Winning Space Race with Data Science

Carlos Buitrago
24/08/23

Outline

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

Executive Summary

- **Summary of methodologies:**
  - Data collected through the SpaceX API and web scraping
  - Data cleaning and wrangling
  - Exploratory data analysis with SQL
  - Exploratory data analysis with data visualization
  - Visual analytics of launch sites locations with folium
  - First stage landing prediction with machine learning

- **Summary of all results:**
  - Exploratory data analysis visualizations
  - Visual analytics with folium
  - Machine learning prediction

Introduction

- **Project background and context:**

  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:**

  - Which variables are involved in the accomplishment of a successful landing?
  - Can we predict whether a landing will be successful based on previous data?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected via the SpaceX API and web scraping (Wikipedia)
- Perform data wrangling
  - The label used for training supervised models was determined from patterns in the data
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Classification models were built, tuned and evaluated using the scikit-learn library

Data Collection – SpaceX API

- The get request was used to collect data from the SpaceX API. Some basic data wrangling and formatting was done after.
- GitHub URL of the Notebook: https://github.com/Carlos-Buitrago/IBM-DS-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

To make the requested JSON results more consistent, we will use the following static response object for this project:

In [12]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork
static_response = requests.get(static_json_url)
```

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

In [25]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(static_response.json())
```

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

In [40]:
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']
```

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

In [49]:
```
# Calculate the mean value of PayloadMass column
payload_mean = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, payload_mean)

data_falcon9.isnull().sum()
```

Data Collection - Scraping

- Falcon 9 historical launch records were web scraped from Wikipedia using BeautifulSoup. The table was parsed and converted to a dataframe.
- GitHub URL of the notebook: https://github.com/Carlos-Buitrago/IBM-DS-Capstone/blob/main/jupyter-labs-data-collection-webscraping.ipynb

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

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

response = requests.get(static_url)
```

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text)
```

```
# 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)
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

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

Data Wrangling

- Some EDA was made to find some patterns in the data and determine what would be the label for training supervised models. The multiple possible outcomes were converted into training labels 1 and 0.
- GitHub URL of the notebook: https://github.com/Carlos-Buitrago/IBM-DS-Capstone/blob/main/labs-jupyter-spacex-data-wrangling.ipynb

In [9]:
```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

We create a set of outcomes where the second stage did not land successfully:

In [10]:
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

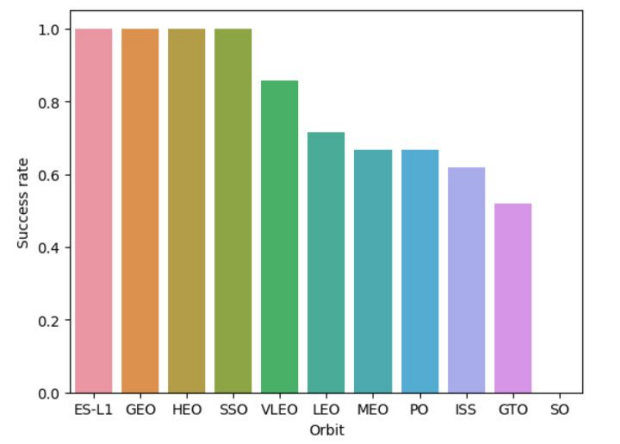
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []

for i,outcome in enumerate(df['Outcome']):
    if outcome in bad_outcomes:
        landing = 0
    else:
        landing = 1
    landing_class.append(landing)
```

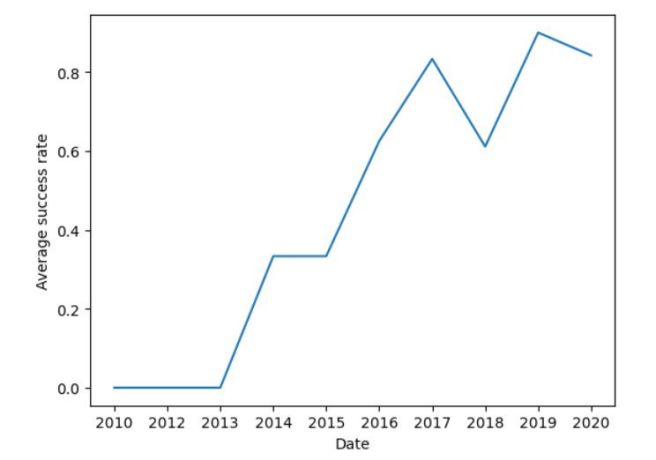
This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

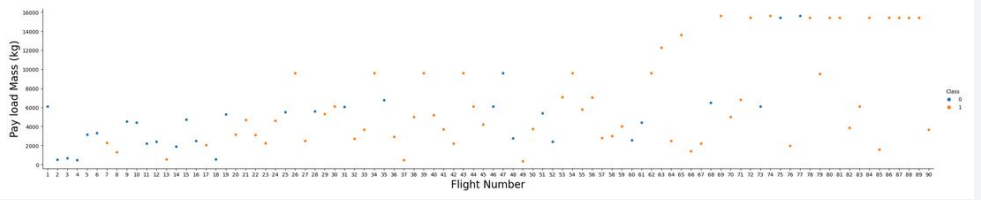
```
df['Class']=landing_class
df[['Class']].head(8)
```

EDA with Data Visualization

- The relationship between the multiple variables were plotted: flight number vs payload mass, flight number vs launch site, flight number and orbit, orbit and payload mass. The success rate of each orbit was analyzed and the yearly trend of the success rate was observed.







- GitHub URL of the notebook: https://github.com/Carlos-Buitrago/IBM-DS-Capstone/blob/main/jupyter-labs-eda-visualization.ipynb

EDA with SQL

- EDA was performed using SQL to:
  - **Display the names of the unique launch sites in the space mission**
  - **Display 5 records where launch sites begin with the string 'CCA'**
  - **Display the total payload mass carried by boosters launched by NASA (CRS)**
  - **Display average payload mass carried by booster version F9 v1.1**
  - **List the date when the first successful landing outcome in ground pad was 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.**
  - **List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.**
- GitHub URL of the notebook: https://github.com/Carlos-Buitrago/IBM-DS-Capstone/blob/main/jupyter-labs-eda-sql.ipynb

Build an Interactive Map with Folium

- All launch sites were marked on the map using circles to highlight them and markers to display their names.
- The success/failed launches for each site were marked using marker clusters, since many launch records had the same coordinates.
- The distances between a launch site and its proximities were calculated.
- GitHub URL of the notebook: https://github.com/Carlos-Buitrago/IBM-DS-Capstone/blob/main/jupyter-labs-visual-analytics-folium.ipynb

Build a Dashboard with Plotly Dash

- An interactive dashboard was built using plotly dash. A dropdown with the different launch sites and a slider to select the desired payload mass were added.

- A pie chart with the total success launches by site and a scatterplot with the correlation between payload and launch success for the different booster versions were displayed.

