SpaceX

Landing Analysis

A.G.SIBIKRISHNAN
APRIL 2024

Outline

- **Executive Summary**
- **Introduction**
- **Methodology**
- **Results**
  - Exploratory Data Analysis (EDA) with Visualisation
  - EDA with SQL
  - Interactive Maps with Folium
  - Plotly - Dash Dashboard
  - Predictive Analytics
- **Conclusion**

Executive Summary

Summary of methodologies:

The project's aim is to identify factors of a successful rocket landing. Steps to identify them:

- Data **Collection** using SpaceX **REST API** and **web scraping** techniques (Beautiful Soup)
- Data **Wrangling** to create success/fail outcome variable.
- **Exploration** with **data visualisation** techniques - payload, launch site, flight number and yearly trend.
- **Analyse** data with **SQL**, calculating statistics such as Total Payload, Payload range for successful launches, and total number of successful and failed outcomes.
- **Geospatial Exploration** to make connections between launch site success rates and proximity to geographical markers.
- **Visuals** for launch sites with most success and successful payload ranges.
- Use of **models** to predict landing outcomes. Models – **Logistic regression, Support Vector Machine(SVM), decision tree and K-nearest Neighbour (KNN)**.

Executive Summary

Summary of all results

- **Exploratory Data Analysis**
  - Success rate for launches has increased over time
  - The ES-L, GEO, HEO, and SSO Orbits have a 100% success rate
  - KSC LC-39A has the highest success rate among landing sites
- **Analytics and Visualisations**
  - The equator seems to be a preferred spot for launch sites as most are located near the equator and are in close proximity to the coast.
- **Analytics from Prediction Models**
  - While all model performed similarly to each other on the prepared test set, the decision tree model outperformed the other models.

Introduction

Background

- SpaceX is a private aerospace manufacturer and space transportation company.
- It aims to revolutionize space technology, with the notable achievements of - sending spacecraft to the international space station, launching a satellite constellation to provide internet access and sending manned missions to space.
- **SpaceX** can carry out these missions for **relatively lower costs**, approx. **$62 Million per launch**, due to its inventive **reuse** of its Falcon 9's **first-stage booster**.
- **Competitors** in the space without the reusable booster capability, such as **ULA** with their **Vulcan** rockets, charge approx. **$110 Million per launch**.
- By determining if the first stage will land, we can determine the price of the launch. To do this, we can use public data and machine learning models to predict whether SpaceX – or a competing company – can reuse the first stage.

Introduction

Areas to explore:

- How the first-stage booster landing success is affected by factors such as: **Payload Mass, Launch Site, Number of flights and Orbit type**.
- Rate of **Successful Landings over time**.
- Applying One-Hot encoding for the categorical data, we build the most optimal predictive model, that performs binary classification, for **successful landing.**

METHODOLOGY

METHODOLOGY

- Data **Collection** [SpaceX REST API + Web Scrapping]
- Data **Wrangling**
- Data **Exploration** [EDA with SQL + Data Visualisation]
- **Interactive Data Visualisations** [Folium + Plotly Dash]
- Build **Predictive Models** to predict landing outcomes using classification models. Models were tuned and evaluated to best optimise them and their parameters.

Data Collection – SpaceX API

1. **Request** rocket launch data from API
2. Using .json(), **decode** the response and **convert to dataframe** using .json_normalize()
3. Use custom functions to **request specific information** about launches from the API
4. Create a **dictionary** from the data
5. **Filter** the dataframe to only contain Falcon 9 Launches
6. **Deal with missing values** – replace missing Payload Mass with calculated .mean() of column.
7. **Export** the data to a **.csv** file

[1]:
```
# Requests allows us to make HTTP requests which we will use to get data from an API
import requests
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
# NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a larg
import numpy as np
# Datetime is a library that allows us to represent dates
import datetime

# Setting this option will print all collumns of a dataframe
pd.set_option('display.max_columns', None)
# Setting this option will print all of the data in a feature
pd.set_option('display.max_colwidth', None)
```

Below we will define a series of helper functions that will help us use the API to extract information using identification numbers in the launch data.

From the `rocket` column we would like to learn the booster name.

[2]:
```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

[3]:
```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

