Winning Space Race with Data Science

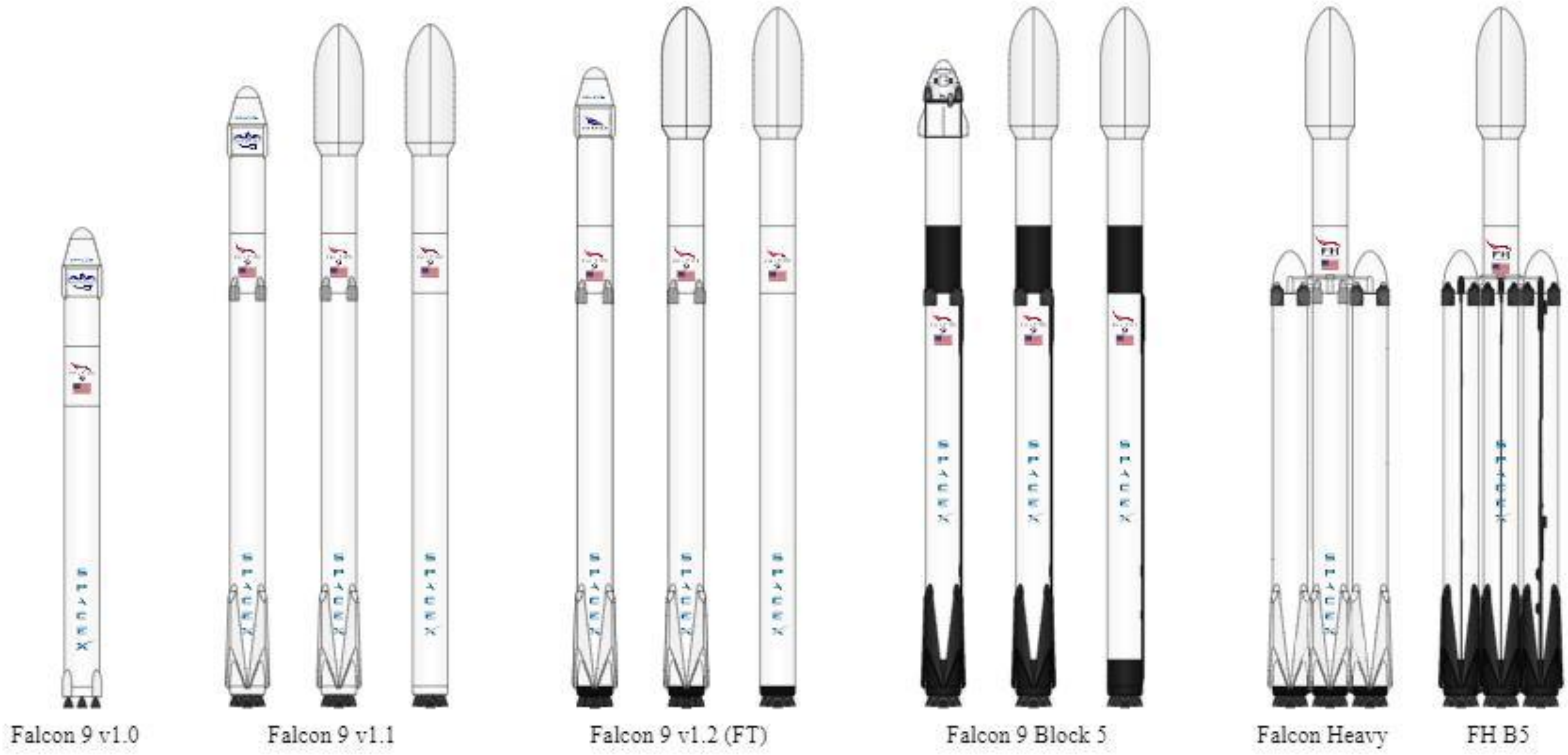
Kadeem P.
03/17/2023

Table of Contents

Executive Summary

This research report describes the development of a highly accurate prediction model for determining the success of Falcon 9's first stage landing. The model was developed using various software packages in Python, including Pandas for data manipulation and BeautifulSoup for web scraping. The dataset was constructed by combining data obtained from the SpaceX Rest API and converted into a Pandas DataFrame. Exploratory data analysis was conducted to classify training labels for successful and unsuccessful landings, revealing that 40% of LandingPad data was null or missing. SQL was used to further analyze the dataset and relationships between variables, while Seaborn algorithm was used to highlight important relationships between variables. The best hyperparameters for SVM, Classification Tree, K Nearest Neighbor, and Logistic Regression models were determined, and all four models demonstrated perfect accuracy for predictiveness on the test data. The report also explores the question of whether knowledge of launch predictability could create an opportunity for alternative companies to bid against SpaceX for a rocket launch, given their current operating costs.

BOOSTER VERSION FALCON 9



Section 1

Methodology

Methodology

In this study, we aimed to develop a high-performing prediction model that could determine the success of Falcon 9 first-stage landings. To achieve this, we followed the following methodology:

Data Collection:

We collected data from SpaceX Rest API (https://api.Spacexdata.com/v4/) that returns the data in JSON format. The data was then converted into a Pandas DataFrame after normalization and manipulation. We also used BeautifulSoup to scrape and convert the parsed data into a Pandas DataFrame for further analysis.

Data Wrangling:

The collected data was processed using Pandas data frames. We performed data wrangling techniques to clean the data and fill in missing values.