Predictive Analysis (Classification)

- The data was standardized and then split into training and test data.

```
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)
```

- For the different models (logistic regression, support vector machine, decision tree and k nearest neighbors) the best hyperparameters were found using GridSearchCV. After that they were trained using the training set and their scores were calculated in order to find the best performing model.

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}#
lr=LogisticRegression()
logreg_cv = GridSearchCV(estimator=lr, param_grid=parameters, cv=10)
logreg_cv.fit(X_train,Y_train)
```

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

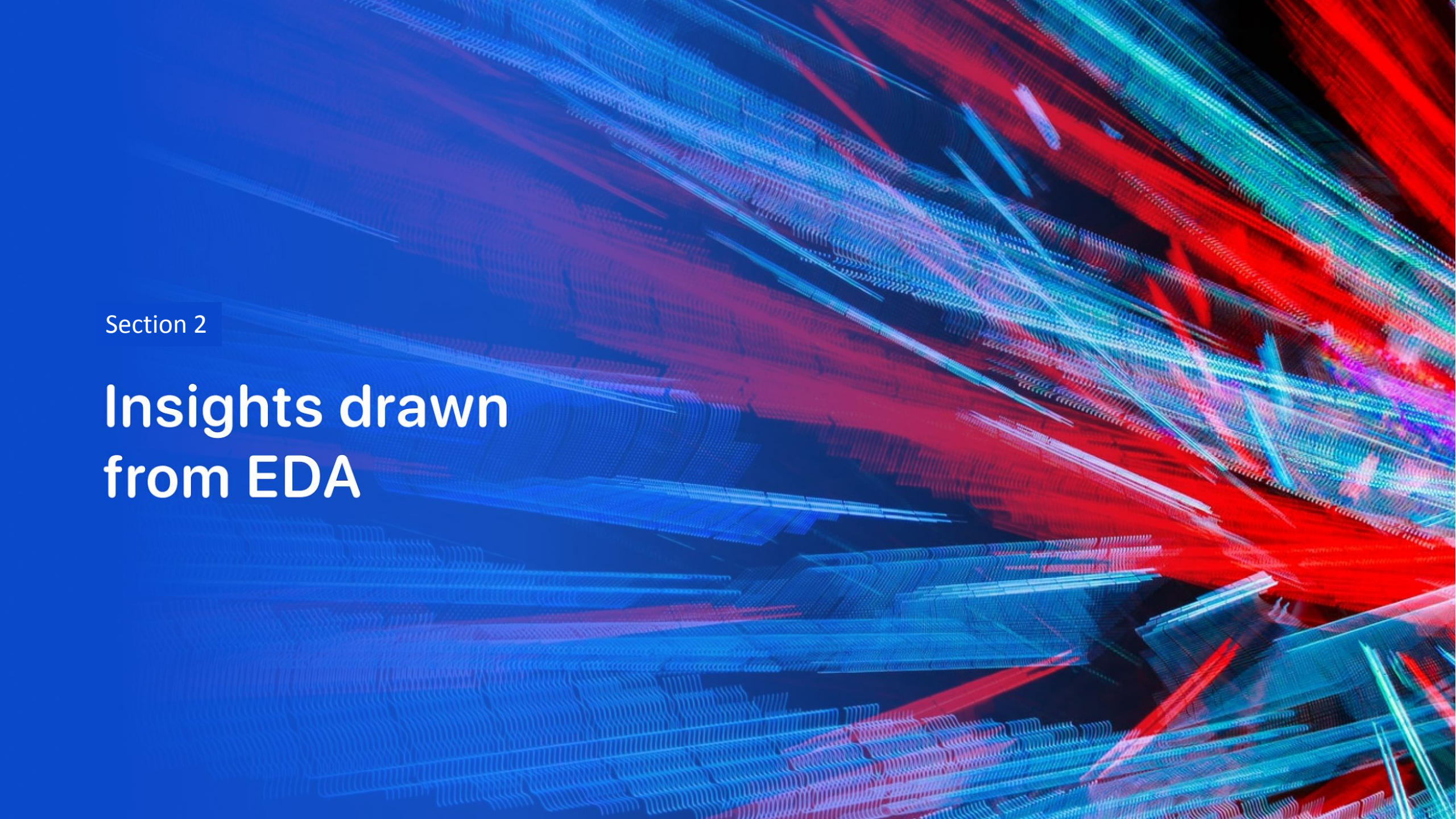
- GitHub URL of the notebook: https://github.com/Carlos-Buitrago/IBM-DS-Capstone/blob/main/jupyter-labs-machine-learning-prediction.ipynb

Results

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

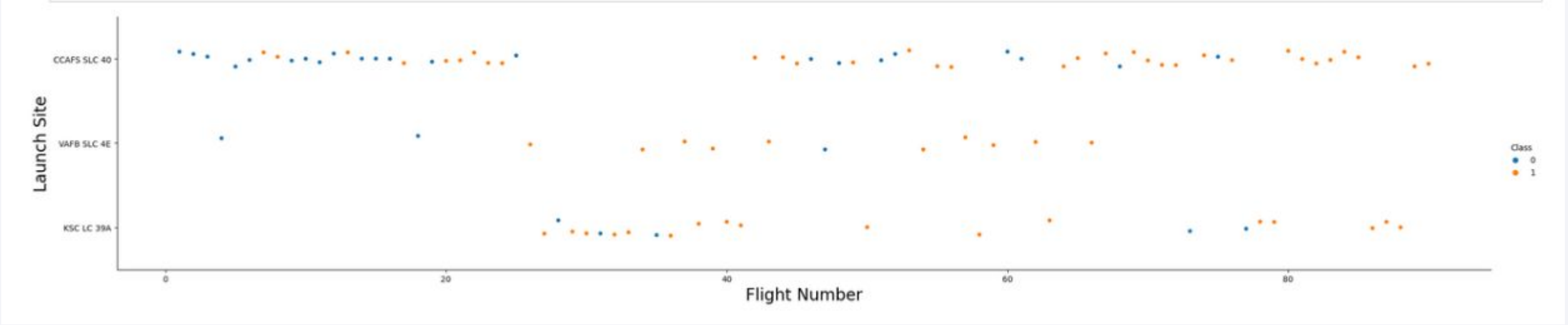
Section 2

Insights drawn from EDA



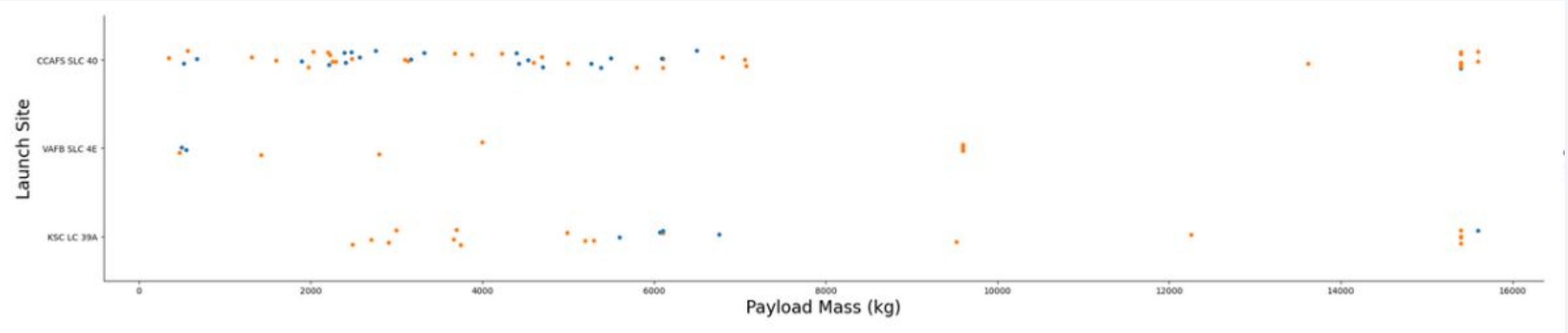
Flight Number vs. Launch Site