[4]:
```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, wheter the core is reused, wheter legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

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

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

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

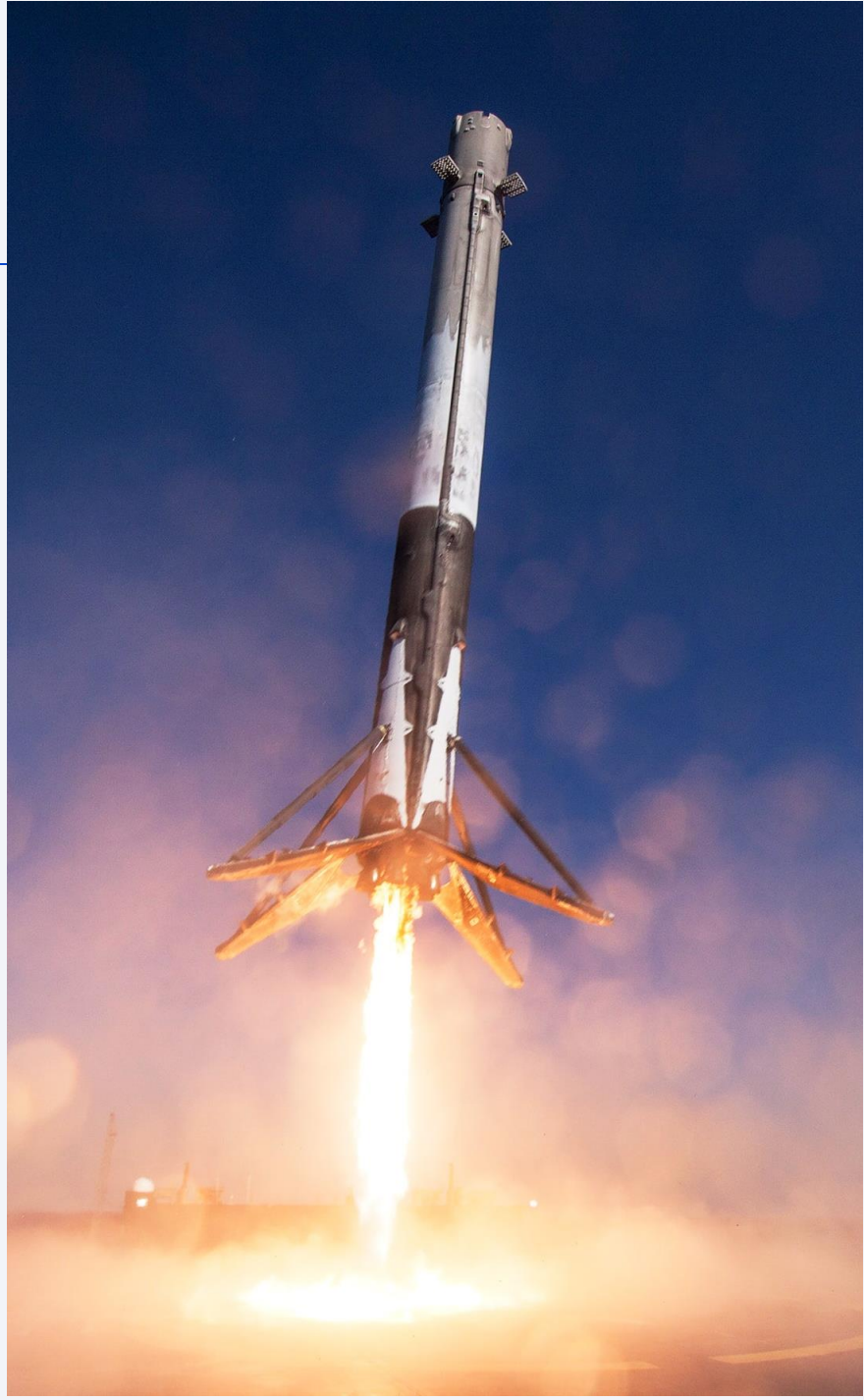
[7]:
```
response = requests.get(spacex_url)
```

Check the content of the response

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

Data Collection - Scraping

1. **Request** Falcon 9 Launch Data from Wikipedia.
2. Create a **BeautifulSoup object** from the HTML Response.
3. From HTML Table Headers, **extract Column Names**
4. Collect data from **parsing** the **HTML tables**.
5. Create a **dictionary** from the data
6. Create **dataframe** from the dictionary.
7. **Export** data to **.csv** file



Data Wrangling –1

Steps

- Perform **Exploratory Data Analysis** and determine data labels.
- **Calculate**:
  - Number of launches of each site
  - Occurrence frequency of the different orbits
  - Occurrence frequency of mission outcome per orbit type
- Data manipulation - create **binary landing outcome column**, acting as the **dependent variable**.
- **Export Data** to .csv file.

Data Wrangling – 2 [Landing Outcomes]

There are successful and unsuccessful landings of multiple variations. This is the breakdown:

- **True Ocean**: Successful mission outcome with landing on a specific region of the ocean.
- **False Ocean**: Unsuccessful mission outcome w.r.t landing on a specific region of the ocean.
- **True RTLS**: Successful mission landing on ground pad.
- **False RTLS**: Unsuccessful mission landing on ground pad.
- **True ASDS**: Successful landing on drone ship.
- **False ASDS**: Unsuccessful landing on drone ship.
- **Conversion of outcomes**: 1 for successful landing and 0 for unsuccessful landings.

EDA with Data Visualization

Analysis:

- Using scatter plots to view relationships between different variables. If a relationship is seen, the variable can potentially be used for machine learning purposes.
- Using Bar charts to compare different discrete categories.

Charts Used:

- Flight Number vs Payload
- Flight Number vs Launch Site
- Payload Mass(kg) vs Launch Site
- Payload Mass(kg) vs Orbit type

EDA with SQL

Display from Queries:

- Distinct (Unique) names of launch sites
- 5 Records of where launch site name began with 'CCA'
- Total payload mass carries by boosters launched by NASA (CRS)
- Average payload mass carries by booster version F9 v1.1

List derived from Queries:

- Date of first successful landing on ground pad
- Name of boosters which had successful landing on drone ship and had payload mass more than 4000kg but less than 6000kg
- Total number of successful and failed missions
- Names of booster versions which have carried the max payload
- Count of landing outcomes (Date range: 2010-06-04 to 2017-03-20)
- Failed landings on drone ship, their respective booster versions and launch site for the months in year 2015.

Build an Interactive Map with Folium

- Launch Sites indicated using **Markers**
  - Blue circle has been added around NASA Johmson Space Center's coordinate, with popup label indicating its name using its coordinates - latitude and longitude.
  - Red circle added at all launch site coordinates with popup label indicating its name using its coordinates - latitude and longitude.
- Added **green** markers for **successful** launches and **red** markers for **unsuccessful** launches at each launch site to make the sites with higher success rate more apparent.
- **Coloured lines** have been added to show **distance between launch site** CCAFS SLC-40 and its proximity **to nearest coastline, railway, highway and city.**

Build a Dashboard with Plotly Dash

Features

- **Dropdown list**
  - Allows users to select either all launch sites or a certain launch site.
- **Pie Chart**
  - Users can see successful and unsuccessful missions split as a percentage of the total pie.
- **Slider**
  - Uses can select payload mass range
- **Scatter Chart**
  - Users can see the relationship, aka correlation, between Payload and Launch Success

Predictive Analysis (Classification)

Represented in Chart form, with the following steps:

- Using values from Class column, create a **NumPy array**
- Use StandardScaler.Fit to **standardize** the data and **transform** it.
- **Split** the data using train_test_split to derive data sets to train and test the models.
- **Create** a **GridSearchCV** object with, Cross-validation, cv=10 for parameter optimisation.
- **Apply GridSearchCV** on different algorithms - Logistic Regression, SVM, Decision tree and K-Nearest Neighour.