Exploratory Data Analysis (EDA):

We performed EDA to determine the training labels that classify the outcome label as 1=Successful landing and 0=Unsuccessful landing. The EDA analysis detected that 40% of LandingPad data was null or missing. We used SQL to perform further analysis, to understand the dataset and relationships between variables. Seaborn algorithm was used to enrich the study highlighting relationships between variables.

Methodology

Interactive Visual Analytics:

We used Folium and Plotly Dash to perform interactive visual analytics of the data, providing an intuitive way to explore the data.

Predictive Analysis:

We built, tuned, and evaluated four different classification models: SVM, Classification Tree, K-Nearest Neighbor, and Logistic Regression. We used the best hyperparameters to train the models and evaluated their performance using test data.

Overall, this study provides an in-depth exploration of developing a high-performing prediction model for determining the success of Falcon 9 first-stage landings. The methodology used ensures that the data is processed and analyzed accurately, providing reliable insights into the data. The results of this study could be useful in making informed decisions related to rocket launches and could create an opportunity for alternative companies to bid against SpaceX for a rocket launch.

Data Collection

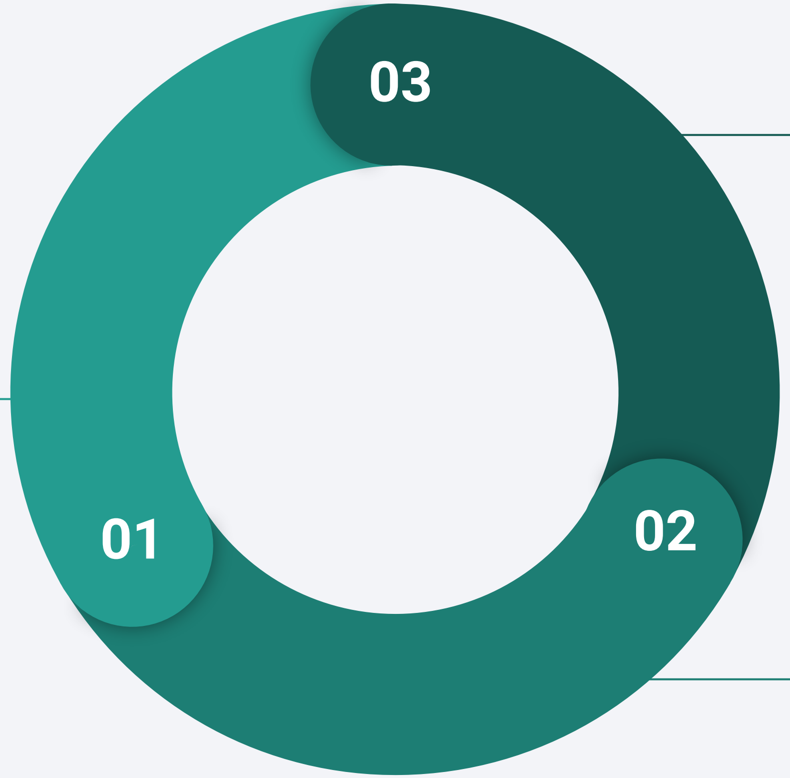
To collect data for the development of a prediction model to determine the success of Falcon 9 first stage landings, the SpaceX REST API was utilized. The SpaceX API returns data in JSON format, which was converted into a Pandas DataFrame using Python. The data was then processed and normalized using Pandas data frames and Beautiful Soup for web scraping.

Once the data was collected, it was combined into a dictionary to form the dataset. Exploratory data analysis was performed to determine the training labels that classify the outcome label 1=Successful landing and 0=Unsuccessful landing. The analysis detected that 40% of LandingPad data was null or missing.

To further analyze the dataset and understand the relationships between variables, SQL was used. Seaborn algorithm was used to enrich the study highlighting relationships between variables.

Finally, classification models such as SVM, Classification Tree, K Nearest Neighbor, and Logistic Regression were built, tuned, and evaluated for predictive accuracy. The process involved determining the best hyperparameters for each model using test data. The best model was determined by evaluating the accuracy score and cross-validation (CV) score.

Data Collection – SpaceX API Flow



Client Request to API Server

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

The Server Retrieves the Data from GET and Passes Data to Response in JSON for Normalization

```
response = requests.get(spacex_url)
response.status_code
```

API Called to Retrieve Data

```
df = pd.json_normalize(response.json())
df.head(5)
```

See NoteBook and Run Cells:
https://github.com/Scoubershare/spaceX/blob/b99c74a5b33ea66feb9fa3a466a9d9f82b6de622/jupyter-labs-spacex-data-collection-api.ipynb#L2

Data Collection - Scraping Process

Required Libraries → Request to Wiki Page → BeautifulSoup ObJ - HTML Response → Data Extraction - HTML Table → Convert to DataFrame

```
import sys

import requests
from bs4 import import
BeautifulSoup
import re
import unicodedata
import pandas as pd
```

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

response =
requests.get(static_url)
```

```
soup =
BeautifulSoup(response.text, 'html.parser')

print(soup.title)
```

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

first_launch_table =
html_tables[2]
print(first_launch_table)

column_names = []
for th in
first_launch_table.find_all('th'):
    name =
extract_column_from_header(th)
    if name is not None
and len(name) > 0:
column_names.append(name)
```

```
launch_dict=
dict.fromkeys(column_names)
launch_dict['Flight No.'] =
[]
launch_dict['Date']=[]
launch_dict['Time']=[]
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']=[]
```

See NoteBook and Run Cells:
https://github.com/Scoubershare/spaceX/blob/b99c74a5b33ea66feb9fa3a466a9d9f82b6de622/jupyter-labs-webscraping (1).ipynb#L1-L19

Data Wrangling - Variables for Outcome

SpaceX missions have three possible landing sites: a specific region of the ocean, a ground pad, or a drone ship. The outcomes for a landing can be categorized as success, failure, or no attempt (represented by `None`).

