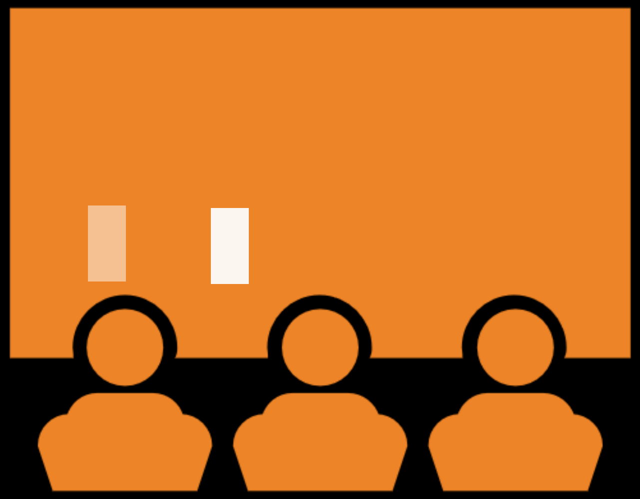
PREDICTING FALCON 9 FIRST STAGE LANDING SUCCESS: A COST-EFFECTIVE APPROACH

Austin Liwanag

April 23, 2024

OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Conclusion
- Appendix



EXECUTIVE SUMMARY

Methodologies:

- Data Collection
- Data Wrangling
- Exploratory Data Analysis (EDA)
- Interactive Data Visualization
- Machine Learning Prediction

Results:

- Exploratory Data Analysis (EDA) results
- Interactive dashboard
- Machine Learning Prediction analysis

INTRODUCTION

Overview

- SpaceX, founded by Elon Musk in 2002, has revolutionized space transportation with its ambitious goal of lessening costs and empowering the colonization of Mars. One of the key factors behind SpaceX's accomplishment is its innovative reuse of the first stage of the Falcon 9 rocket. By reusing this critical component, SpaceX significantly lowers launch costs compared to other providers.

Objective

- In our capstone project, we delve into predicting whether the Falcon 9 first stage will successfully land after a launch. By leveraging machine learning techniques, we aim to develop a predictive model that can regulate the likelihood of a fruitful landing. Join us on this breathtaking journey as we explore data, build models, and contribute to the future of space exploration.

METHODOLOGY

- Data Collection
  - Gathering relevant data related to Falcon 9 launches.
- Data Wrangling
  - Cleaning and formatting the collected data.
- Exploratory Data Analysis (EDA)
  - Exploring patterns and insights.
- Interactive Data Visualization
  - Creating visualizations to understand the data better.
- Machine Learning Prediction
  - Leveraging classification algorithms to predict first stage landing outcomes.

DATA COLLECTION – SPACEX API

- We collect and ensure the data is in the right format from an API that SpaceX offers.
- Then, we request and parse SpaceX launch data using API calls.
- Source:
  - https://github.com/Yosti-Light/SpaceX-capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

DATA COLLECTION – WEB SCRAPING

- We perform web scraping by extracting a Falcon 9 launch records table from web sources like a list of Falcon 9 and Falcon heavy launches page.
- Using BeautifulSoup, we parse the table and convert it to a Pandas data frame for collecting information on launch outcomes, dates, and conditions.
- Source:
  - https://github.com/Yosti-Light/SpaceX-capstone/blob/main/jupyter-labs-webscraping.ipynb

DATA WRANGLING

- Some Exploratory Data Analysis (EDA) was used on the dataset to look for some patterns in the data and to find out what would be the label for training supervised models.
- We determine Training Labels with 1 meaning a successful booster landing and 0 meaning it is unsuccessful.
- Source:

  https://github.com/Yosti-Light/SpaceX-capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EXPLORATORY DATA ANALYSIS (EDA) WITH SQL

- If we want to know if the first stage will land, we can determine the cost of a launch.
- If a company would like to bid against SpaceX for launch, we can use this information.
- The SpaceX dataset consists of a record for each payload toted during a SpaceX mission.
- So we load the dataset into a database and execute the required SQL queries.
- Source:

  https://github.com/Yosti-Light/SpaceX-capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

EXPLORATORY DATA ANALYSIS (EDA) WITH VISUALIZATION

- While performing Exploratory Data Analysis (EDA), we also prepare Data Feature Engineering using Pandas and Matplotlib.
- To analyze data, we use scatterplots and bar plots to envision the relationship between Flight Number, Payload Mass, and Orbit.
- Source:

  https://github.com/Yosti-Light/SpaceX-capstone/blob/main/edadataviz.ipynb

INTERACTIVE VISUAL ANALYTICS WITH FOLIUM

- To make a Folium map for SpaceX launch dataset visualization, we use markers, marker clusters, and lines.
  - Markers indicate all launch sites on a map.
  - Marker clusters indicate the success/failed launches for each site.
  - Lines calculate the distance between two coordinates.
- Source:
  - https://github.com/Yosti-Light/SpaceX-capstone/blob/main/lab_jupyter_launch_site_location.ipynb

BUILD AN INTERACTIVE DASHBOARD WITH PLOTLY DASH

- On the Plotly dashboard, we make:
  - A pie chart for visualizing the total launch success counts per site. This makes it so that we know which sites are most successful. If we want to see the success or failure for a site, we can filter the chart using dcc.Dropdown().
  - A scatter plot for observing the correlation between mission outcomes and payload mass (kg). We can filter the chart using dcc.**RangeSlider**().
- Source:
  - https://github.com/Yosti-Light/SpaceX-capstone/blob/main/spacex_dash_app.py