- From the plot we can see that smaller flight numbers were done in launch site CCAFS SLC 40 and were mostly unsuccessful. In general, higher flight numbers were more successful.



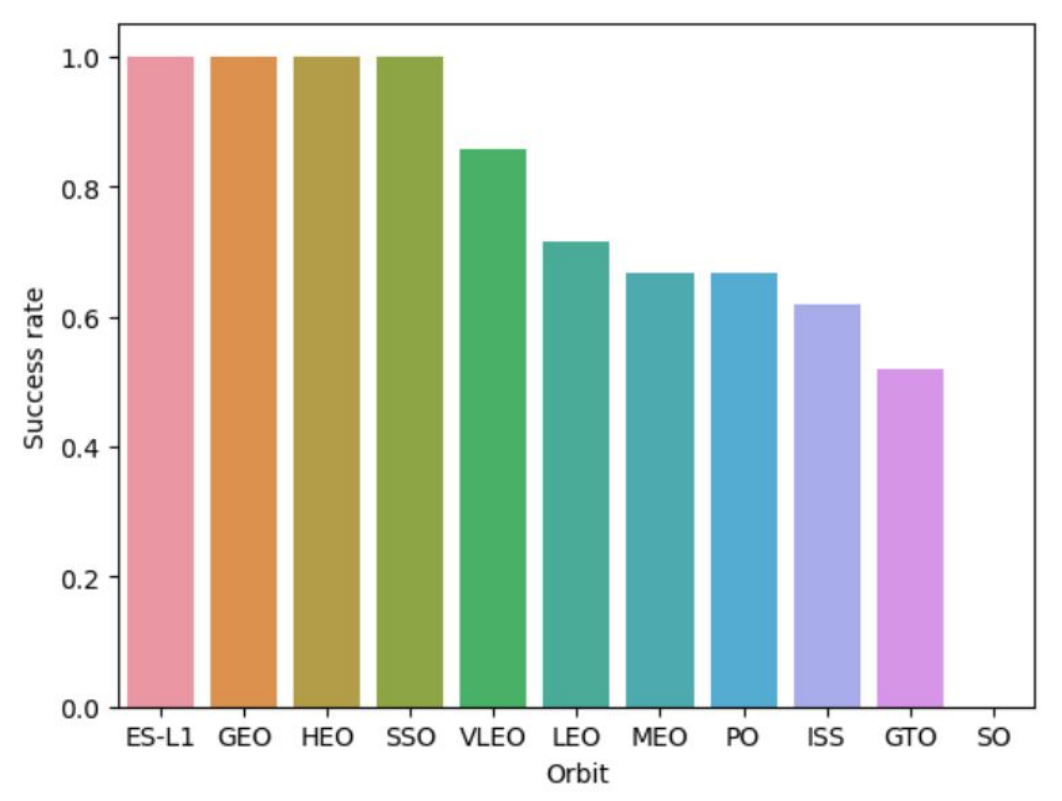
Payload vs. Launch Site

- For heavier payload masses the success rate is greater. Most launches were made with lighter payload masses. For the VAFB-SLC launch site there are no rockets launched with heavy payload mass.



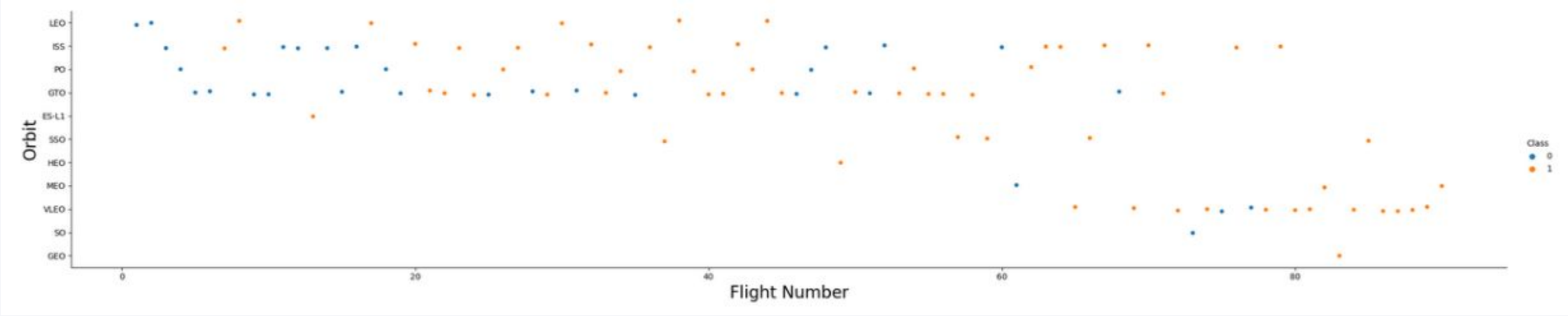
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO and SSO orbits had the highest success rate (of 1.0). SO orbit had the lowest one (of 0.0).



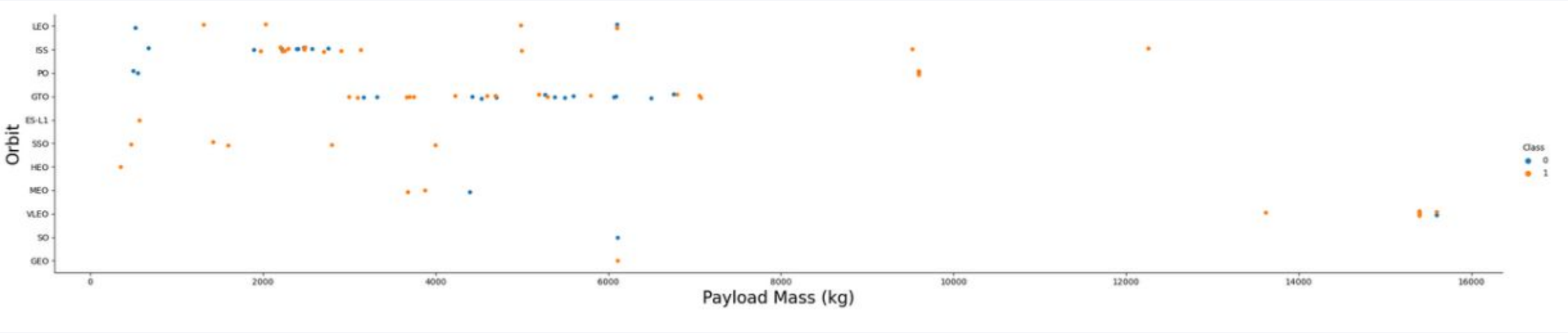
Flight Number vs. Orbit Type

- The LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit



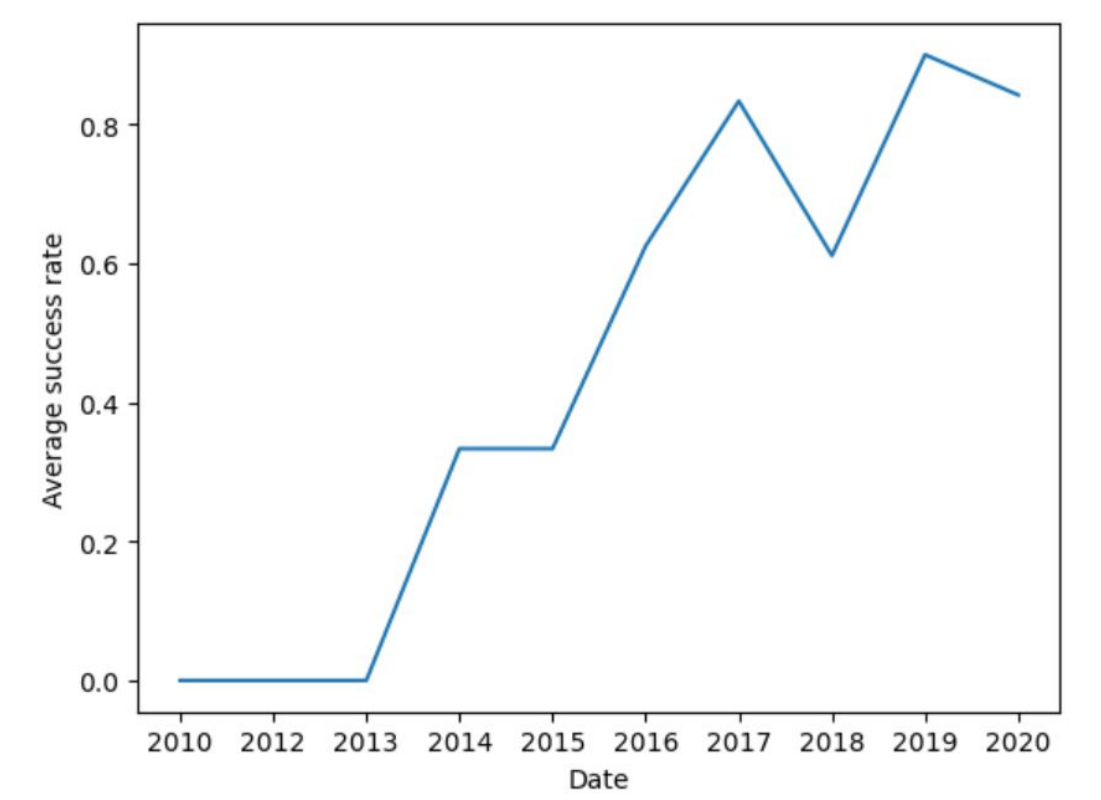
Payload vs. Orbit Type

- With heavy payloads the rate of successful landings is greater for Polar,LEO and ISS. For GTO we cannot distinguish this well.



Launch Success Yearly Trend

- The success rate kept increasing since 2013 until 2020.



All Launch Site Names

- DISTINCT is used to obtain unique entries in the Launch_Site column.

