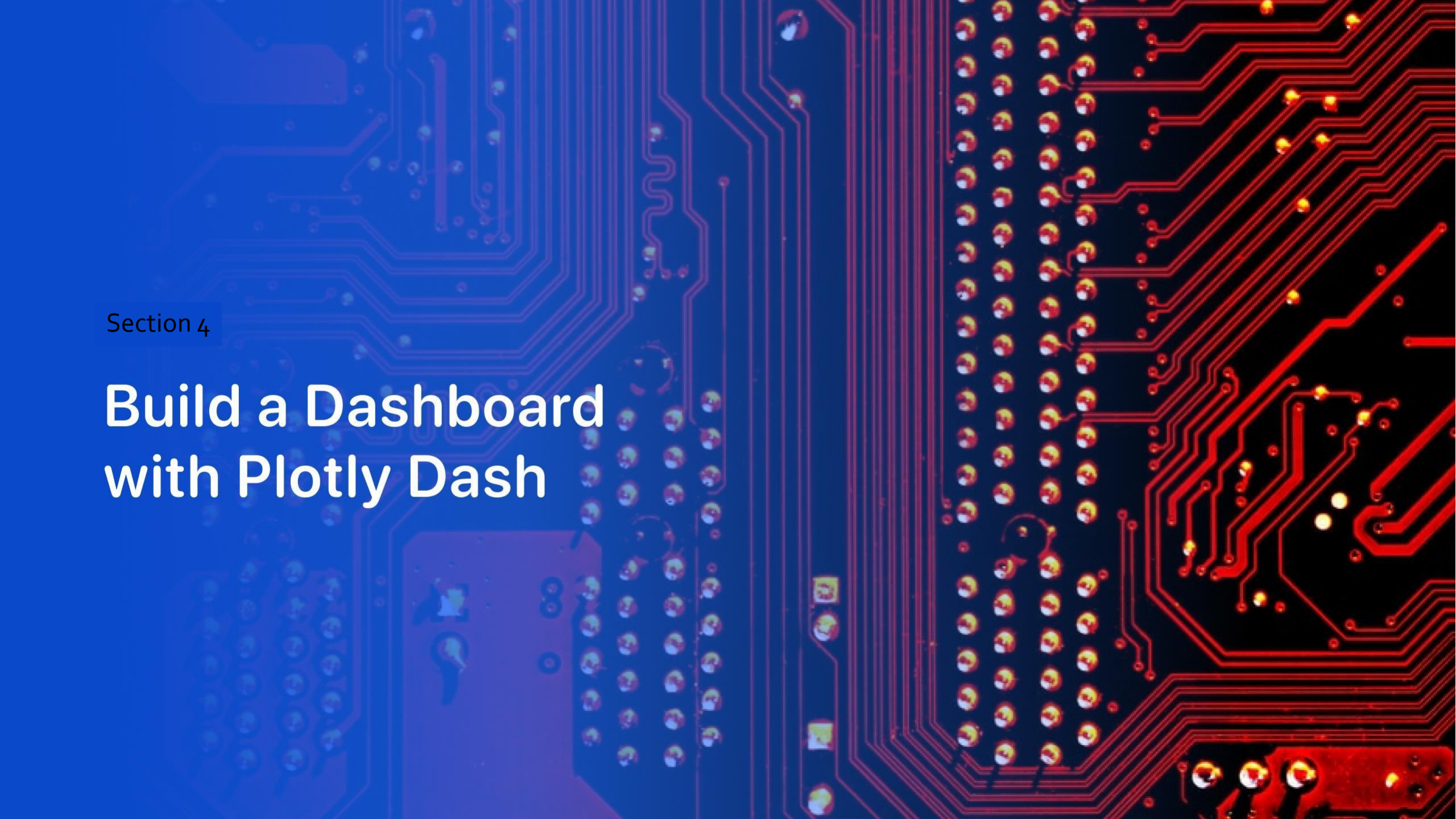


Distance between Launch Sites and Landmarks