MACHINE LEARNING PREDICTION

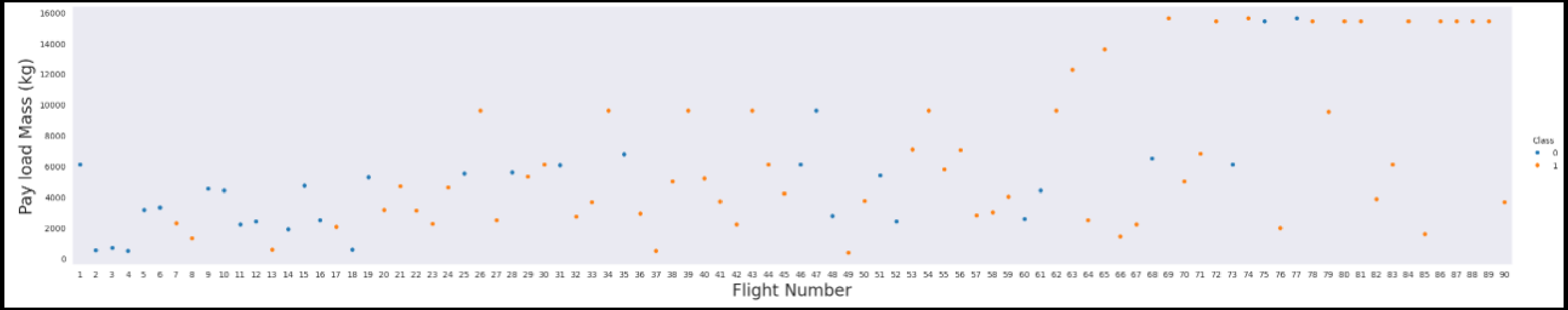
- Some Exploratory Data Analysis (EDA) and Training Labels were used to standardize the data and split it into two types of data: training and test.
- We find the best hyperparameter with the help of an SVM, a classification tree, and logistic regression.
- Source:
  - https://github.com/Yosti-Light/SpaceX-capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

RESULTS

- Exploratory Data Analysis (EDA) results
- Interactive dashboard
- Machine Learning Prediction analysis

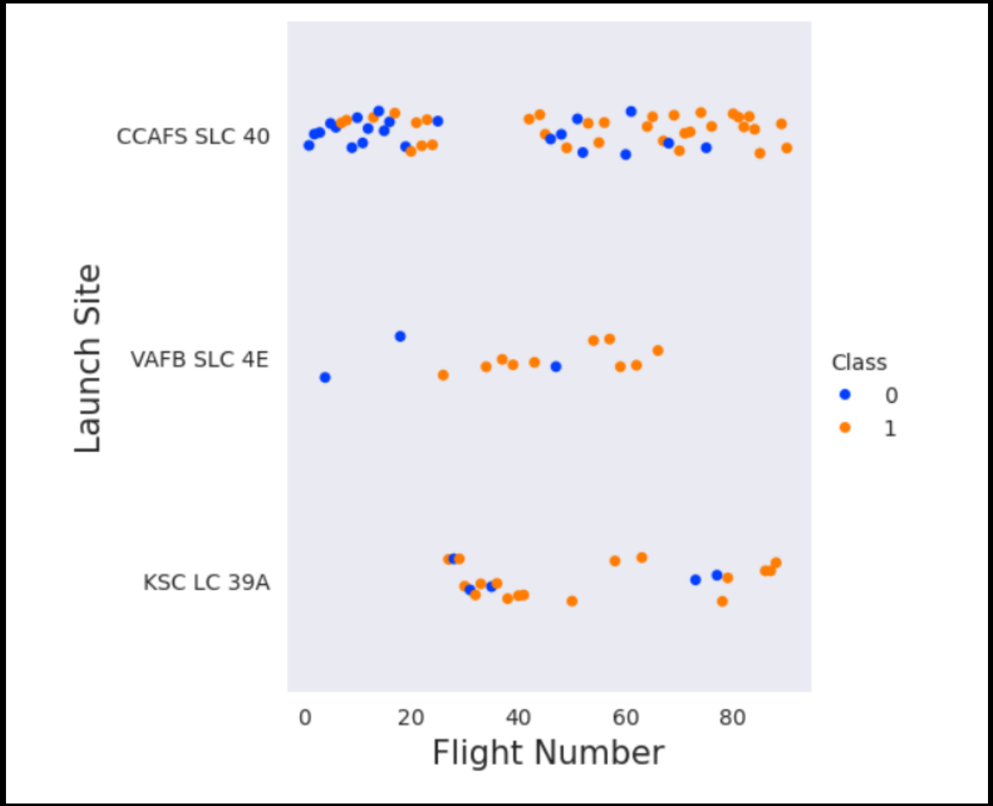
EXPLORATORY DATA ANALYSIS (EDA) - FLIGHT NUMBER VS. PAYLOAD MASS

- The higher the flight number, the more likely the first stage will succeed in landing.
- As the payload mass increases, the first stage is less likely to return.
- So SpaceX has honed is landing competence over time. Later flights prioritized superior outcomes relating to recovery of the first stage.