```
%sql SELECT DISTINCT "Launch_Site" from SPACEXTABLE
```

```
 * 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'

- LIKE is used in order to obtain records where the Launch_Site entry begins with CCA. LIMIT is used to display the first 5 records.

```
%sql SELECT * FROM SPACEXTABLE 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_
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (p
2010-08-12	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 (p
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

Total Payload Mass

- SUM is used in order to obtain the total value of the sum of all the entries in the PAYLOAD_MASS_KG column, for which the customer was NASA

```
%sql SELECT SUM(PAYLOAD_MASS__KG_       ) FROM SPACEXTABLE WHERE "Customer" == "NASA (CRS)"
```

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

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- AVG is used in order to obtain the average value of the sum of all the entries in the PAYLOAD_MASS_KG column, for which the booster version was F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_       ) FROM SPACEXTABLE WHERE "Booster_Version" LIKE "F9 v1.1"
```

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

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- MIN is used to obtain the lowest (first) date for which the landing outcome was a success (ground pad).

```
%sql SELECT MIN("Date") FROM SPACEXTABLE WHERE "Landing_Outcome" == "Success (ground pad)"
```

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

MIN("Date")
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- DISTINCT is used to obtain non repeated booster versions and BETWEEN to specify the payload mass range.

```
%sql SELECT DISTINCT "Booster_Version" FROM SPACEXTABLE WHERE "Landing_Outcome" == "Success (drone ship)" AND (PAYLOAD_MASS_
```

BETWEEN 4000 AND 6000

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

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

Total Number of Successful and Failure Mission Outcomes

- COUNT is used to obtain the number of entries with the specified mission outcome.

```
%sql SELECT COUNT("Mission_Outcome") AS "Successful_Misions" FROM SPACEXTABLE WHERE "Mission_Outcome" LIKE "Success%"
```

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

Successful_Misions
100

```
%sql SELECT COUNT("Mission_Outcome") AS "Failed_Misions" FROM SPACEXTABLE WHERE "Mission_Outcome" LIKE "Failure%"
```

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

Failed_Misions
1

Boosters Carried Maximum Payload

- A subquery is used in order to use the MAX of payload mass as a condition clause.

```
%%sql SELECT "Booster_Version", PAYLOAD_MASS__KG_ FROM SPACEXTABLE
    WHERE PAYLOAD_MASS__KG_ == (SELECT MAX(PAYLOAD_MASS__KG_ ) FROM SPACEXTABLE)
```

```
 * 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

- SQLite doesn’t support month names, therefore SUBSTR was used in the Date column to specify the months and years.