There are four possible combinations of landing site and outcome: successful or failed landings on a specific region of the ocean (`True Ocean` or `False Ocean`), successful or failed landings on a ground pad (`True RTLS` or `False RTLS`), and successful or failed landings on a drone ship (`True ASDS` or `False ASDS`).

In addition, there are two special cases: `None ASDS`, which indicates that a mission attempted to land on a drone ship but there was no data available on the outcome, presumably due to a lack of telemetry data, and `None None`, which indicates that a mission did not attempt a landing, either because it was not designed to do so or because the landing was aborted for some reason.

Data Wrangling - EDA/ Determine Training Label

Successful Landing Rate: 66%

Cal. Number & Occurrence of mission outcome per orbit type

	Event Occurrences
True ASDS	41
None None	19
True RTLS	14
False ASDS	6
True Ocean	5

Classification variable representing Outcome for each launch 1= Success & 0 = Did not land Successful

	Class
41	1
42	1
43	1
44	1
45	0

Cal. Number of launches per site

	Launches per site
CCAFS SLC 40	55
KSC LC 39A	22
VAFB SLC 4E	13

Cal. Number and Occurrence of each orbit

	Launches per Orbit
GTO	27
ISS	21
VLEO	14
PO	9
LEO	7
SSO	5
MEO	3
ES-L1	1
HEO	1
SO	1
GEO	1

See NoteBook and Run Cells:
https://github.com/Scoubershare/spaceX/blob/b99c74a5b33ea66feb9fa3a466a9d9f82b6de622/labs-jupyter-spacex-Data wrangling.ipynb#L2

EDA with Data Visualization

The data was visualized using Matplotlib and Seaborn. Flight number was plotted against payload mass, with the target label outcome overlaid to show the success or failure of landings as the number of flights and payload size increased. Seaborn was used to plot flight number and launch site against the target label, revealing that launches from VAFB SLC 4E were more successful as the number of flights increased. The relationship between payload and launch site was also plotted in Seaborn, and payloads greater than 10,000 kg were found to be more likely to have a successful outcome. A bar graph was used to analyze the success rate of different orbit types, with 4 launch sites having the highest success rate for stage two launches in GEO, HEO, SSO, and ES-L1 orbits. The relationship between orbit type and flight number was also examined, with LEO flights becoming more successful with each attempt. Finally, the relationship between payload and orbit type was explored, and it was found that payloads under 4,000 kg launched into SSO had a 100% success rate for landings.

See NoteBook and Run Cells:
https://github.com/Scoubershare/spaceX/blob/b99c74a5b33ea66feb9fa3a466a9d9f82b6de622/jupyter-labs-eda-dataviz (1).ipynb

EDA with SQL - SQL Common Queries

See NoteBook to Run Queries: https://github.com/Scoubershare/spaceX/blob/b99c74a5b33ea66feb9fa3a466a9d9f82b6de622/jupyter-labs-eda-sql-coursera_sqllite.ipynb#L2-L3

```
launch_sites = df['Launch_Site'].unique()
```

```
df3 = df3[['Booster_Version', 'PAYLOAD_MASS__KG_', 'Landing_Outcome']]
```

```
df1 = df.loc[df['Booster_Version'] == 'F9 v1.1']

df1 = df1['PAYLOAD_MASS__KG_'].mean()
```

```
filtered_df = df.loc[df['Launch_Site'].str.startswith('CCA')]
```

```
df4 = df['Mission_Outcome'].value_counts()
```

```
df2 = df.loc[df["Landing _Outcome"] == 'Success (ground pad)']
df2 = df2['Date']
```