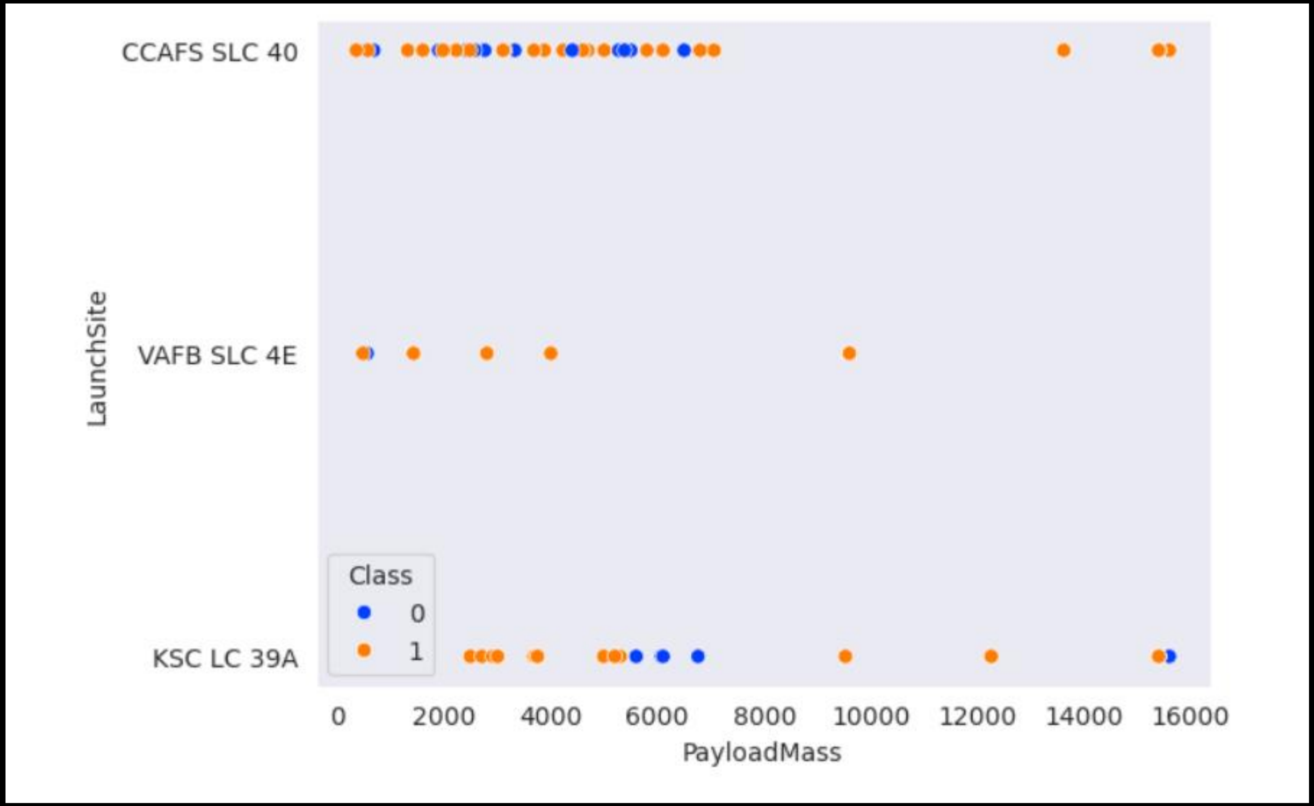
EXPLORATORY DATA ANALYSIS (EDA) - FLIGHT NUMBER VS. LAUNCH SITE

- The clustering on this scatter plot could mean that launch sites are accompanying specific flight numbers.
- Launches from a site like CCAFS SLC-40 might constantly use a certain range of flight numbers while another one like VAFB SLC-4E uses a different range.
- This could be because of operational practices or mission types.