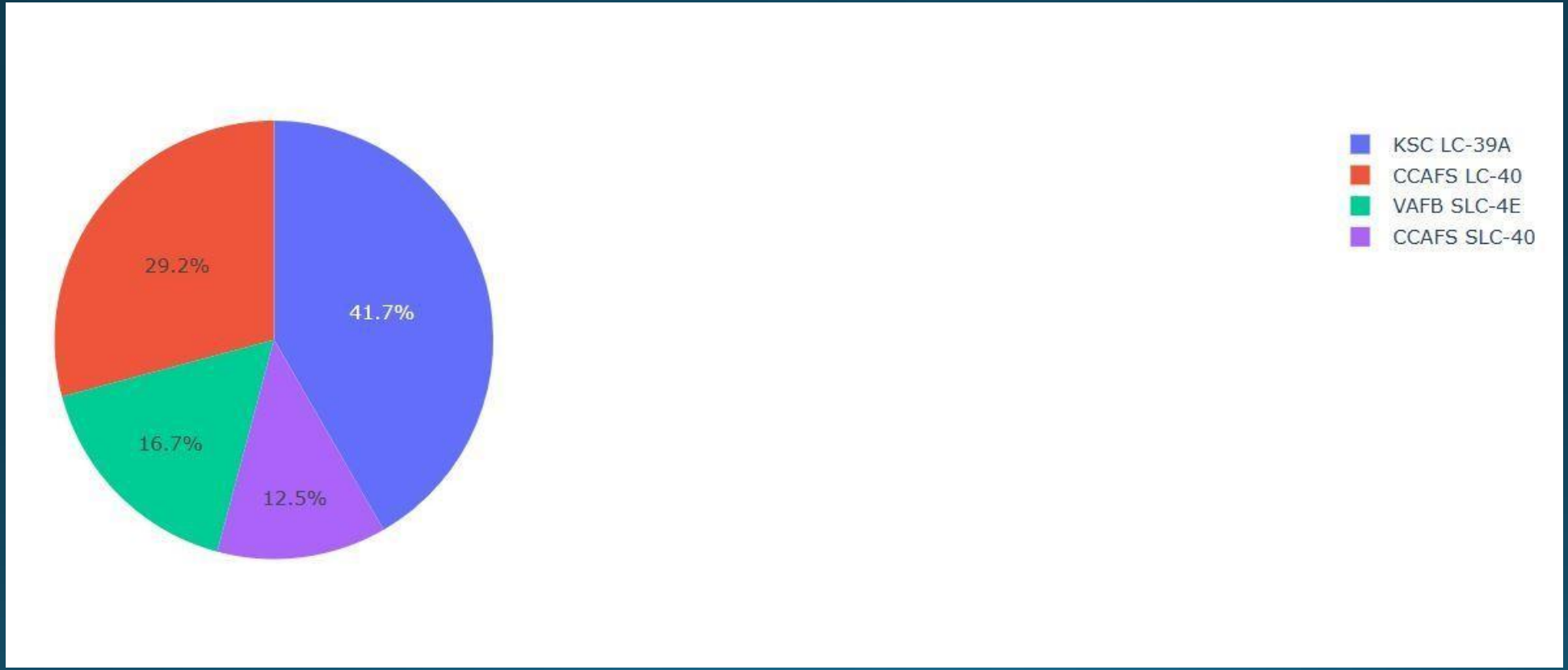
- Distance to closest coastline, highway, railway and city are shown below:





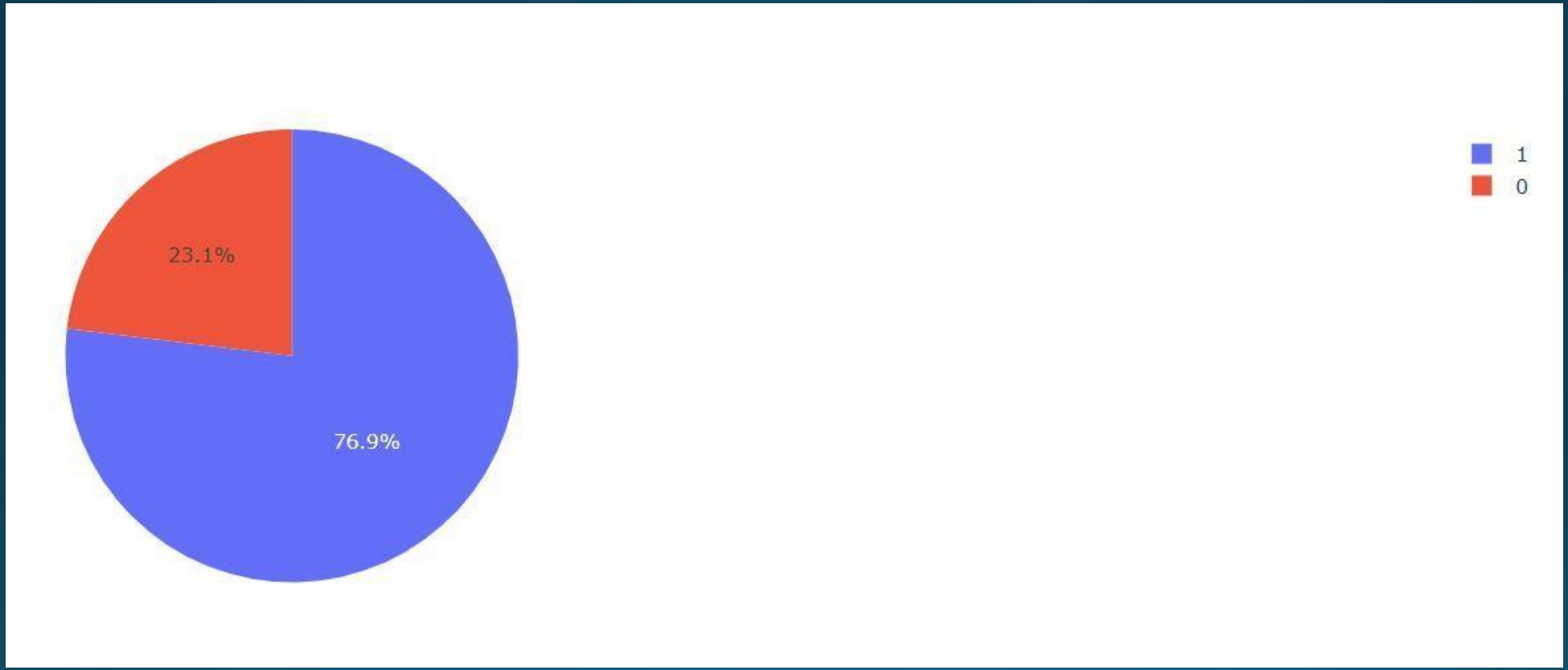

Section 4
Build a Dashboard with Plotly Dash

Total number of successful launches from each launch site.