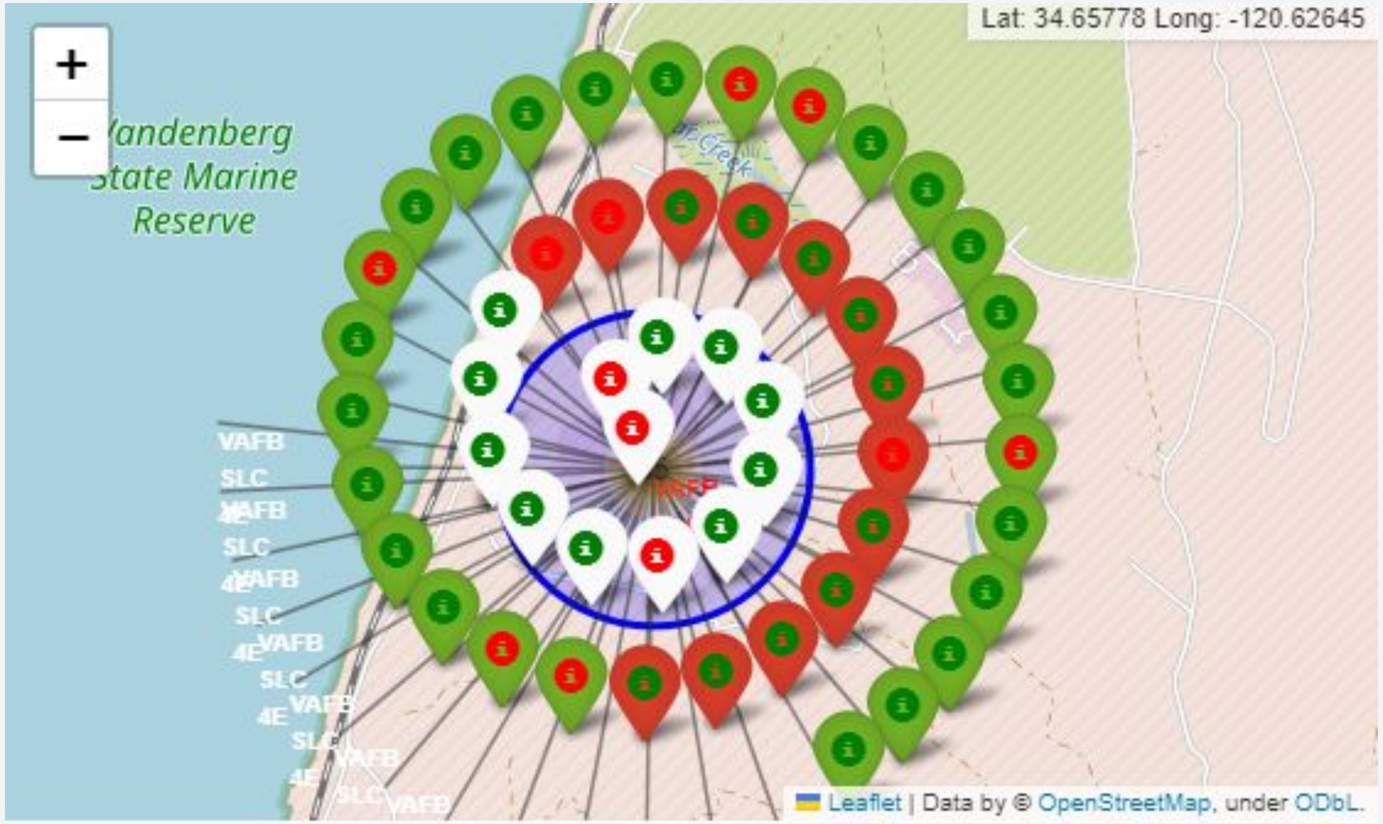
```
filtered_df = df.loc[df['Customer'] == 'NASA (CRS)']
filtered_df= filtered_df['PAYLOAD_MASS__KG_'].sum()
```

```
df5 = df.loc[df['PAYLOAD_MASS__KG_'] == df['PAYLOAD_MASS__KG_'].max()]
```

```
df3 = df.loc[(df["Landing _Outcome"] == 'Success (drone ship)') & (df["PAYLOAD_MASS__KG_"] > 4000) & (df["PAYLOAD_MASS__KG_"] < 6000)]
```

Build an Interactive Map with Folium

- The Folium package was used to map Maker Clusters, which were used to display launch sites with successful and failed outcomes, as well as their proximity.



See NoteBook and Run Cells:
https://github.com/Scoubershare/spaceX/blob/b99c74a5b33ea66feb9fa3a466a9d9f82b6de622/lab_jupyter_launch_site_location.ipynb#L1

Build a Dashboard with Plotly Dash

In our project, we utilized the Plotly package to add interactivity to our visualizations. Specifically, we created a dropdown menu in Plotly that allowed users to select a launch site, and the corresponding pie chart would display the success rate for that site. We also created a slider range for the payload mass, overlaid with the target value, to investigate how outcomes varied as the payload mass changed. This allowed us to better understand the relationship between payload mass and mission success.

Url: http://Localhost:8085

See Python File Must run Py file then search for url above in browser:
https://github.com/Scoubershare/spaceX/blob/b99c74a5b33ea66feb9fa3a466a9d9f82b6de622/spacex_dash_app.py

Predictive Analysis (Classification)