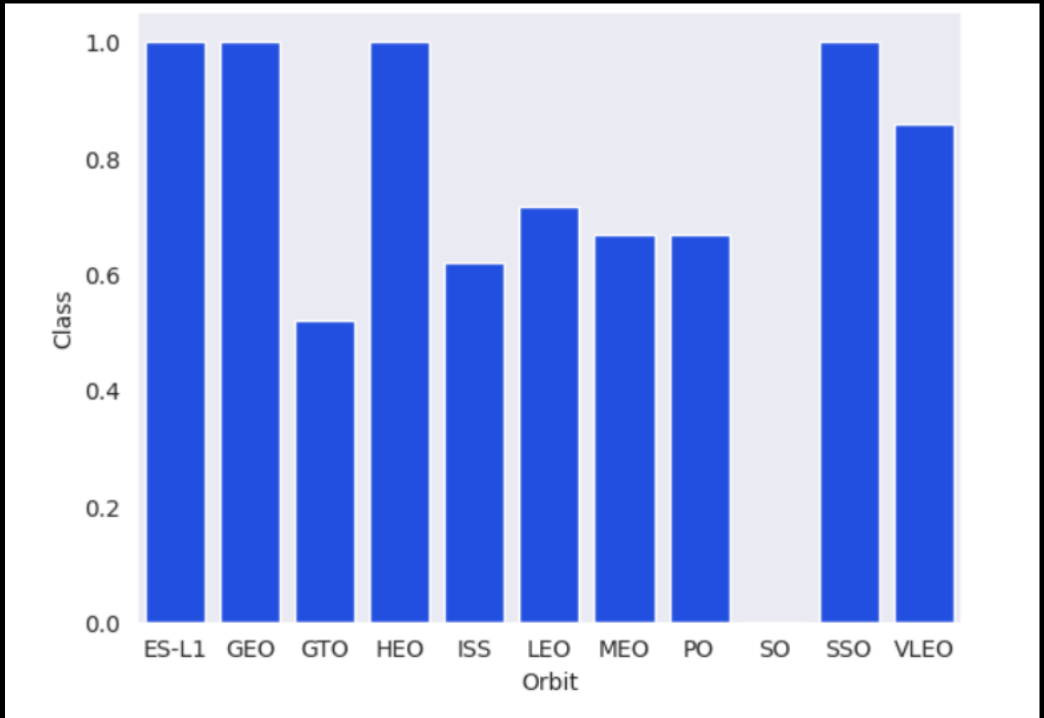


EXPLORATORY DATA ANALYSIS (EDA) - PAYLOAD MASS VS. LAUNCH SITE

- The payload mass has a crucial role: heavy payloads are clearly linked to a lower likelihood of prosperous first-stage return.
- If we observe the Payload Mass vs. Launch Site scatter point plot, we can see that VAFB-SLC does not have any rockets launched for mass greater than 10000.