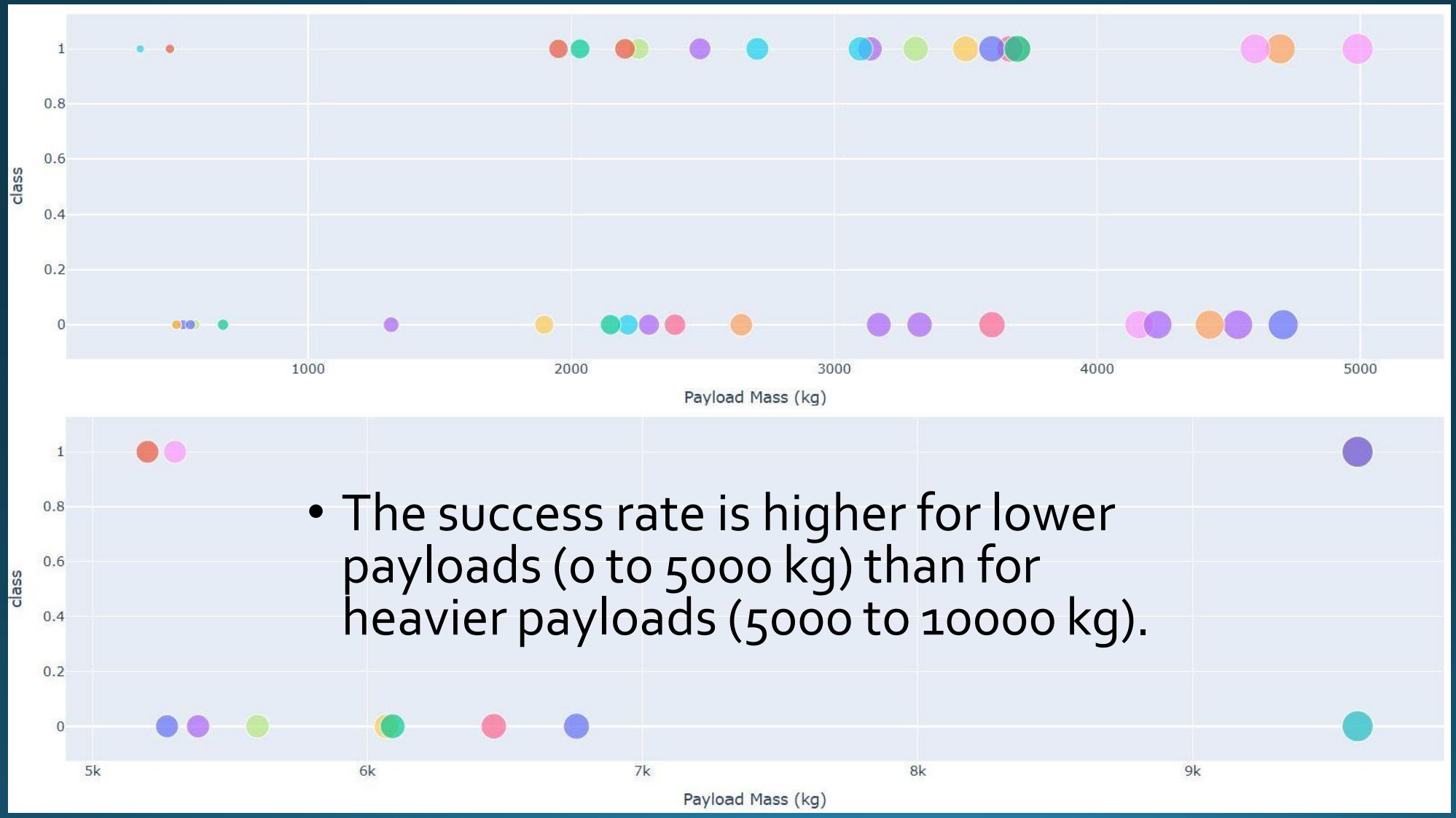
- KSC LC-39A has the highest success rate.

Launch site with the highest success rate



- KSC LC-39A has a 76.9% success rate and only a 23.1% failure rate.

Scatter plot of Payload vs. Outcome for all sites, with different payloads.