The study involved testing various classification models including KNN, Classification Tree, Logistic Regression, and SVM to determine the best model for predictive analysis. The models were tested with different hyperparameters, and for the Classification Tree, the cross-validation was improved to 11. Confusion Matrix were used to evaluate predictive performance.

```
GridSearchCV(cv=11, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

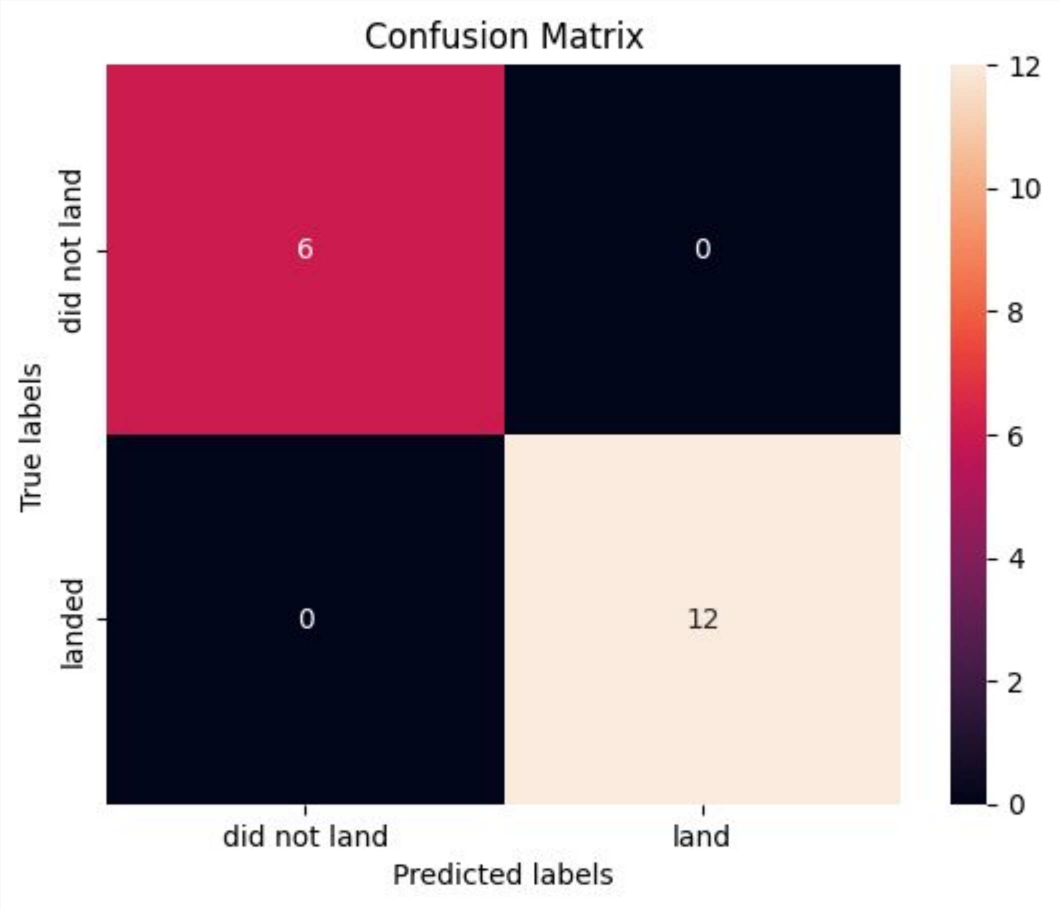
```
GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'p': [1, 2]})
```