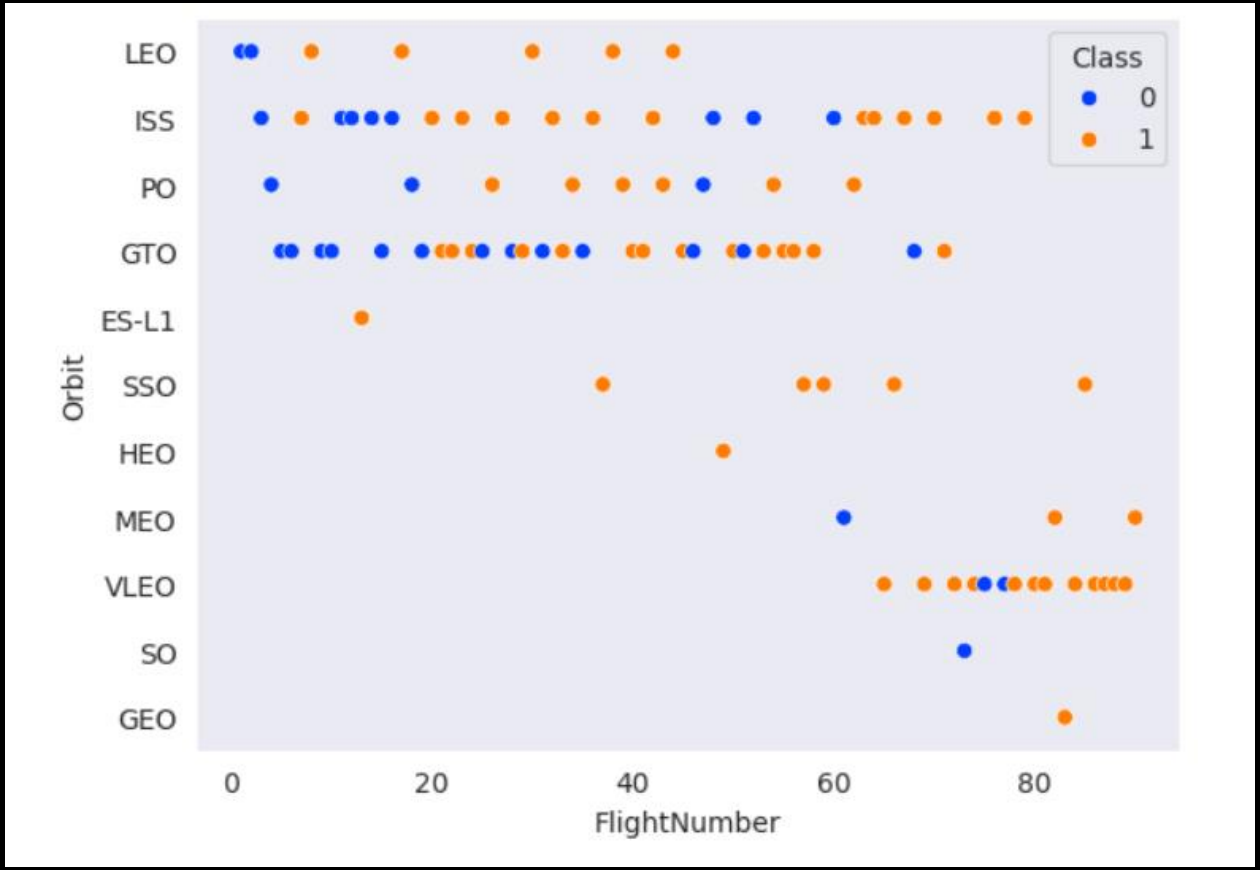


EXPLORATORY DATA ANALYSIS (EDA) - SUCCESS RATE VS. ORBIT TYPE



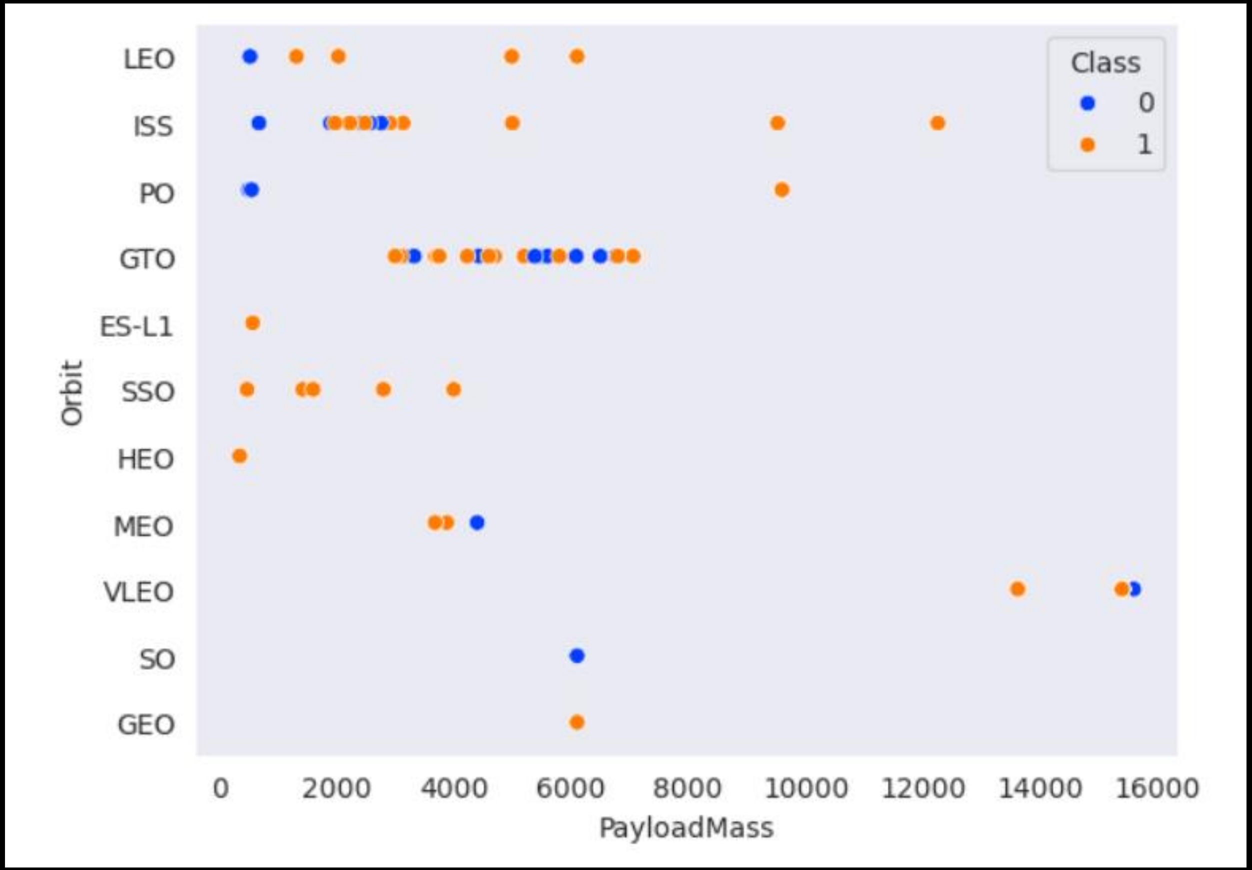
- Orbits like ES-L1, GEO, HEO, and SSO have a high success rate of 1.0. This could mean that these orbits are correlated with accomplished first-stage landings.
- However, the SO orbit has a low rate of 0, meaning that suborbital launches are difficult in respect of first-stage recovery.

EXPLORATORY DATA ANALYSIS (EDA) - FLIGHT NUMBER VS. ORBIT TYPE



- We can see that in an orbit like LEO, the success appears related to the number of flights, although we see that there is no relationship between flight number when in GTO.

EXPLORATORY DATA ANALYSIS (EDA) - PAYLOAD MASS VS. ORBIT TYPE



- With hefty payload, the landing rates are more optimistic for LEO and ISS despite that GTO cannot be distinguished because both positive and negative landing rates are there.

EXPLORATORY DATA ANALYSIS (EDA) - SQL

- The query %sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL displays the names of the unique launch sites.

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

- The query %sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5 displays five records with launch sites that begin with the string 'CCA'.

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

EXPLORATORY DATA ANALYSIS (EDA) - SQL

- The query %sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)' displays the total payload mass carried by boosters launched by NASA (CRS).

TOTAL_PAYLOAD_MASS
45596

- The query %sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVERAGE_PAYLOAD_MASS FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1' displays the average payload mass brought by booster version F9 v.11.

AVERAGE_PAYLOAD_MASS
2928.4

- The query %sql SELECT min(date) FROM SPACEXTBL where "Landing_Outcome"='Success (ground pad)' lists the date when the first successful landing outcome in ground pad was attained.

min(date)
2015-12-22

EXPLORATORY DATA ANALYSIS (EDA) - SQL

- The query %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE ("Landing_Outcome" = 'Success (drone ship)') AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000) lists the names of the boosters that succeeded in drone ship and have payload mass greater than 4000 but less than 6000.

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

- The query %sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL GROUP BY MISSION_OUTCOME lists the total number of successful and failure mission outcomes.