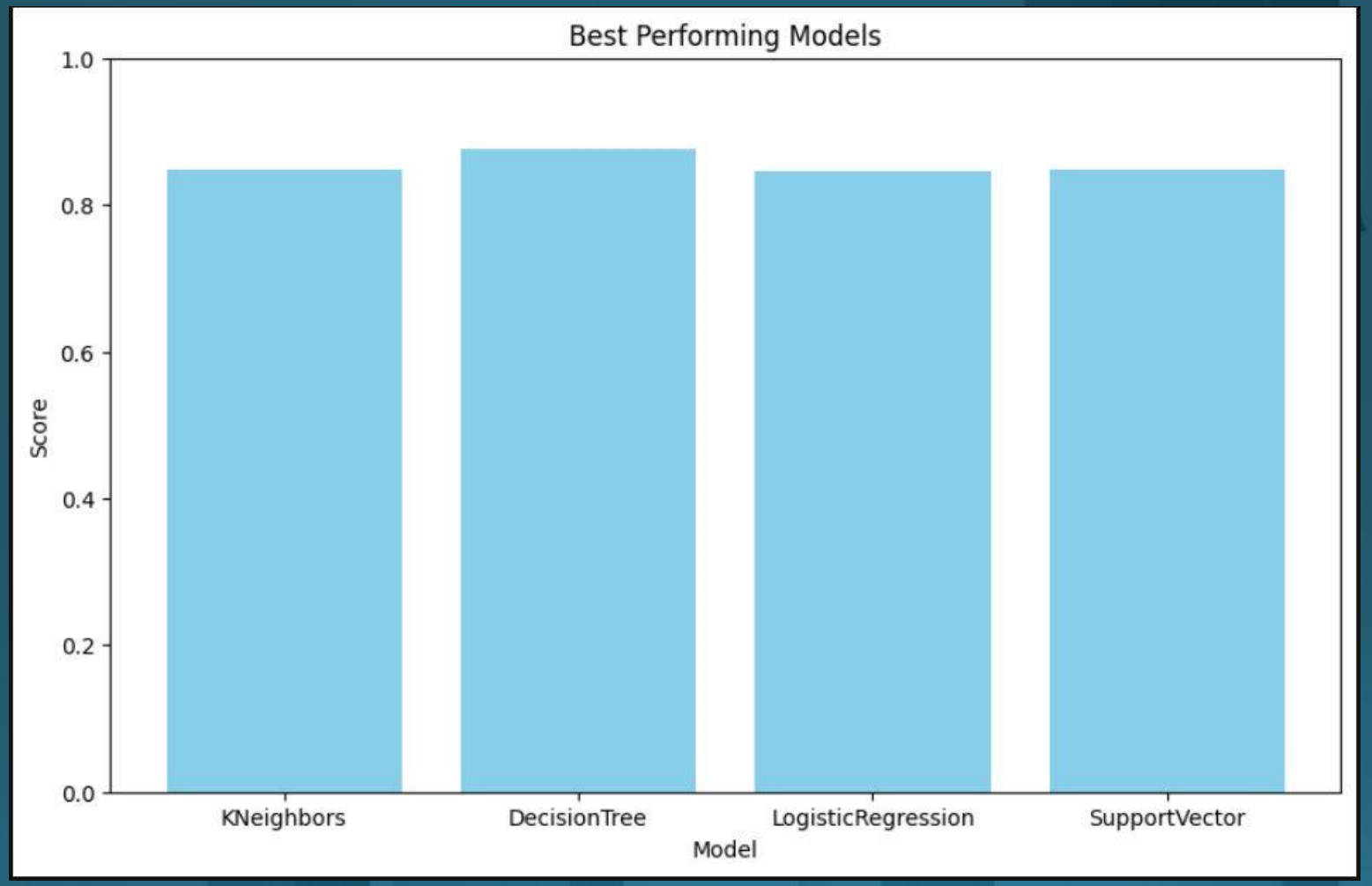


- The success rate is higher for lower payloads (0 to 5000 kg) than for heavier payloads (5000 to 10000 kg).