```
%%sql SELECT SUBSTR("Date",6,2) AS "Month", SUBSTR("Date",0,5) AS "Year", "Landing_Outcome", "Booster_Version", "Launch_Site
    WHERE "Landing_Outcome" == "Failure (drone ship)" AND SUBSTR("Date",0,5) == '2015'
```

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

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

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

- The records were grouped by Landing outcome in order to count the number of entries in each category. ORDER BY is used with DESC to display the ranking in descending order

```
%%sql SELECT "Landing_Outcome", COUNT("Landing_Outcome") FROM SPACEXTABLE
    WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY "Landing_Outcome"
    ORDER BY COUNT("Landing_Outcome") DESC
```

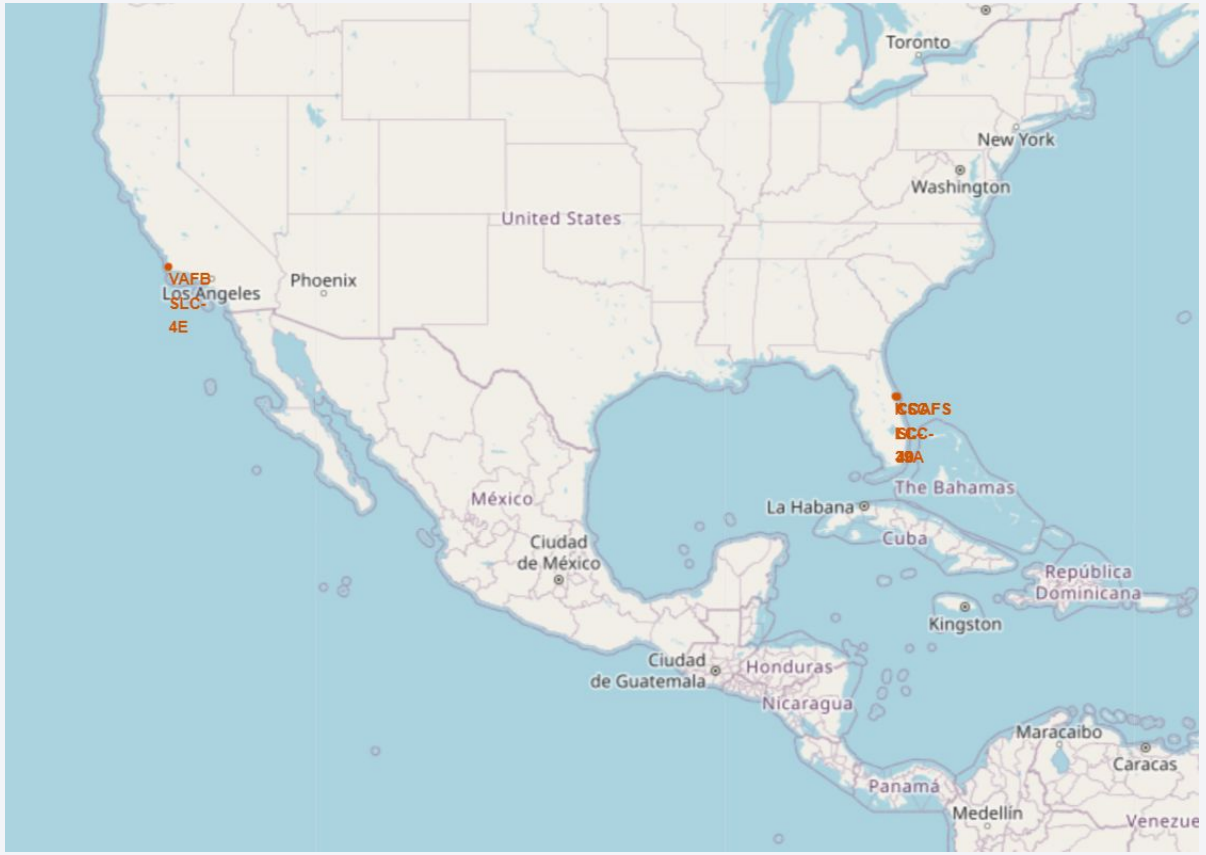
```
 * sqlite:///my_data1.db
Done.
```

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


Section 3
Launch Sites
Proximities Analysis

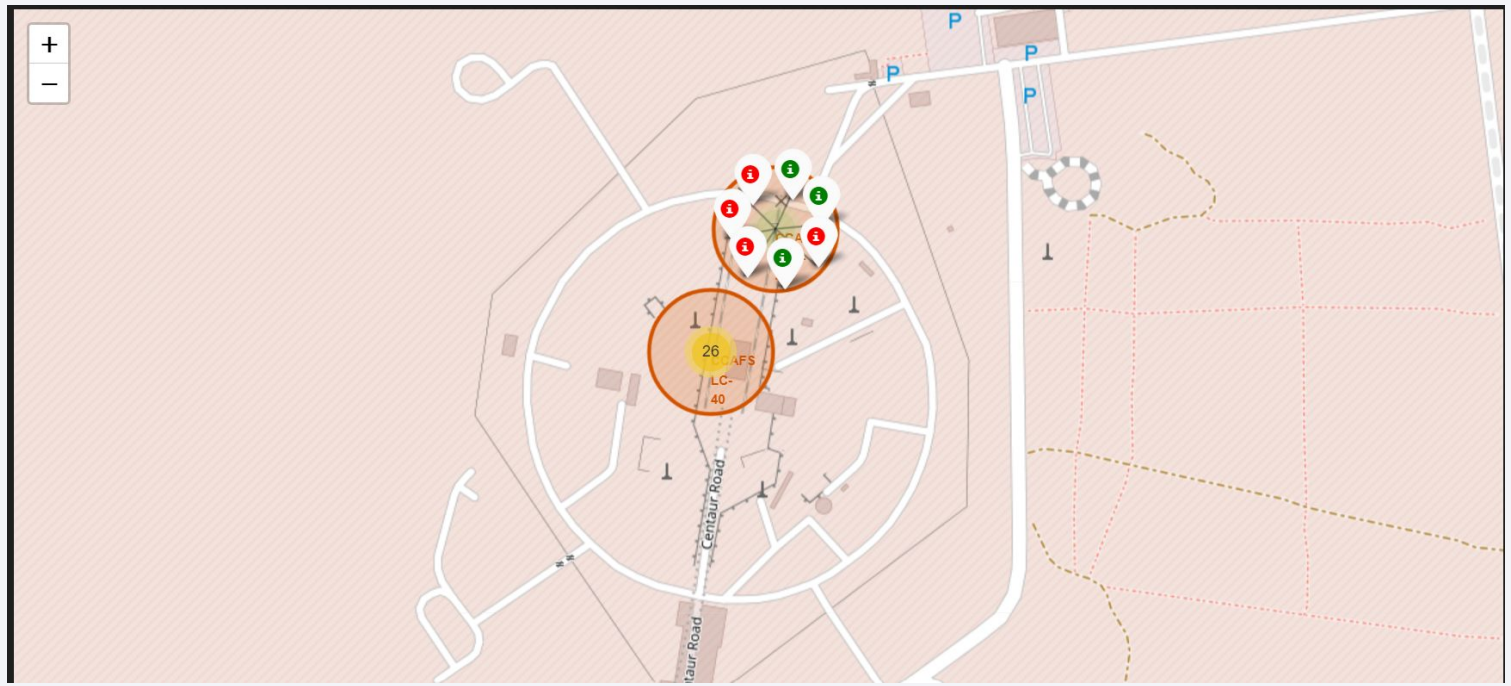
SpaceX launch sites