- Calculate **accuracy** of models on test data using **.score().**
- Assess **confusion matrix** for all models.
- Identify best models using **Jaccard Index, F1 score** and **Accuracy score**.

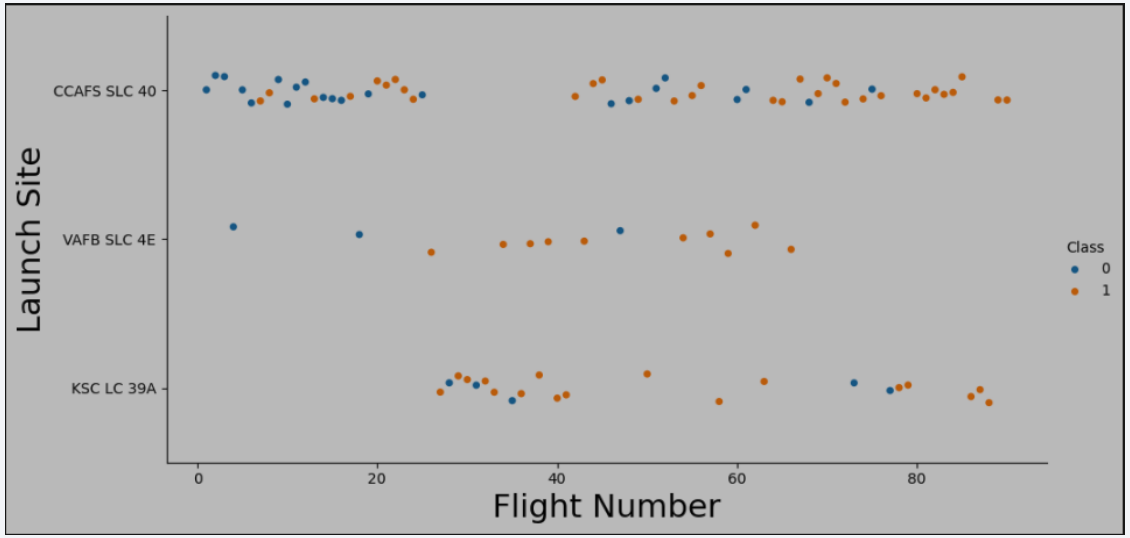
RESULTS

Summary of Results

- **EDA:**
  - Over time, launch success rate has increased.
  - Orbits GEO, HEO, SSO and ES-L1 have a 100% success rate.
- **Visual Analytics:**
  - Most launch sites close to equator and coast
  - Location of launch sites distant enough from infrastructure like cities, highway and railways to not affect them in case of failed launch but close enough to enable movement of manpower and supplies.
- **Predictive Analysis:**
  - Decision Tree model performed the best for this dataset.

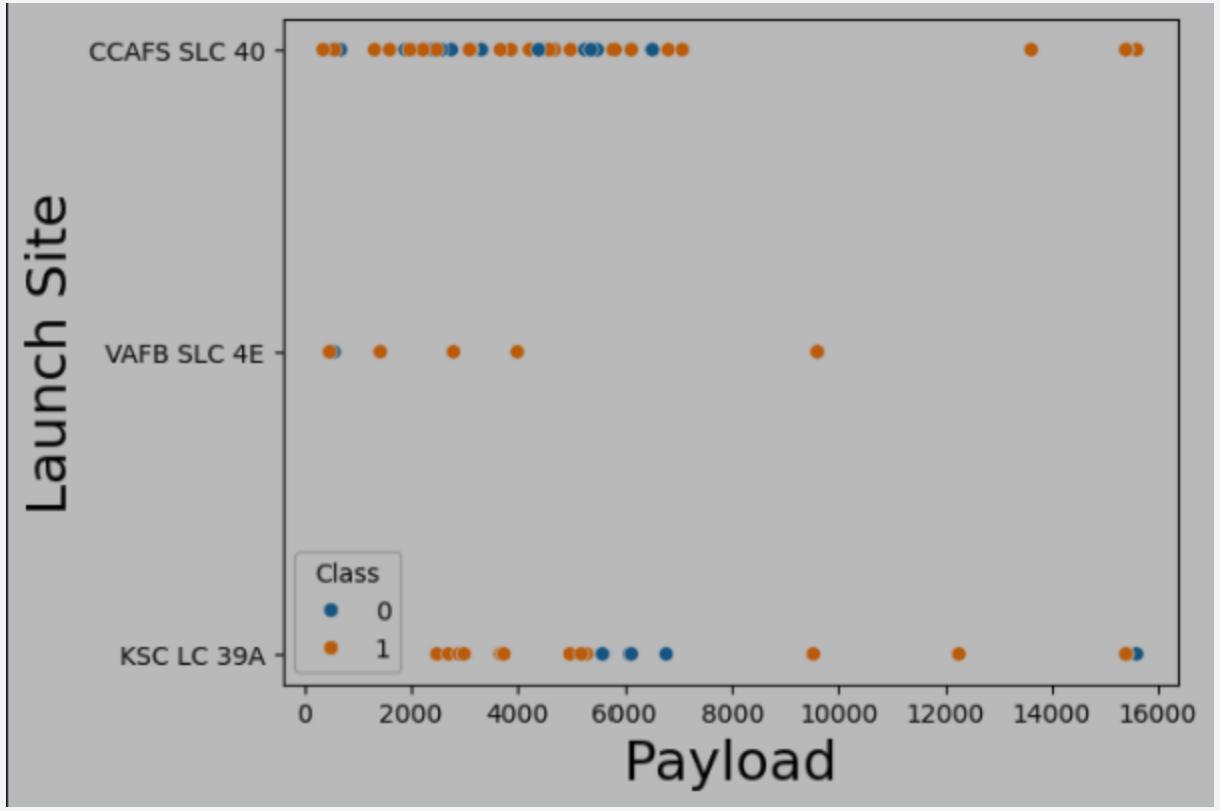
Flight Number vs. Launch Site

- **Earlier** missions had a **lower** success rate (Blue = Unsuccessful)
- **Later** flights had a **higher** success rate (Orange = Success)
- Approx. half of missions were from CCAFS SLC 40
- We can infer that new launches have a higher success rate



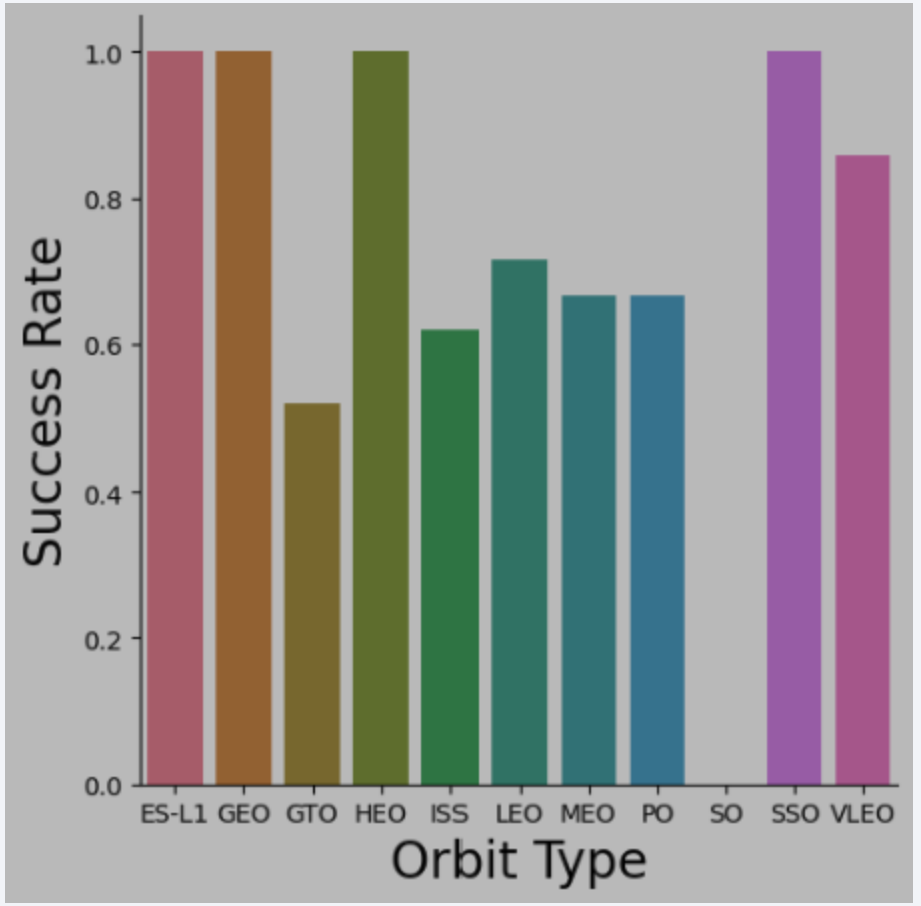
Payload vs. Launch Site

- Higher the payload, more likely for successful launch
- Most launches above 8,000kg were successful
- VAFB SKC 4E has not launched anything greater than ~10,000 kg
- KSC LC 39A has a 100% success rate for launches less than 5,500 kg



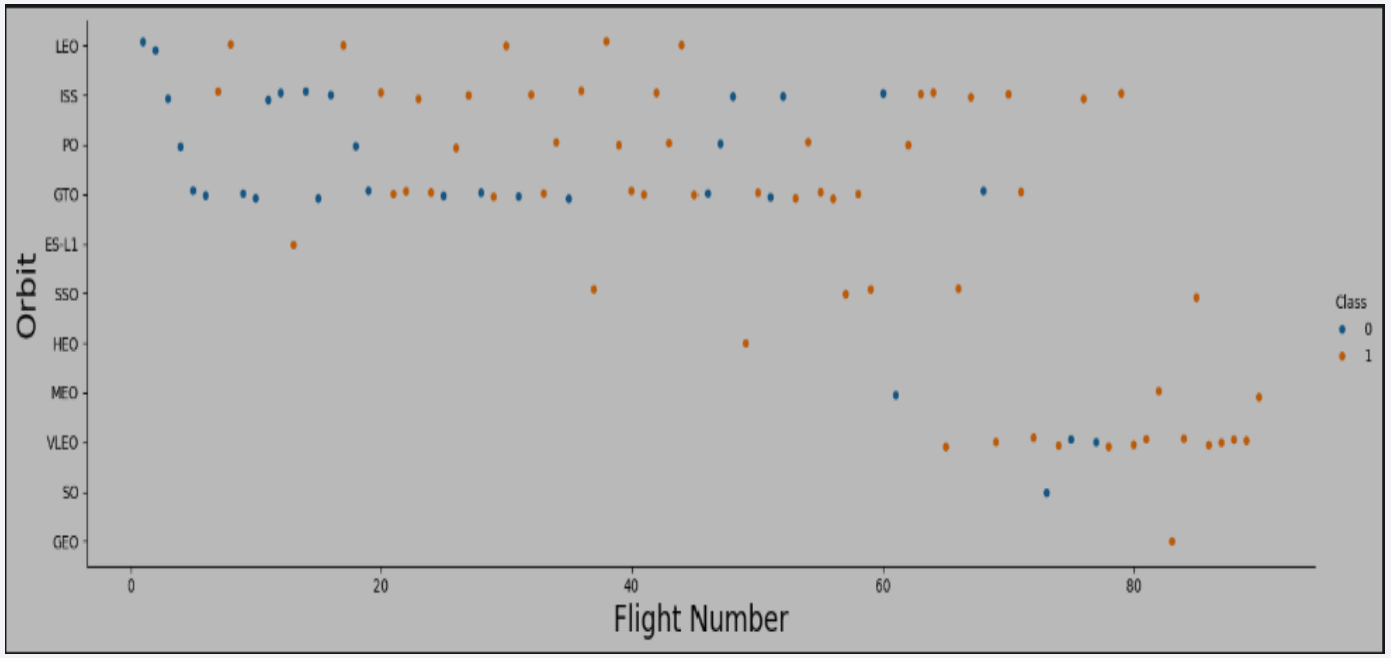
Success Rate vs. Orbit Type

- 0% Success Rate: SO
- 50%-80% Success Rate: GTO, ISS, LEO, MEO, PO
- 100% Success Rate: ES-L1, GEO, HEO and SSO



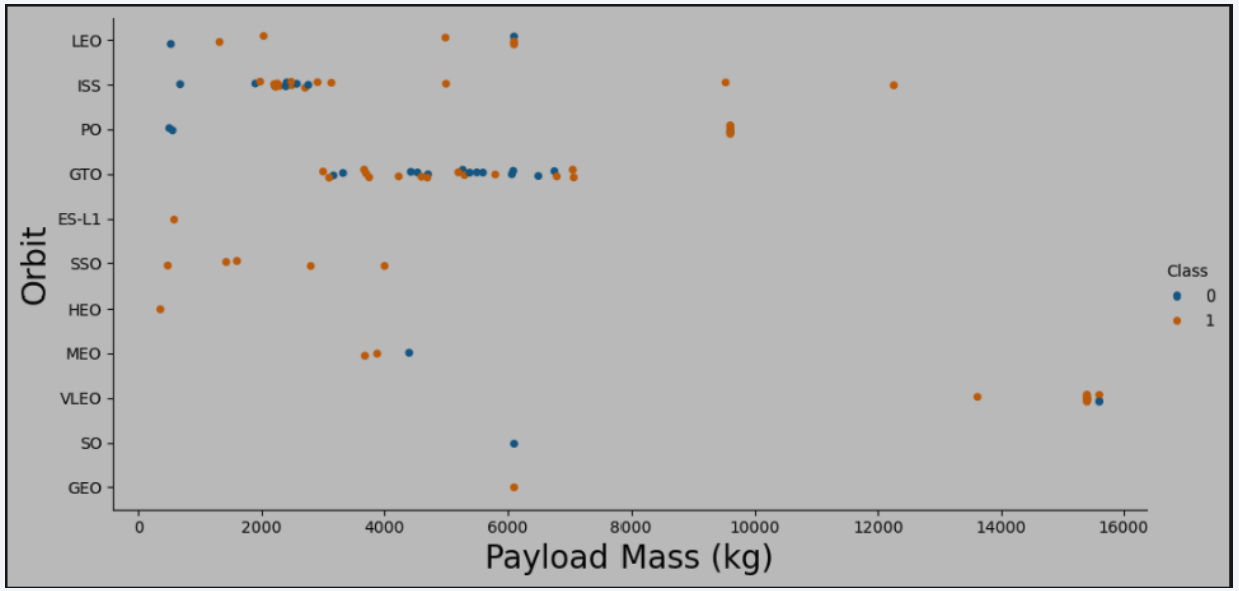
Flight Number vs. Orbit Type

- **Success rate typically increases with the number of flights for each orbit**; highly apparent for the LEO orbit.
- GTO orbit, does not follow the above trend.



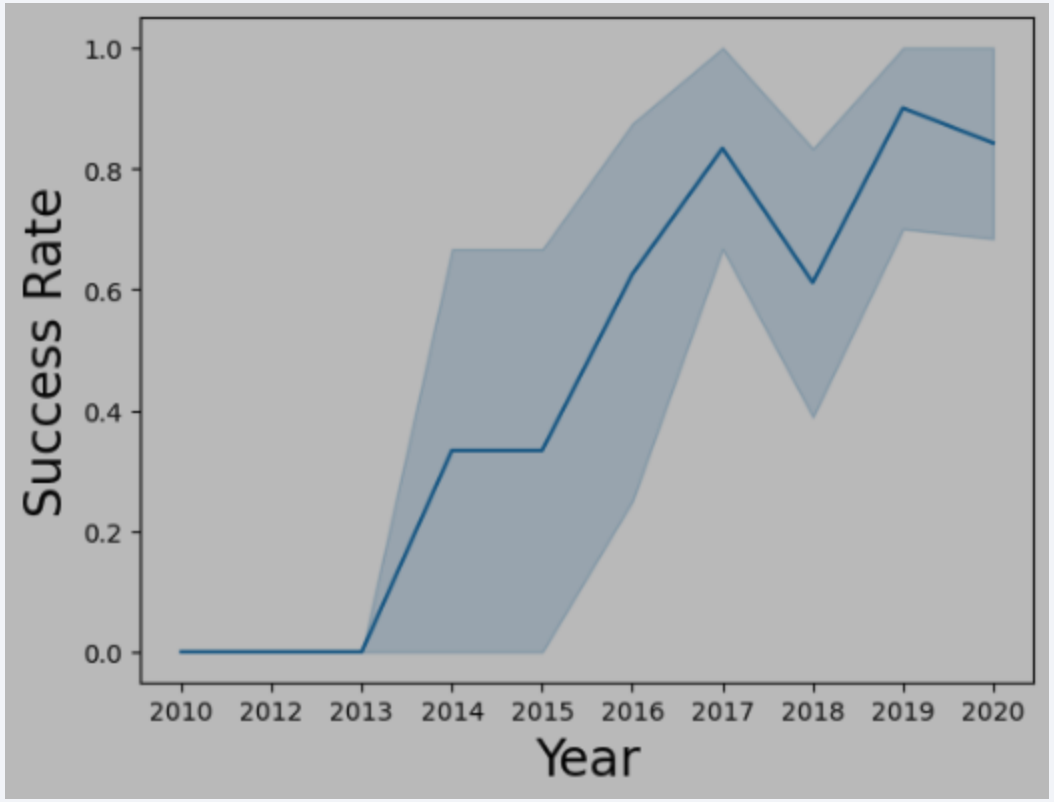
Payload vs. Orbit Type

- Heavy payloads are better with LEO, ISS and PO orbits
- The GTO orbit has mixed success with heavier payloads.



Launch Success Yearly Trend

- Success rate improved from 2013-2017 and 2018-2019
- Success rate decreased from 2017-2018 and from 2019-2020.
- Overall Trend: success rate has improved since 2013



All Launch Site Names (Unique)

Launch Sites:

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