Section 5

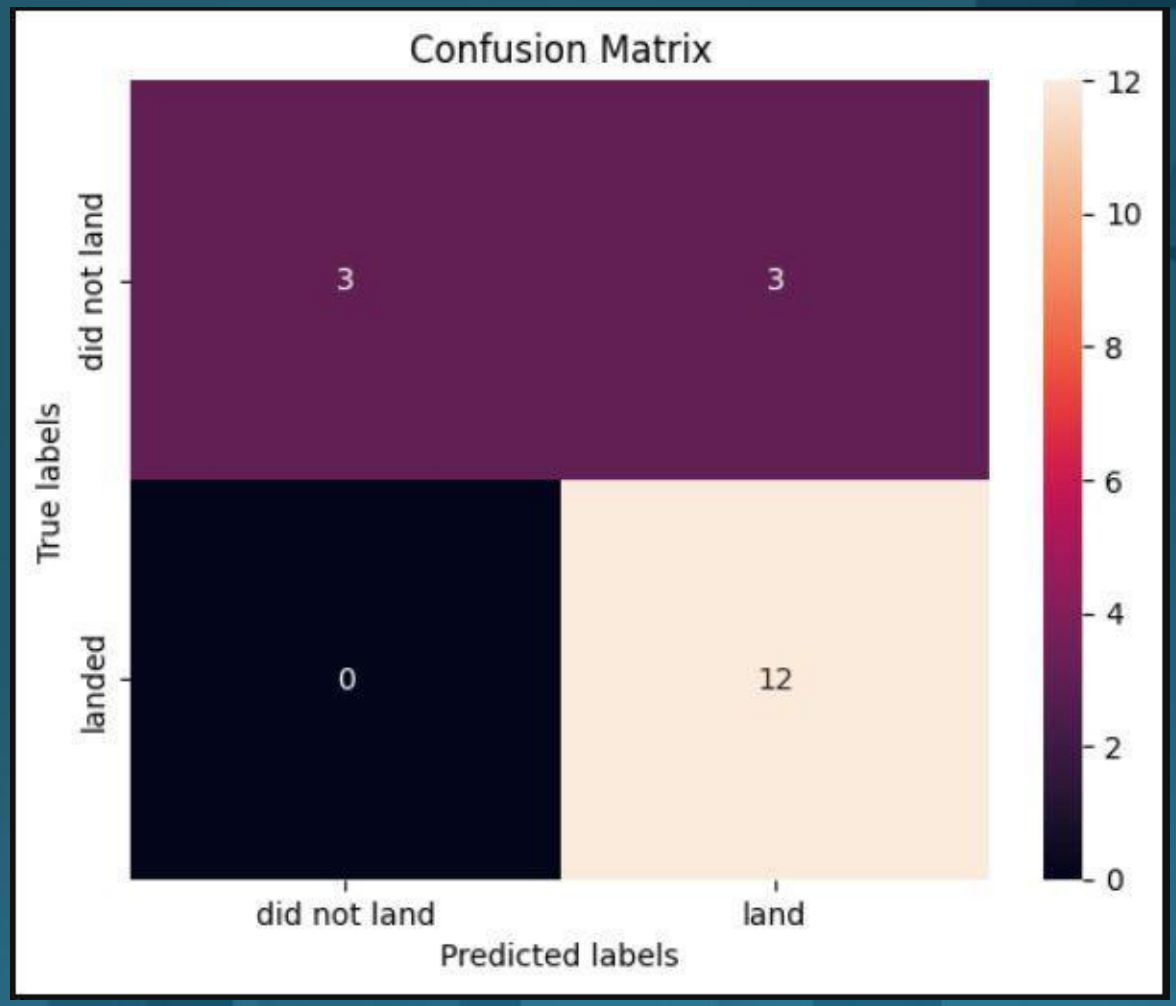
Predictive Analysis (Classification)