```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e
       1.00000000e+03]),
                         'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227
       1.00000000e+03]),
                         'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
```

See NoteBook and Run Cells:
https://github.com/Scoubershare/spaceX/blob/b99c74a5b33ea66feb9fa3a466a9d9f82b6de622/SpaceX_Machine Learning Prediction_Part_5.ipynb#L4

Results

All four models demonstrated an accuracy rate of 1, indicating that they were accurate enough on the predictive test set. The test set used 18 samples

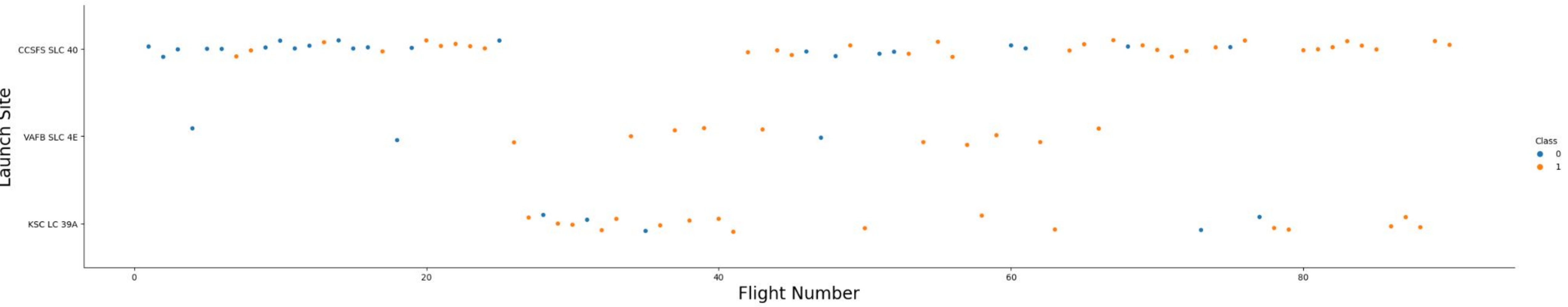


Section 2

Insights drawn from EDA

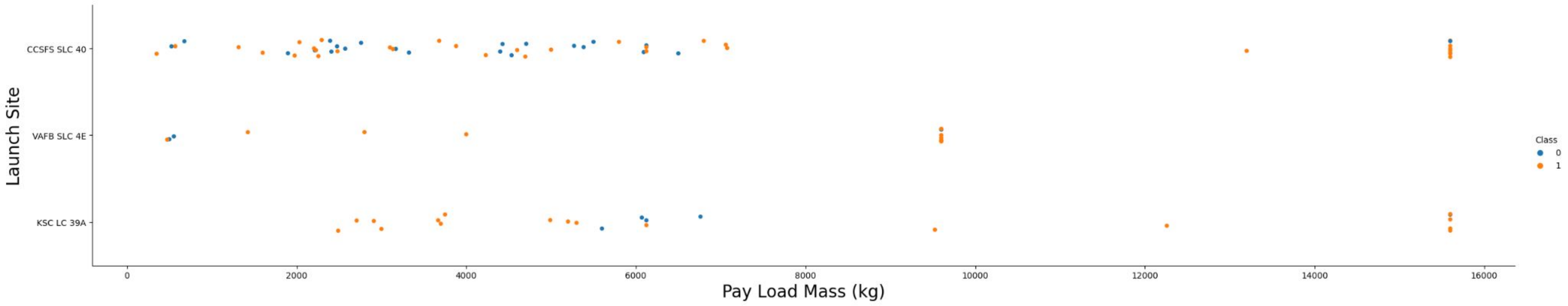
Flight Number vs. Launch Site

- CCSFS SLC 40 launch site showed a weak relationship with outcomes as the number of flights increase.