```
[11]: %sql select distinct LAUNCH_SITE from SPACEXTBL;
 * sqlite:///my_data1.db
Done.
[11]:
```

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

Launch Site Names Begin with 'CCA'

5 records where launch sites begin with `CCA`.

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

[12]:

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

Total Payload Mass: **45,596kg**

```
[16]: %sql select sum(PAYLOAD_MASS__KG_) as 'Total payload by NASA CRS' from SPACEXTBL where Customer = "NASA (CRS)";
 * sqlite:///my_data1.db
Done.
```

[16]:

Total payload by NASA CRS
45596

Average Payload Mass by F9 v1.1

Average payload mass carried by booster version F9 v1.1:
2928.4kg

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

```
[17]: %sql select AVG(PAYLOAD_MASS__KG_) as 'Average Payload F9 v1.1' from SPACEXTBL where Booster_Version = "F9 v1.1";
```

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

[17]:

Average Payload F9 v1.1
2928.4

First Successful Ground Landing Date

Date: 22-December-2015

```
[19]: %sql select min(Date) from SPACEXTBL where Landing_Outcome = 'Success (ground pad)'
```

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

[19]:

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000: **JSCAT-14, JSCAT-16, SES-10, SES-11 / EchoStar 105**

```
[20]: %sql select Payload from SPACEXTBL where Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000;
 * sqlite:///my_data1.db
Done.
```

[20]:

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

Total Number of Successful and Failure Mission Outcomes

- 1 Failure in Flight
- 99 Successful
- 1 Successful(payload status unclear)

List the total number of successful and failure mission outcomes

```
[21]: %sql select Mission_Outcome, count(*) as 'Total Number' from  SPACEXTBL group by Mission_Outcome;
```

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

[21]:

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

Boosters Carried Maximum Payload

- F9 B5 B1048.4
- F9 B5 B1049.4
- F9 B5 B1051.3
- F9 B5 B1056.4
- F9 B5 B1048.5
- F9 B5 B1051.4
- F9 B5 B1049.5
- F9 B5 B1060.2
- F9 B5 B1058.3
- F9 B5 B1051.6
- F9 B5 B1060.3
- F9 B5 B1049.7

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

```
[24]: %sql select Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
 * sqlite:///my_data1.db
Done.
```

[24]:

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

2015 Launch Records

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

[33]:
```
%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.
```

[33]:

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

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.

```
[41]: %sql SELECT Landing_Outcome, count(*) as count_outcomes FROM SPACEXTBL WHERE Date between '2010-06-04' and '2017-03-20' group by Landing_Outcome order
```

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

[41]:

Landing_Outcome	count_outcomes
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

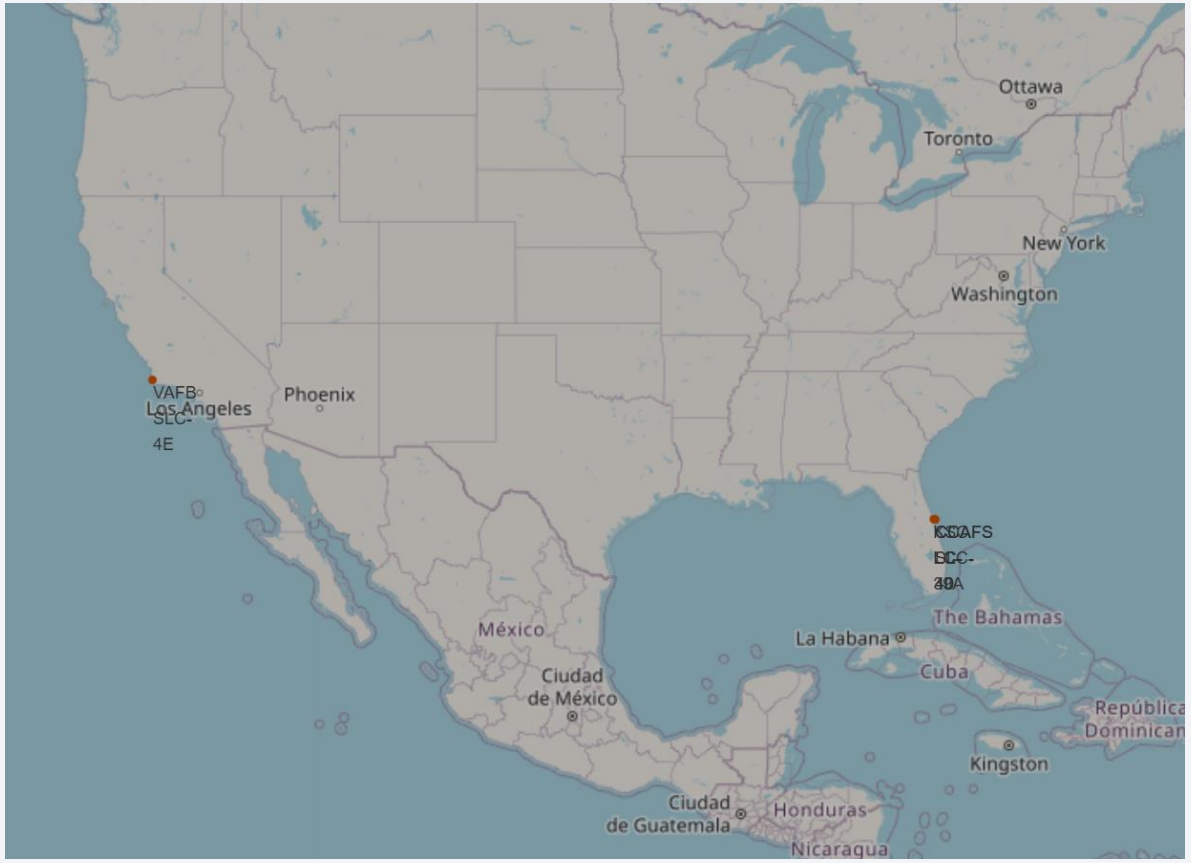
Query:

```
%sql SELECT Landing_Outcome, count(*) as
count_outcomes
FROM SPACEXTBL
WHERE Date between '2010-06-04' and '2017-03-20'
group by Landing_Outcome
order by count_outcomes DESC
```

LAUNCH SITE ANALYSIS

LAUNCH SITES

- Launch sites are placed closer to equator as it allows for a shorter distance to launch into equatorial orbit.
- The angular velocity of the earth's rotation decreases the amount of energy needed for launch and therefore increases cost efficiency due to lesser resource need compared to non-equatorial launches.

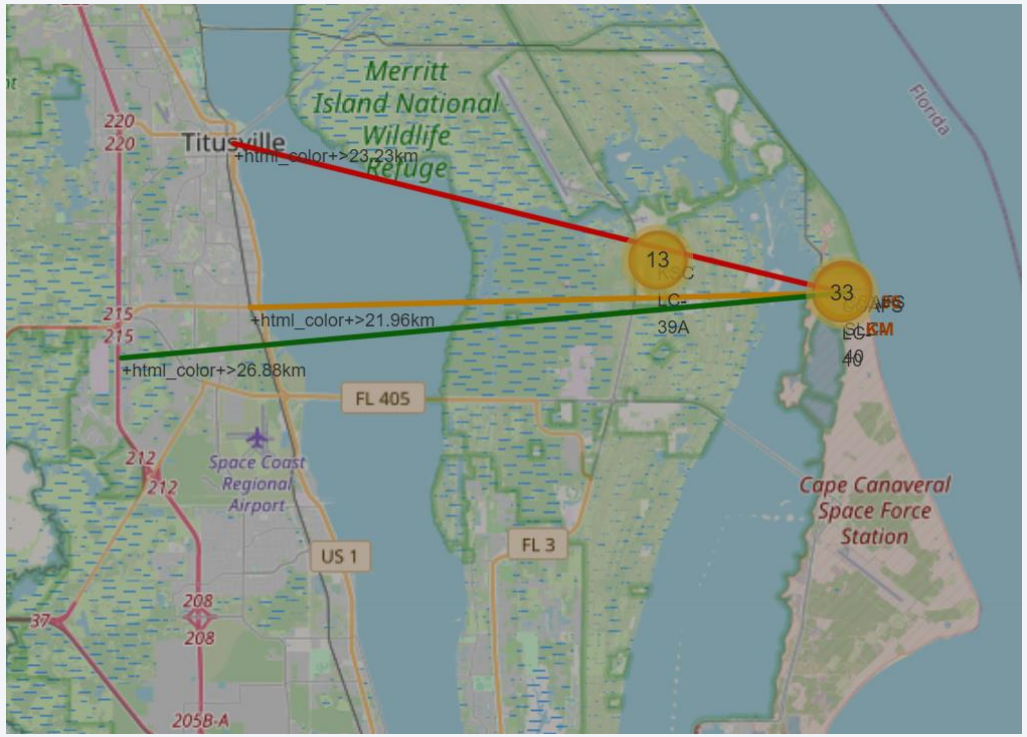


Launch Outcomes

- Launch Site in focus: **CCAFS SLC-40**
- Successful Launches indicated by green marker
- Unsuccessful Launches indicated by red marker.
- This launch site has a ~43% success rate.

Distance to Infrastructure and Coastline – CCAFS SLC-40

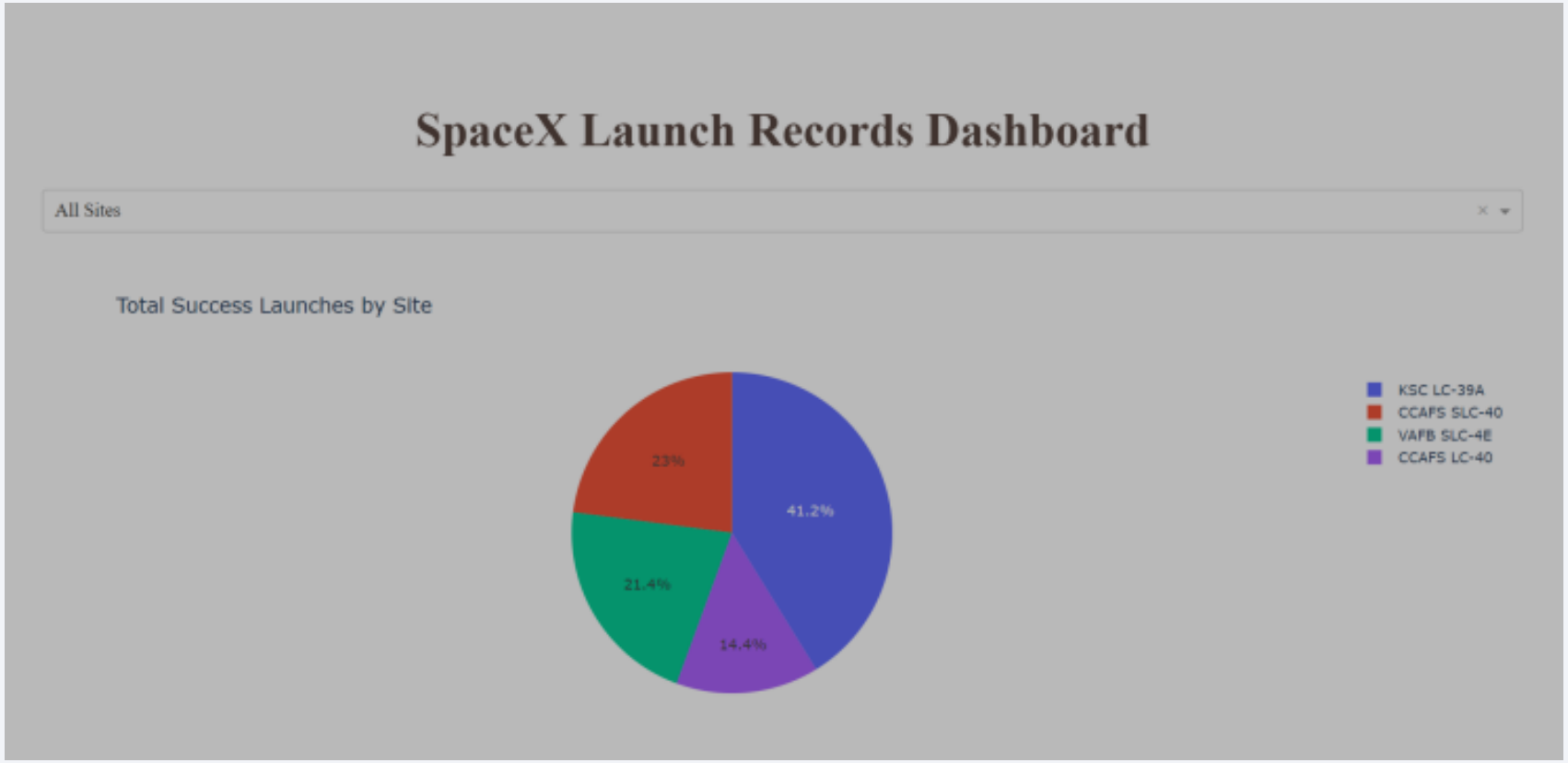
- **21.96 km** from nearest railway
- **23.23 km** from nearest city
- **26.88 km** from nearest highway