Classification Accuracy



- Best model is DecisionTree with a score of 0.87

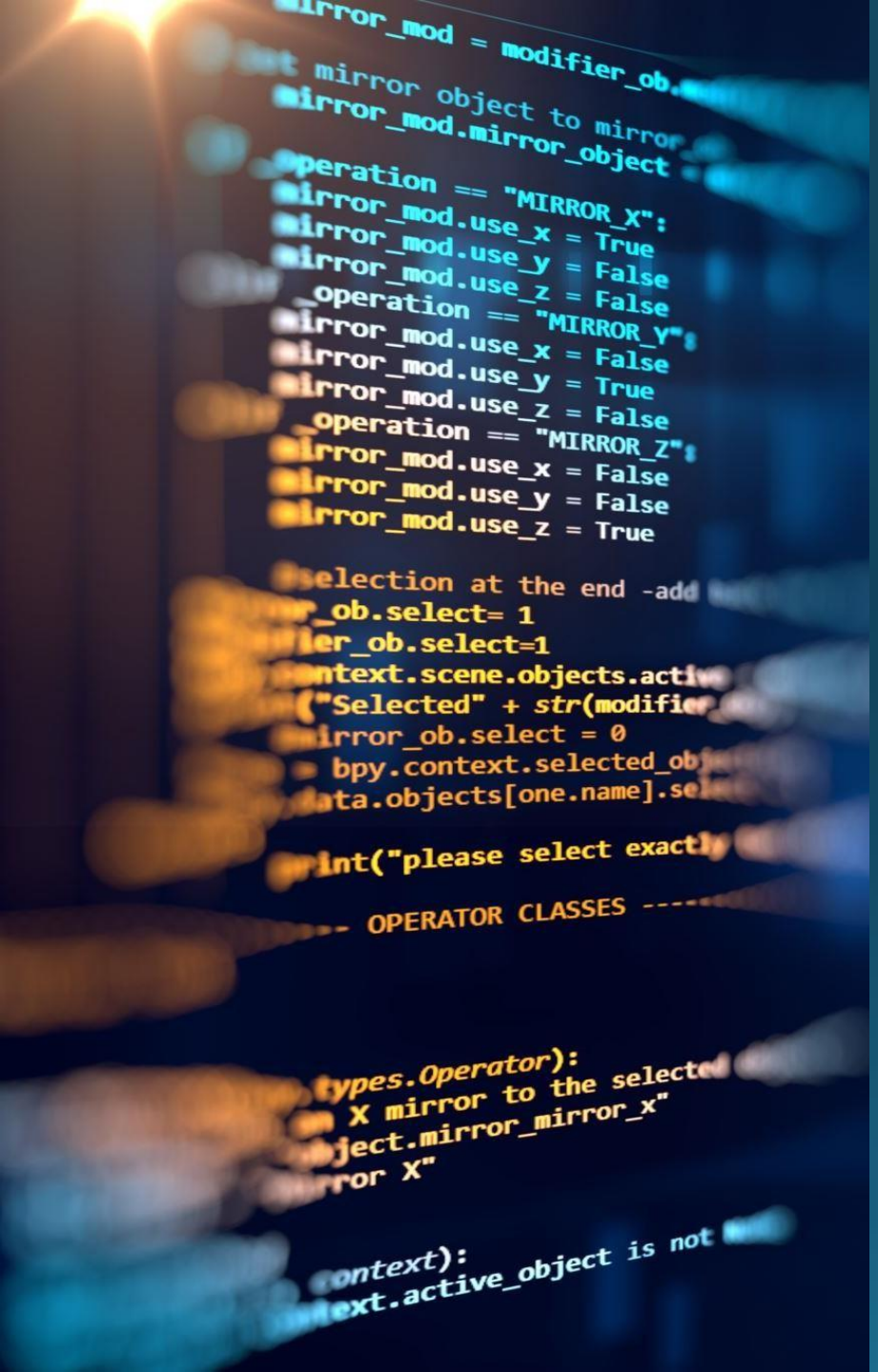
Confusion Matrix



- The confusion matrix of the decision tree classifier clearly shows that it can differentiate between different classes, but it also shows an issue with false positive values.

Conclusions

- Decision Tree is the best performing model.
- The launch site with the highest success rate is KSC LC-39A
- The launch success rate started to increase in 2013, reaching the highest rate in 2019.
- The payload range from 0 to 5000 kg has a higher success rate than heavier payloads.
- We have explained why it's better for launch sites to be near coastlines, highways, railroads and not located near populated areas.



Appendix

- My GitHub repository with all the notebooks and python code for this project can be found at the following link:
- harshal2424/IBM-applied-data-science-capstone: IBM Data Science Professional Certificate final project (github.com)

Thank you!