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

EXPLORATORY DATA ANALYSIS (EDA) - SQL

- The query %sql SELECT DISTINCT(BOOSTER_VERSION) FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL) lists the names of the booster versions which have carried the maximum payload mass.

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

EXPLORATORY DATA ANALYSIS (EDA) - SQL

- The query %sql SELECT "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Failure (drone ship)' AND substr(Date,1,4) = '2015' lists the records that will show the month names(However, we cannot use monthnames on SQLite. Instead, we use substr(Date,6,2) for months and substr(Date,0,5)='2015' for year), failure landing outcomes in drone ship, booster versions, and launch site for the months in 2015.

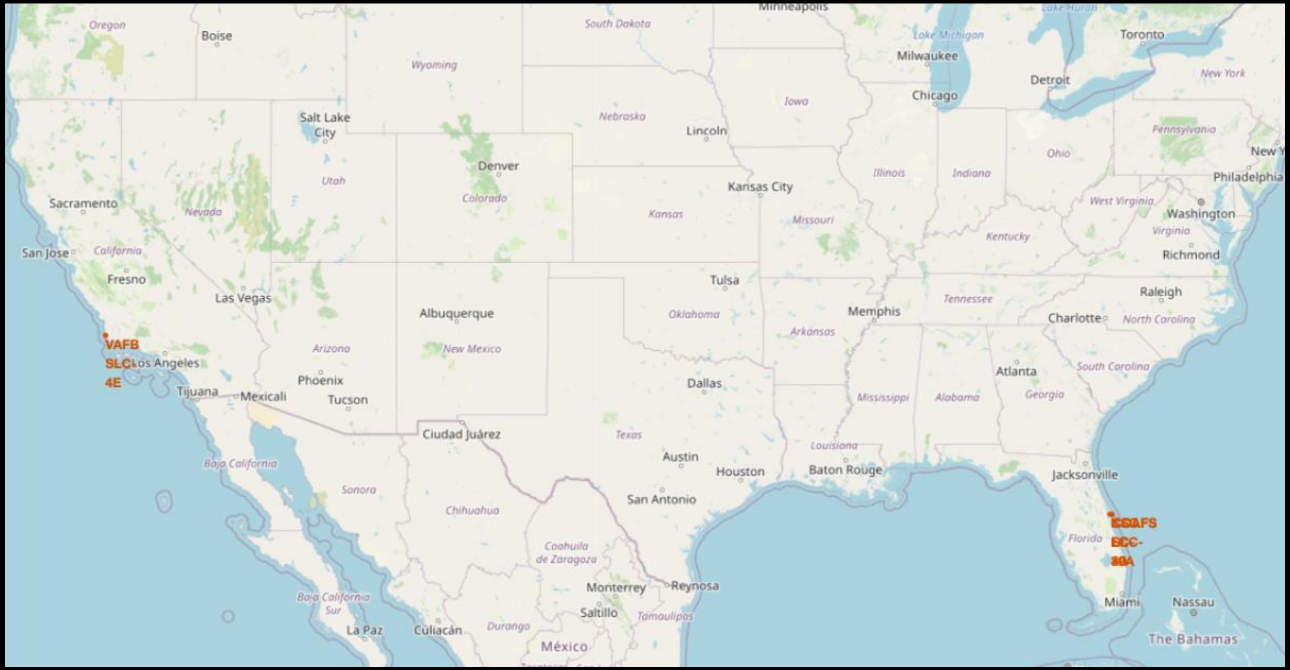
Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

- The query %sql SELECT "LANDING_OUTCOME", COUNT(*) as 'COUNT' FROM SPACEXTBL WHERE substr(Date,1,4) || substr(Date,6,2) || substr(Date,9,2) between '20100604' and '20170320' GROUP BY "Landing_Outcome" ORDER BY "COUNT" DESC ranks the count of landing outcomes (e.g. Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20 in descending order.

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

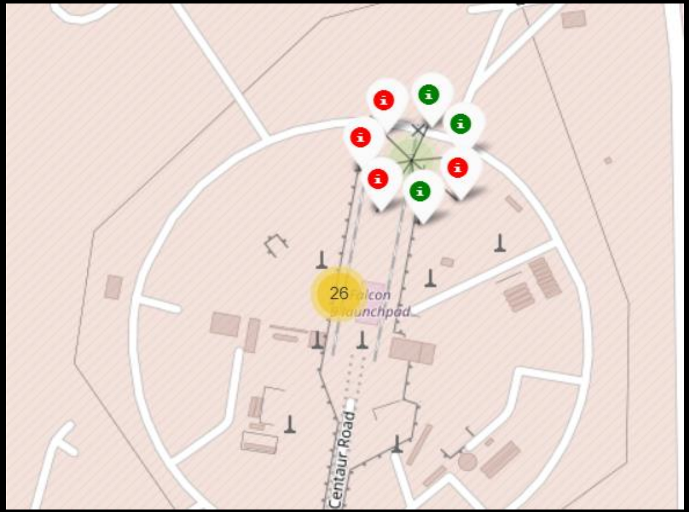
INTERACTIVE MAP WITH FOLIUM

- All the launch sites are on the coasts to Florida and California.



INTERACTIVE MAP WITH FOLIUM

- For successful/failed launches, we use markers for all launch records. A green marker indicates a successful launch while a red marker indicates a failed launch. A launch only happens in one of the four sites, meaning that several launch records will have duplicate coordinates. For this, we use marker clusters.