- **0.86 km** from nearest coastline
- Near proximity to coast allows for safer dropping of expended stages
- Distance from infrastructure ensures they are protected from any launch fails or by-product while ensuring it is close enough for supply and manpower movement.



DASHBOARD WITH PLOTLY DASH

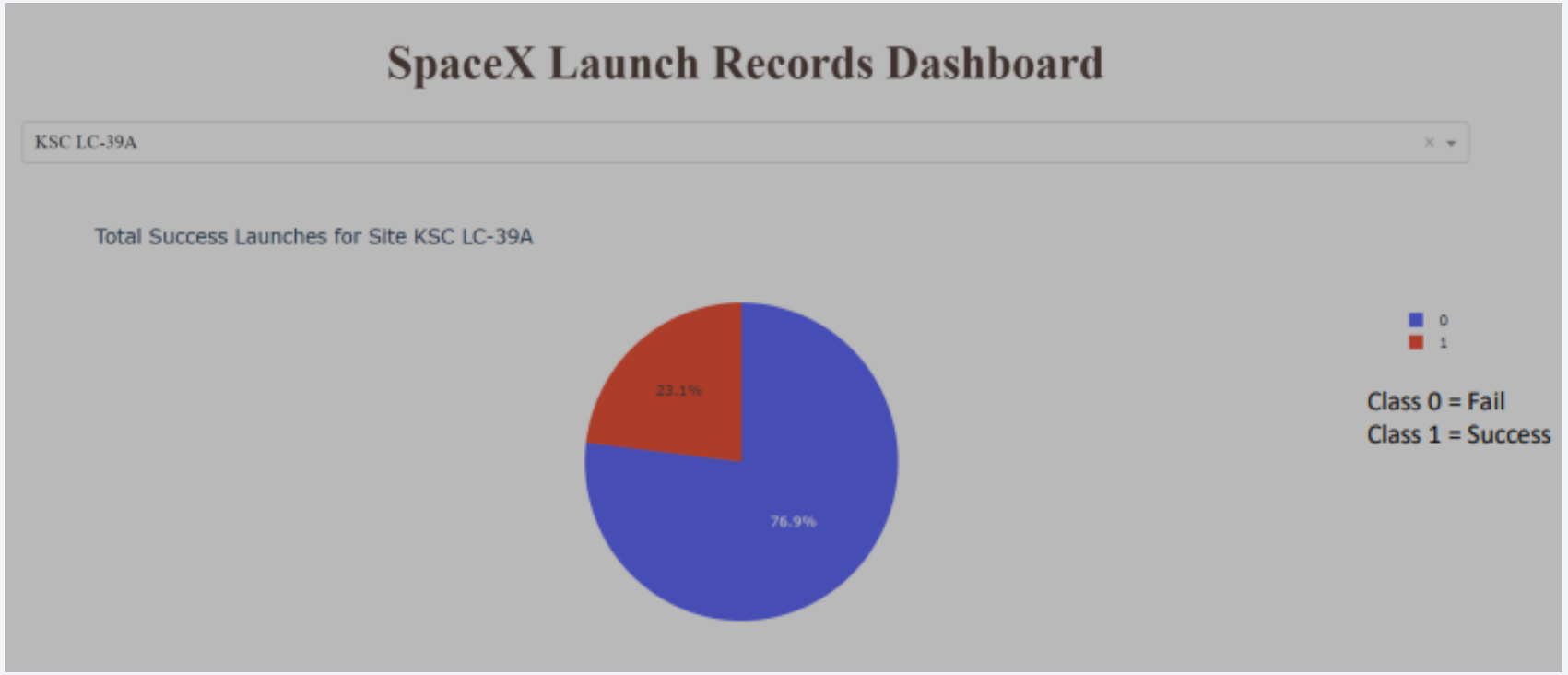
Launch Success by Site

- KSC LC-39A has the most successful launches amongst launch sites (41.2%)



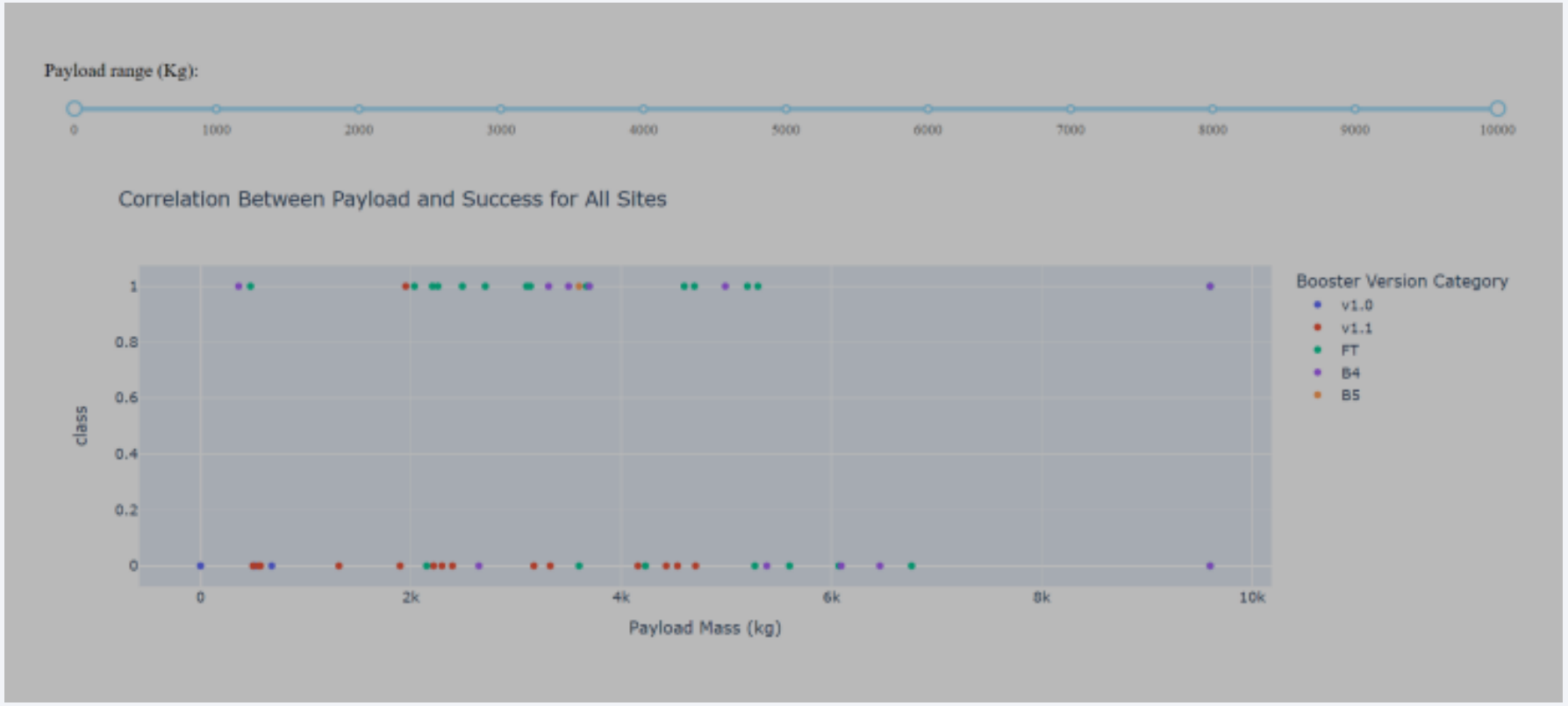
Launch Success

- KSC LC-39A has the highest success rate amongst launch sites (~77%), 10/13 successful launches



Payload Mass and Success

- The highest success rate is found for paylaods between 2000kg and 5000kg.



PREDICTIVE ANALYSIS (CLASSIFICATION)

Classification Accuracy

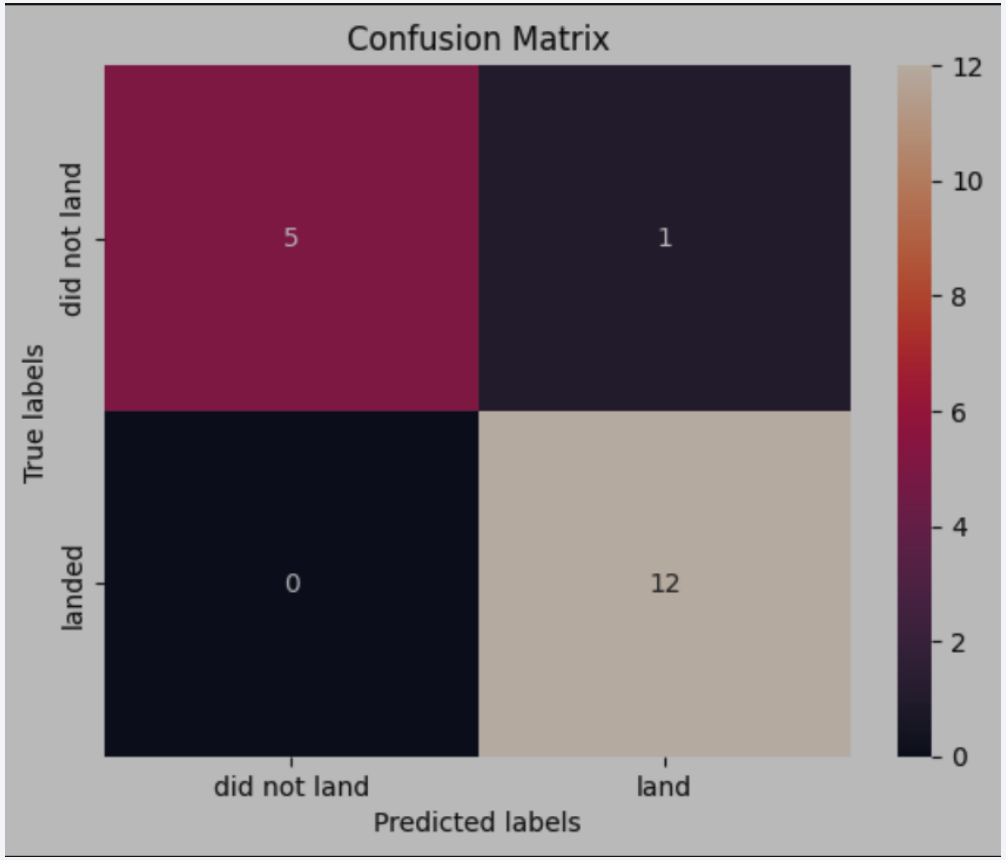
- Due to the small dataset size, most of the models performed similarly. However, the **decision tree model** outperformed the rest of the models.