- The different launch sites were added to the map using circles and markers.



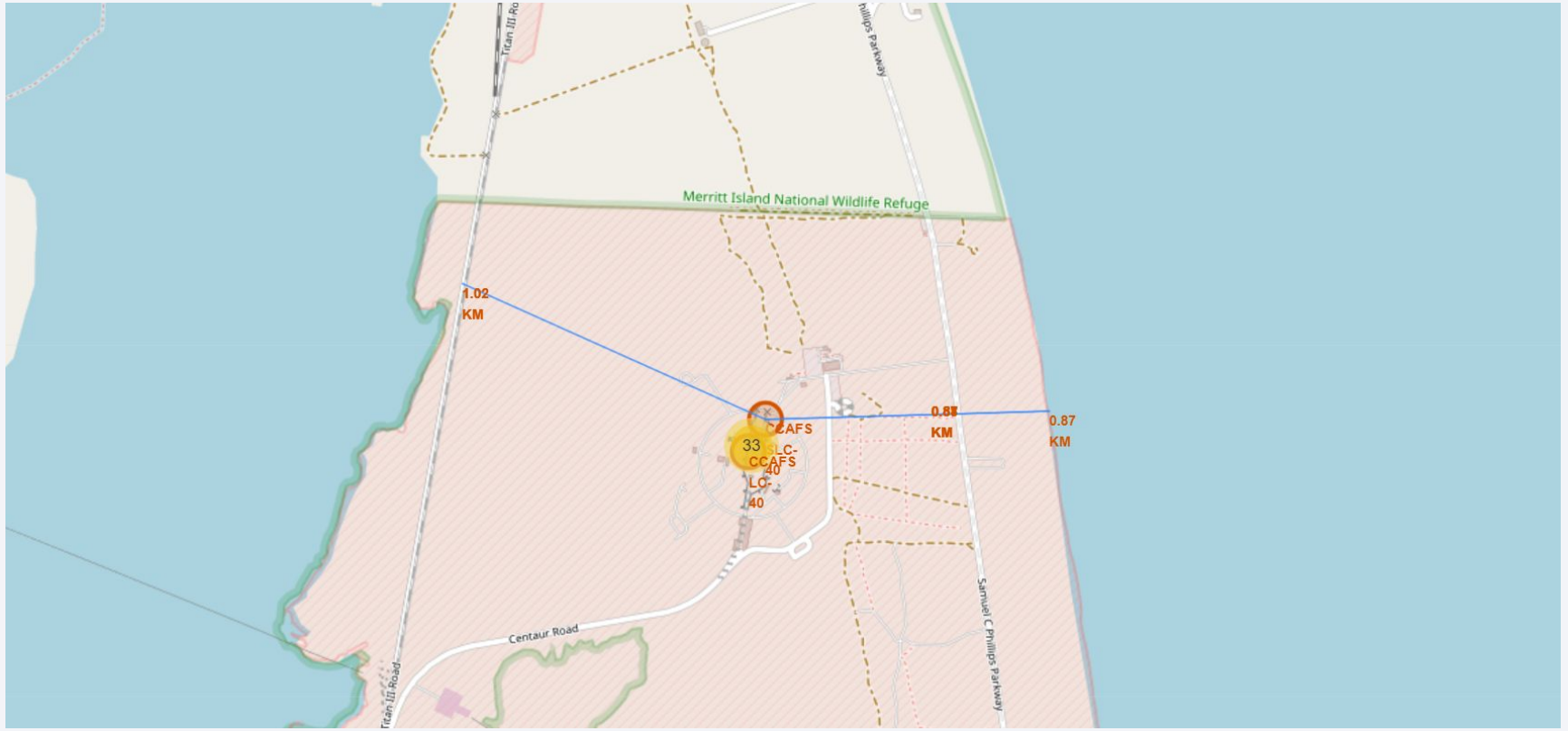
Launch results for each launch site

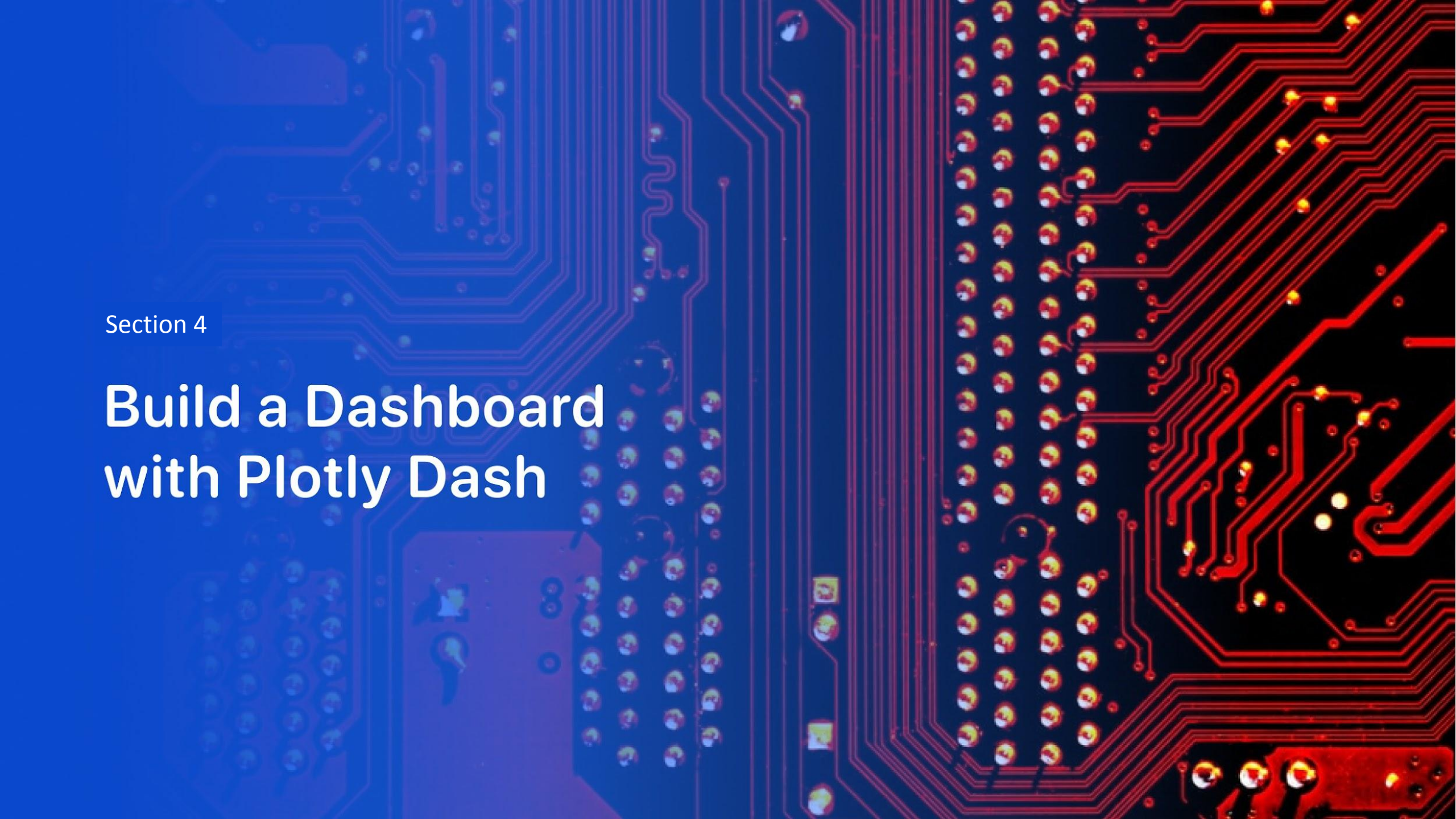
- The success or failure of each launch were displayed using marker clusters for each launch site, since they had the same coordinate



Distance from launch site to proximities

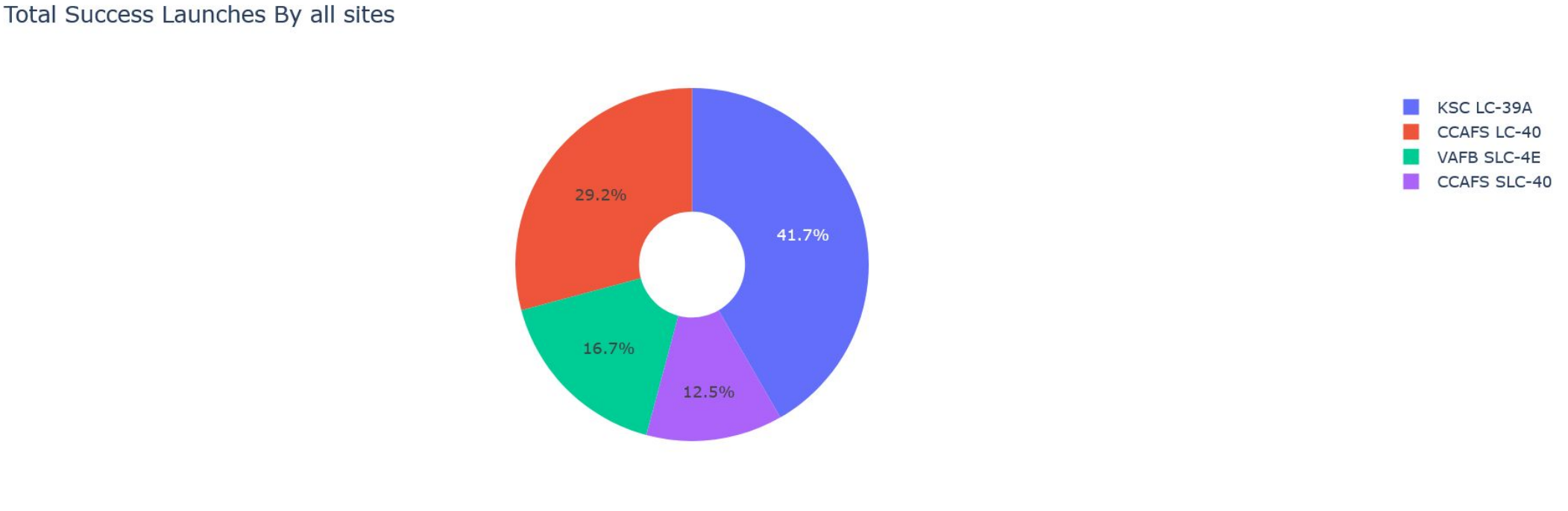
- The distance from launch site CCAFS SLC 40 to the coastline and to a railroad was calculated and drawn by adding a PolyLine element to the map.




Section 4
Build a Dashboard with Plotly Dash

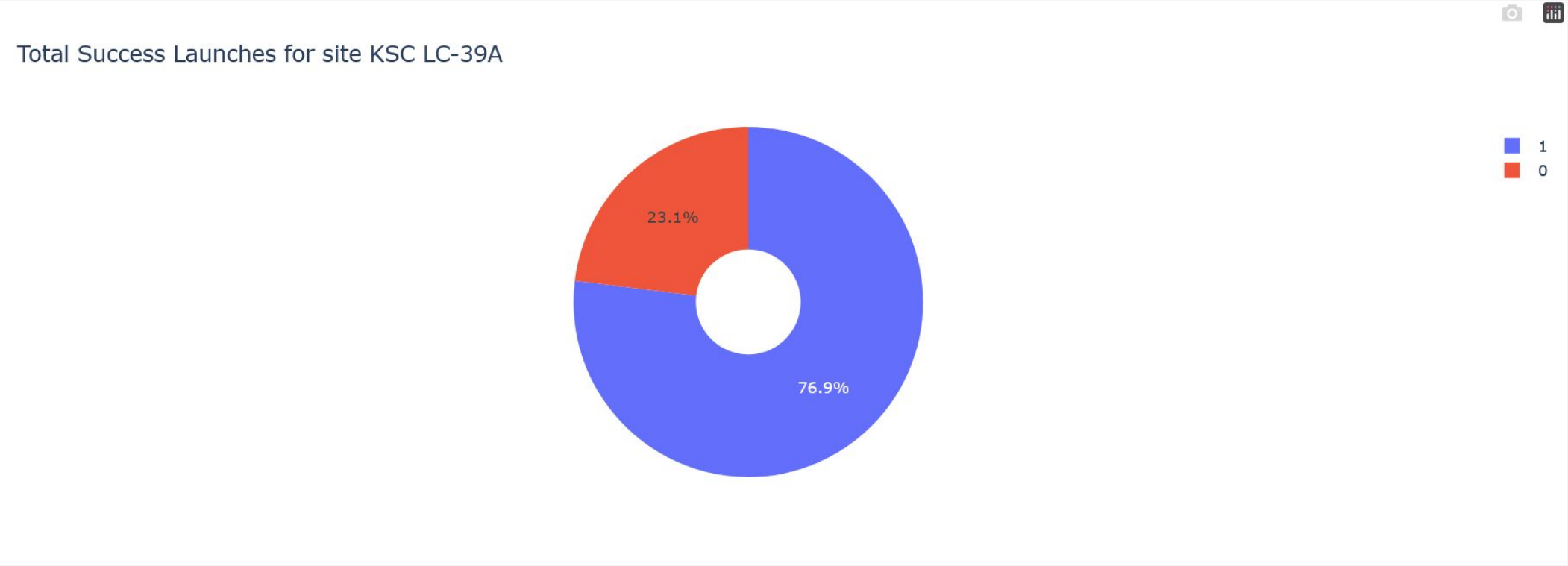
Total success launches for all sites

- A pie chart is displayed showing the total success launches by all sites. KSC LC-39A had the biggest number of successful launches.



Success ratio for most successful launch site

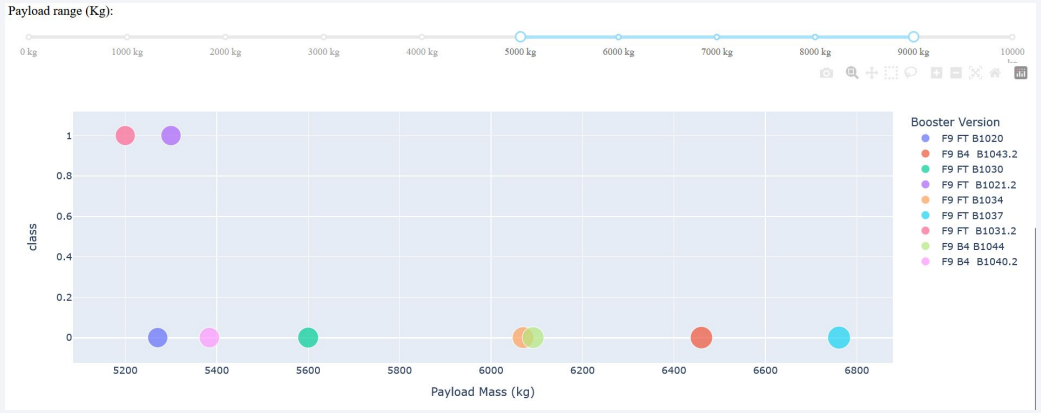
- KSC LC-39A had the highest success rate, it was 76.9% as seen on the pie chart below.



Payload vs. Launch outcome

- Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider. There were more successful launchesr for records with lower payload mass.

Payload mass range: 1000 - 4000 kg



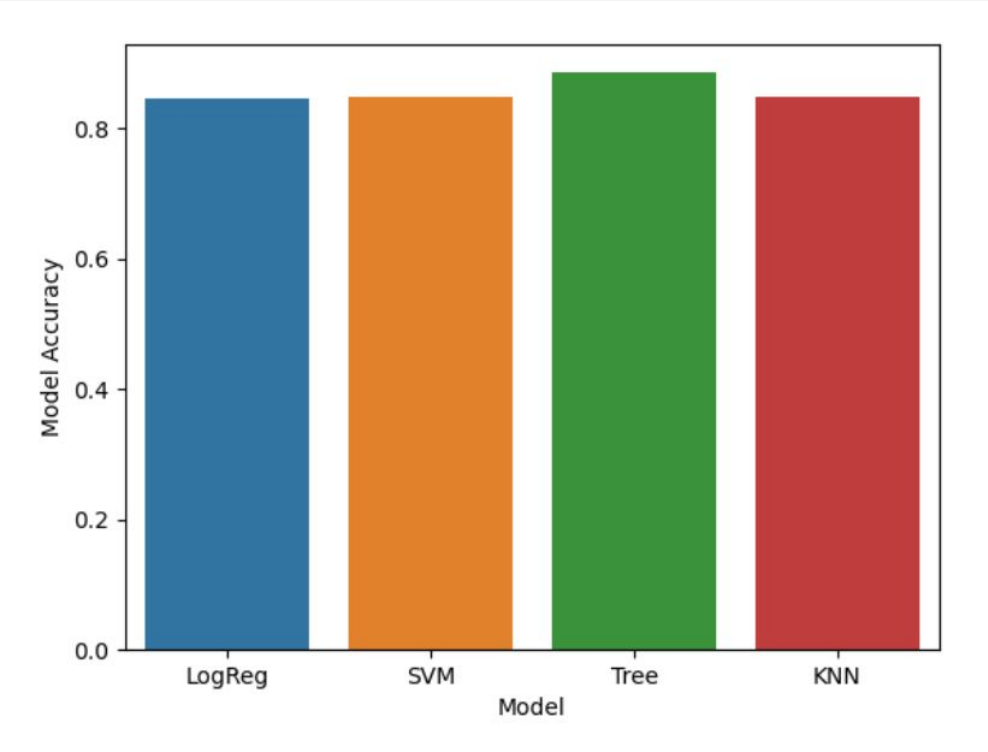
Payload mass range: 5000 - 9000 kg

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Model accuracy for different models trained. The best performing model was the decision tree, it's accuracy and best parameters are specified on the right.

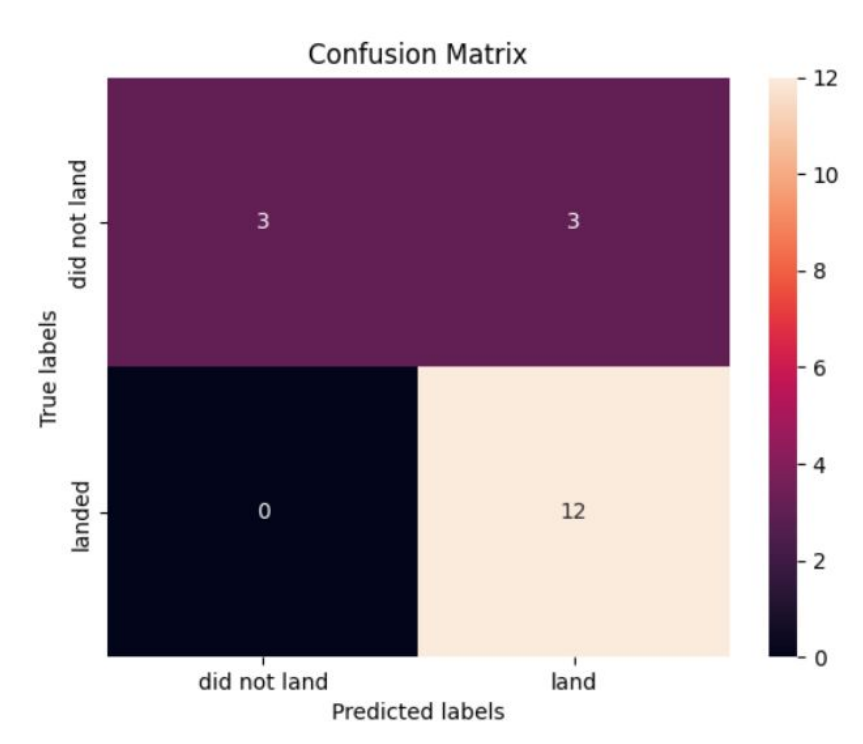



```
best_score = max(logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_)

if best_score == logreg_cv.best_score_:
    print('The best model is Logistic Regression with parameters: ', logreg_cv.best_params_)
elif best_score == svm_cv.best_score_:
    print('The best model is Support Vector Machine with parameters: ', svm_cv.best_params_)
elif best_score == tree_cv.best_score_:
    print('The best model is Decision Tree with parameters: ', tree_cv.best_params_)
elif best_score == knn_cv.best_score_:
    print('The best model is K Nearest Neighbors with parameters: ', knn_cv.best_params_)
```

```
The best model is Decision Tree with parameters:  {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt', 'min_sampl
es_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
```

Confusion Matrix

- The confusion matrix for the best performing model (decision tree) shows that every successful landing was classified as such, however the model classified half of the unsuccessful landings as successful (false positives).



Conclusions

- For heavier payload masses the success rate is greater. Most launches were made with lighter payload masses.
- ES-L1, GEO, HEO and SSO orbits had the highest success rate.
- The success rate kept increasing since 2013 until 2020.
- KSC LC-39A had the biggest number of successful launches and the highest success rate.
- The best performing machine learning model for the classification was the decision tree.
- The missclassifications corresponded to false positives.

Thank you!