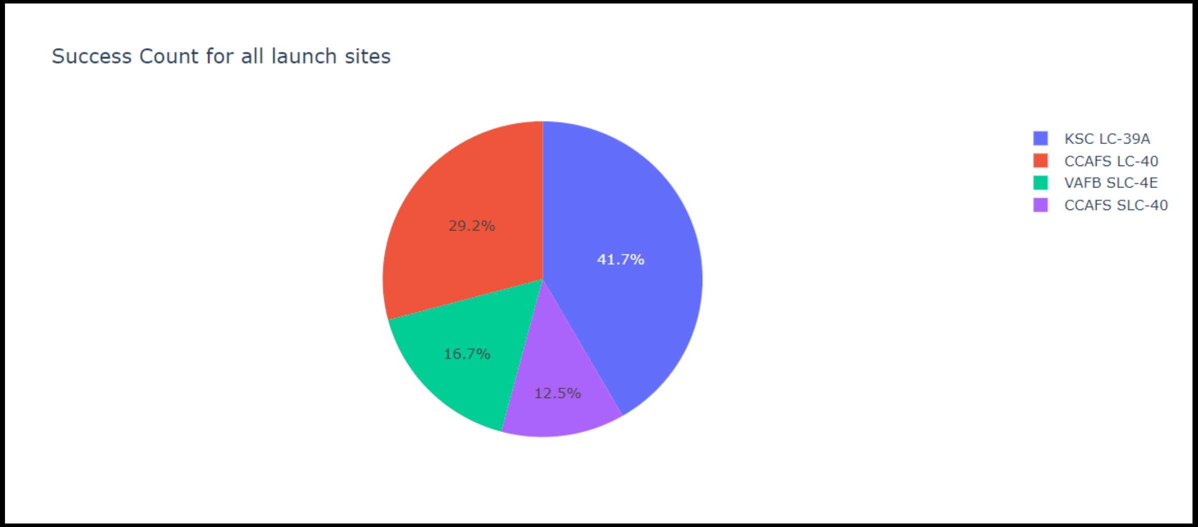


INTERACTIVE MAP WITH FOLIUM

- After plotting distance lines to the proximities, we can see that:
  - The nearest railway is 1.29 km away.
  - The nearest highway is 0.59 km away.
  - The coastline is 0.87 km away.
  - The nearest city (Melbourne) is 51.74 km away.

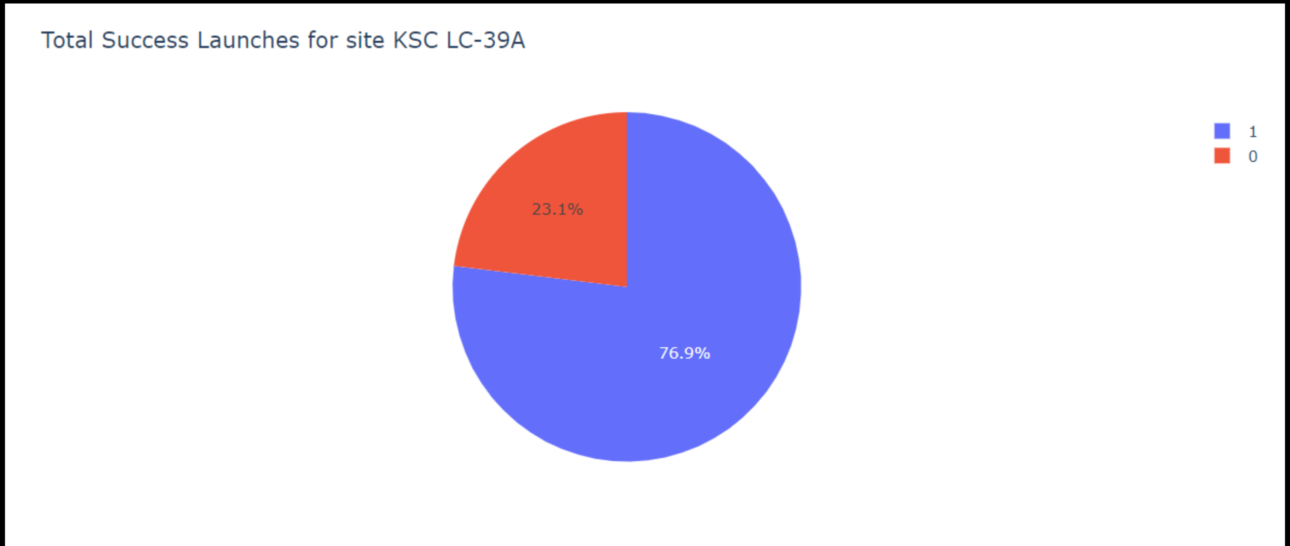
INTERACTIVE DASHBOARD WITH PLOTLY

- We created a pie chart that shows the total success count for all four launch sites or a success count of a site.
- The KSC LC-39A site had 41.7% of successful launches, so it had the most accomplished launches.