```
from sklearn.metrics import jaccard_score, f1_score

# Examining the scores from Test sets
jaccard_scores = [
                  jaccard_score(Y_test, logreg_yhat, average='binary'),
                  jaccard_score(Y_test, svm_yhat, average='binary'),
                  jaccard_score(Y_test, tree_yhat, average='binary'),
                  jaccard_score(Y_test, knn_yhat, average='binary'),
                 ]

f1_scores = [
             f1_score(Y_test, logreg_yhat, average='binary'),
             f1_score(Y_test, svm_yhat, average='binary'),
             f1_score(Y_test, tree_yhat, average='binary'),
             f1_score(Y_test, knn_yhat, average='binary'),
            ]

accuracy = [logreg_accuracy, svm_accuracy, tree_accuracy, knn_accuracy]

scores = pd.DataFrame(np.array([jaccard_scores, f1_scores, accuracy]), index=['Jaccard_Score', 'F1_Score', 'Accuracy'] , columns=['LogReg', 'SVM', 'Tree', 'KNN'])
scores
```

	LogReg	SVM	Tree	KNN
Jaccard_Score	0.800000	0.800000	0.923077	0.800000
F1_Score	0.888889	0.888889	0.960000	0.888889
Accuracy	0.833333	0.833333	0.944444	0.833333

Confusion Matrix

- Due to the way the GridSearchCV's parameters were changed for decision tree, the confusion matrix was also affected.
- A confusion matrix summarizes the performance of a classification algorithm
- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1 Score = 2 * (Precision * Recall) / (Precision + Recall)
- Accuracy = (TP + TN) / (TP + TN + FP + FN)



Conclusions

- Most of the models performed similarly, however Decision Tree outperformed them, but the adjustment of parameters to better read the dataset could have caused the increase in accuracy.
- Equator: Most of the launch sites are near the equator for short travel distance to orbit and cost saving.
- Payload Mass: Across all launch sites, the higher the payload mass (kg), the higher the success rate
- Coast: All the launch sites are close to the coast
- Launch Success increases over time
- KSC LC-39A has the highest success rate among launch sites. Has a 100% success rate for launches less than 5,500 kg
- Orbits: GEO, HEO, ES-L1 and SSO have a 100% success rate
- For future studies, a larger dataset could help with building better predictive models that are more generalised.

Thank you!