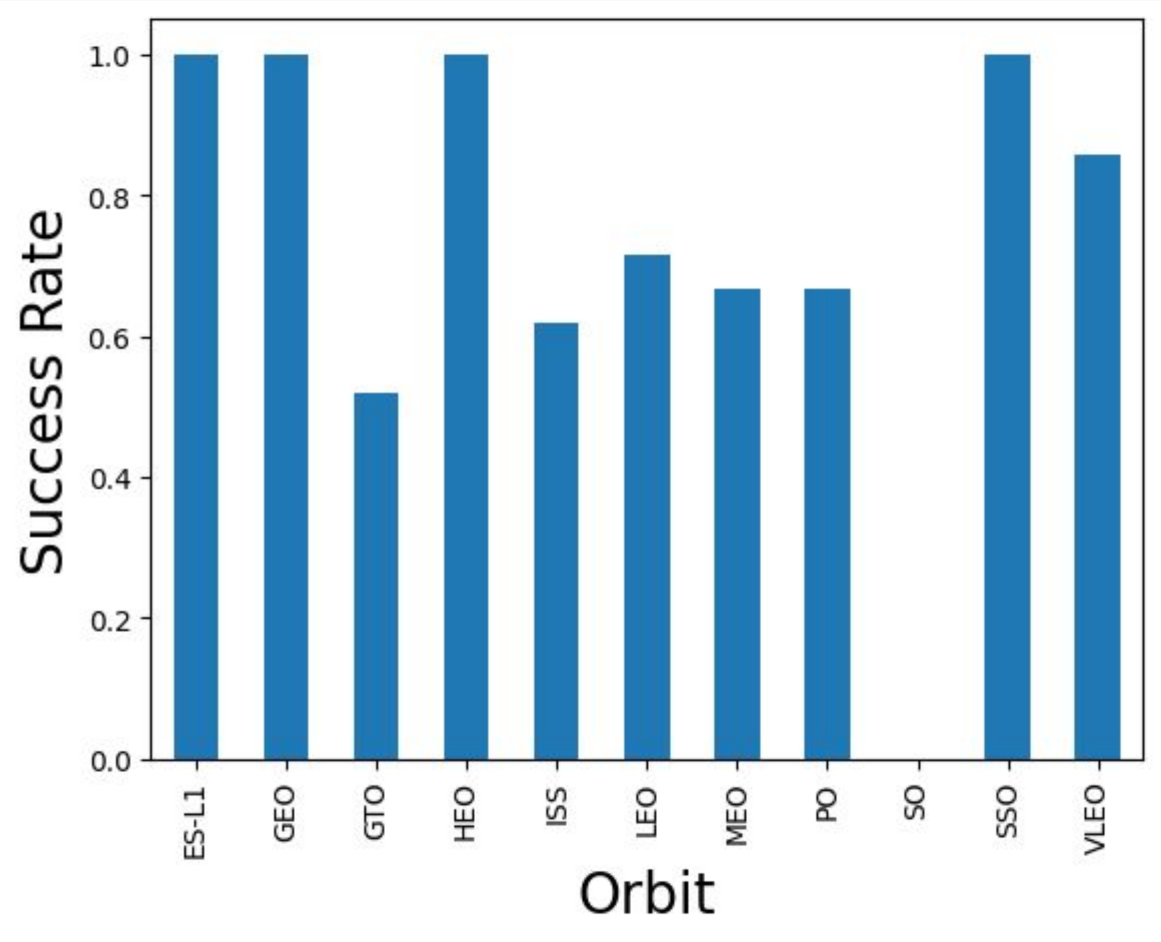
Payload vs. Launch Site

- Flights launched from KSC LC 39A less than 5,000_KG had successful outcomes.



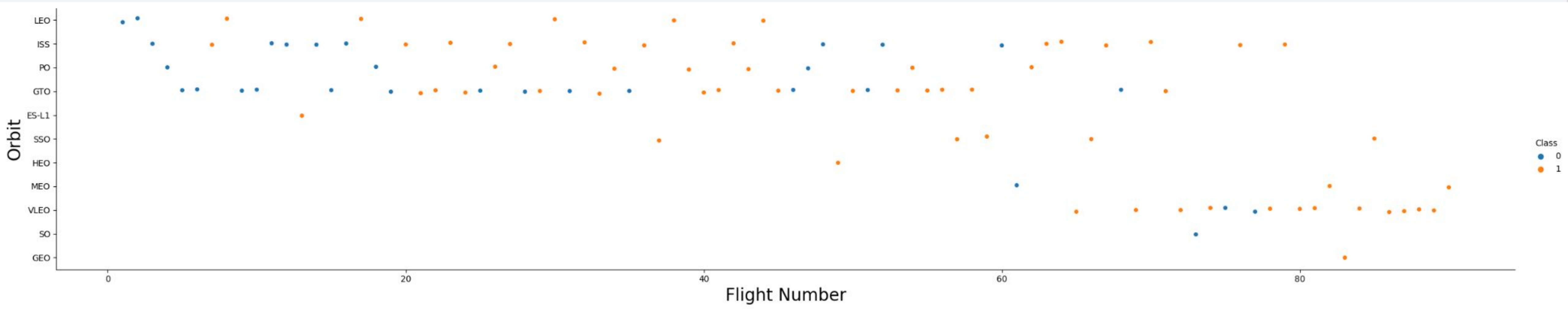
Success Rate vs. Orbit Type

- Flights launched into these orbit types were the most successful:
    - ES-L1
    - GEO
    - HEO
    - SSO



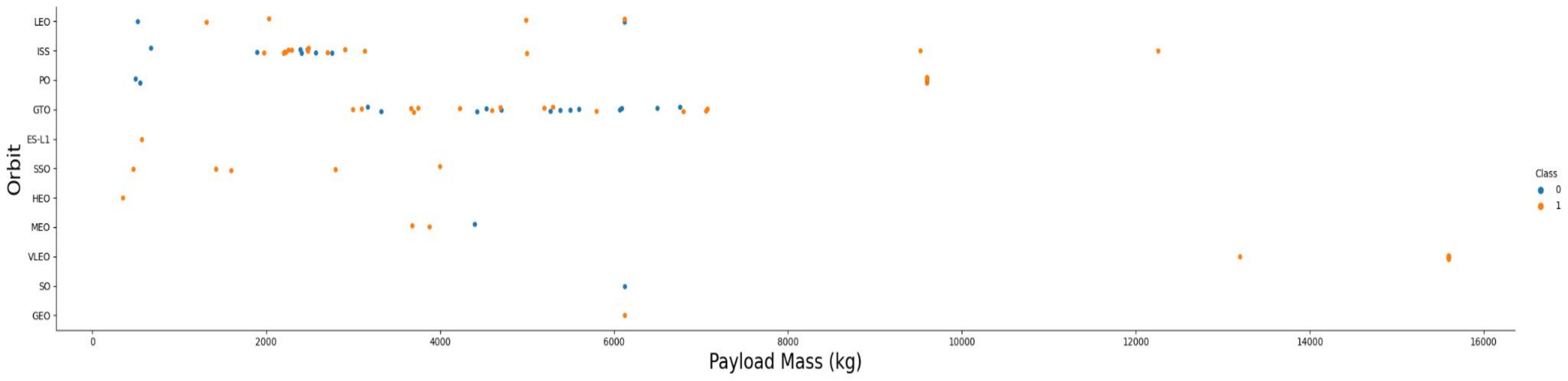
Flight Number vs. Orbit Type

- As the number of flights into LEO orbit increase so did the its successful outcomes. Successful flights into GTO orbit showed not relationship as flight numbers increased.



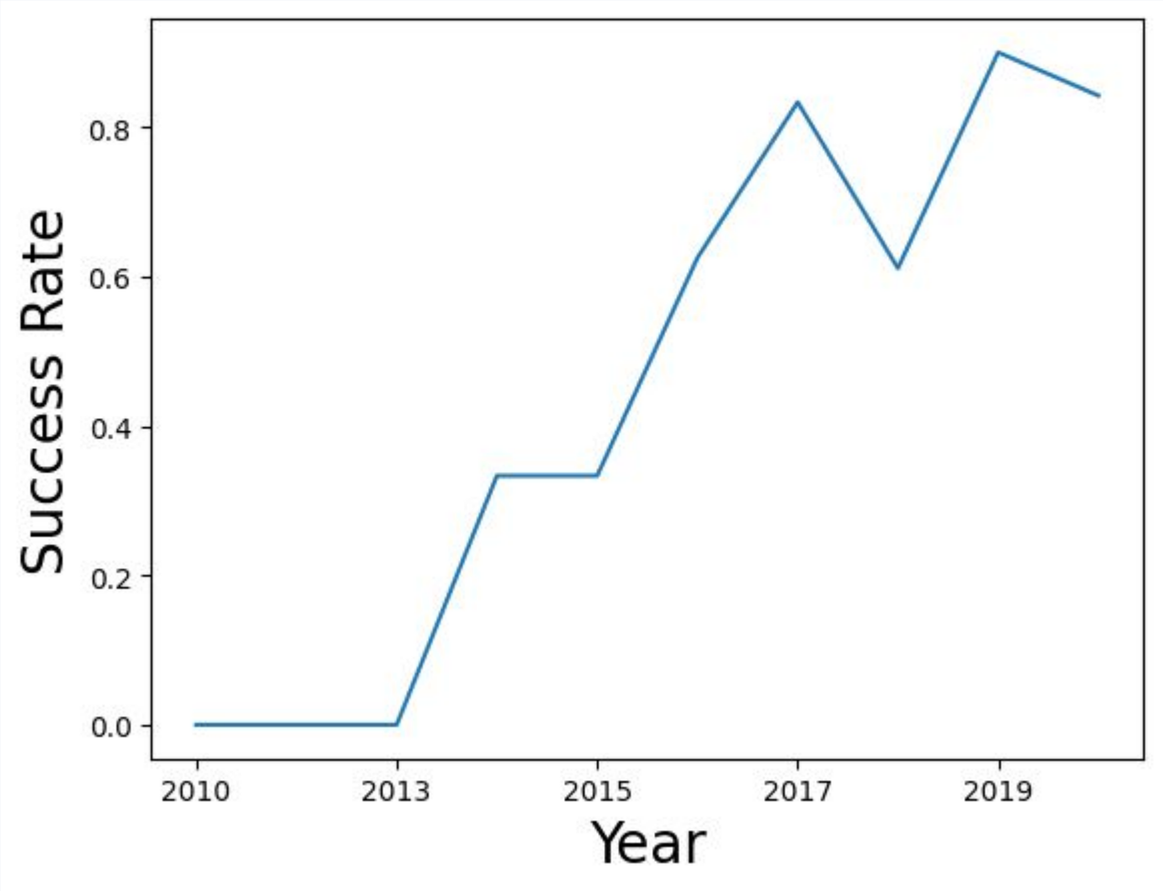
Payload vs. Orbit Type

- The most successful outcomes were less than 4000_KG into SSO orbit.



Launch Success Yearly Trend

- According to the trend line launch success continued to increase between 2013 to 2019.



All Launch Site Names

```python
launch_sites = df['Launch_Site'].unique()

# Print the unique launch sites
print('Unique launch sites:')
for site in launch_sites:
    print(site)
```

```
Unique launch sites:
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
```

Launch Site Names Begin with 'CCA'

```
filtered_df =
df.loc[df['Launch_Site'].str.st
artswith('CCA')]
filtered_df.head(5)
```

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
0	04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

```
filtered_df = df.loc[df['Customer'] == 'NASA (CRS)']
filtered_df= filtered_df['PAYLOAD_MASS__KG_'].sum()
print(filtered_df)
```

```
Total Payload Mass_KG: 45,596_KG
```

Average Payload Mass by F9 v1.1

```
df1 = df.loc[df['Booster_Version'] == 'F9 v1.1']

#filter the data to only include average payload mass
df1 = df1['PAYLOAD_MASS__KG_'].mean()

print(df1)

Average Payload Mass_KG: 2,928_KG
```

First Successful Ground Landing Date

```
df2 =
df.loc[df["Landing
_Outcome"] == 'Success
(ground pad)']


#filter the data to
only include the date
df2 = df2['Date']

#display the first date
#df2.head(1)
```

```
The first successful landing outcome in ground pad was acheived on:
```

	Date
19	22-12-2015

Successful Drone Ship Landing with Payload between 4000 and 6000

```
df3 = df.loc[(df["Landing _Outcome"] == 'Success (drone ship)') &
(df["PAYLOAD_MASS__KG_"] > 4000) & (df["PAYLOAD_MASS__KG_"] < 6000)]

#filter the data to only include the booster version and the payload mass
df3 = df3[['Booster_Version', 'PAYLOAD_MASS__KG_', 'Landing _Outcome']]

#convert the data to a dataframe
df3 = pd.DataFrame(df3)
df3
```

	Booster_Version	PAYLOAD_MASS__KG_	Landing _Outcome
23	F9 FT B1022	4696	Success (drone ship)
27	F9 FT B1026	4600	Success (drone ship)
31	F9 FT B1021.2	5300	Success (drone ship)
42	F9 FT B1031.2	5200	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

```
df4 = df['Mission_Outcome'].value_counts()
df4 = pd.DataFrame(df4)
df4
```

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

Boosters Carried Maximum Payload

```
#list the names of the booster versions which
carried the maximum payload mass
df5 = df.loc[df['PAYLOAD_MASS__KG_'] ==
df['PAYLOAD_MASS__KG_'].max()]

#filter the data to only include the booster
version and the payload mass
df5 = df5[['Booster_Version',
'PAYLOAD_MASS__KG_']]
df5= pd.DataFrame(df5)
df5
```

	Booster_Version	PAYLOAD_MASS__KG_
74	F9 B5 B1048.4	15600
77	F9 B5 B1049.4	15600
79	F9 B5 B1051.3	15600
80	F9 B5 B1056.4	15600
82	F9 B5 B1048.5	15600
83	F9 B5 B1051.4	15600
85	F9 B5 B1049.5	15600
92	F9 B5 B1060.2	15600
93	F9 B5 B1058.3	15600
94	F9 B5 B1051.6	15600
95	F9 B5 B1060.3	15600
99	F9 B5 B1049.7	15600

2015 Launch Records

```
#filter the date to only include the year
df['Year'] = pd.DatetimeIndex(df['Date']).year

#filter the date to show month name and year
df['Month'] = pd.DatetimeIndex(df['Date']).month_name()

# 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.
df6 = df.loc[(df['Year'] == 2015) & (df['Landing _Outcome'] == 'Failure (drone ship)')]
df6 = df6[['Month','Year', 'Landing _Outcome', 'Booster_Version', 'Launch_Site']]
df6
```

	Month	Year	Landing _Outcome	Booster_Version	Launch_Site
13	October	2015	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
16	April	2015	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

```
df10 = df.loc[(df['Date'] > '04-06-2010') &
(df['Date'] < '20-03-2017') & (df['Landing
_Outcome'].str.startswith('Success'))]
df10
```

Thank you!