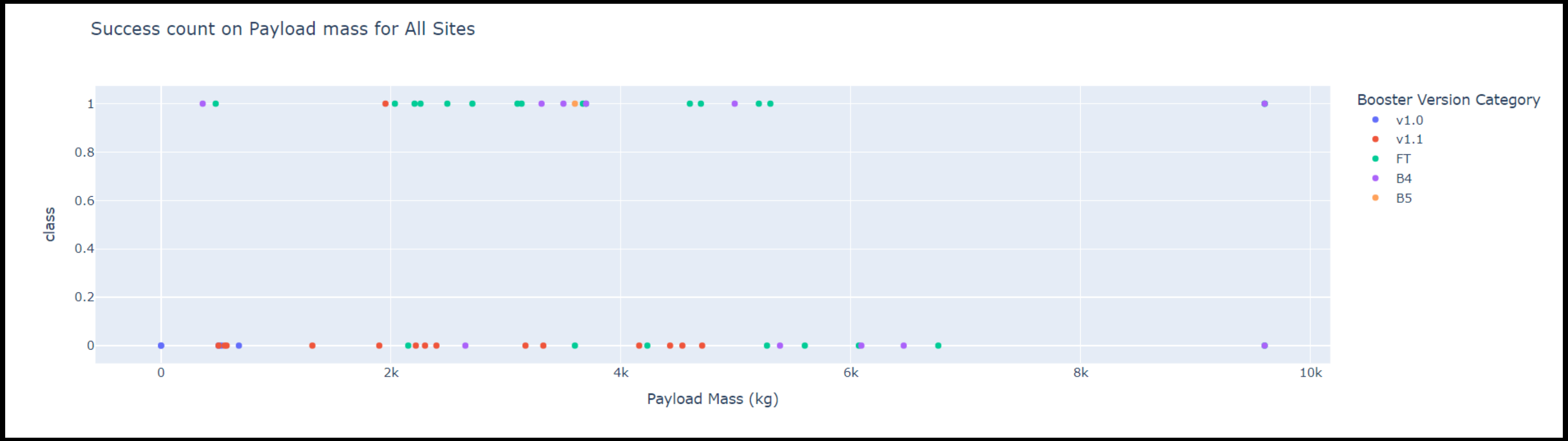
INTERACTIVE DASHBOARD WITH PLOTLY

- The KSC LC-39A site also had 76.9% of successful launches, and its success rate is high enough.



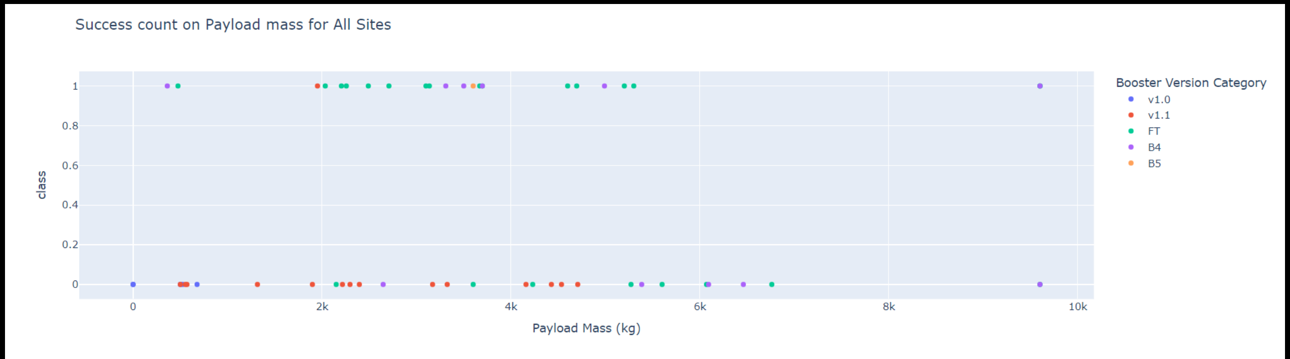
INTERACTIVE DASHBOARD WITH PLOTLY

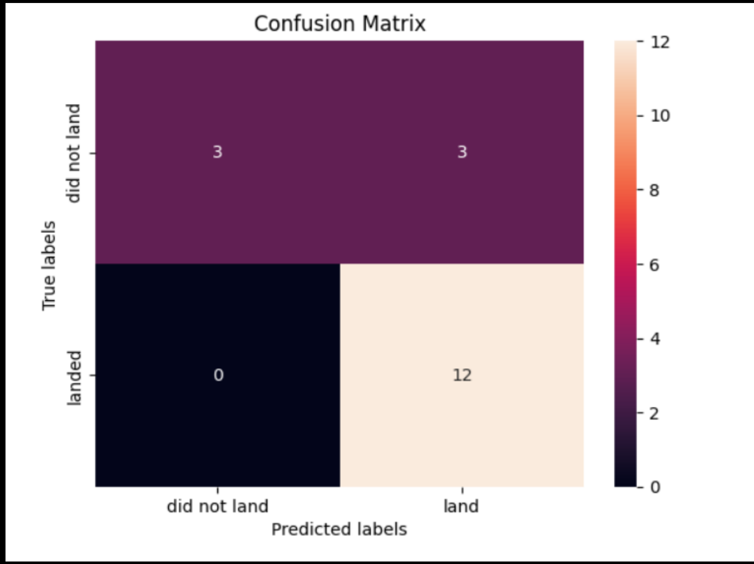
- We created a scatter plot to observe the correlation between payload and mission outcomes for selected site(s).
- A payload range of 10K has the highest launch success rate while a payload range of 4K has the lowest launch success rate. The F9 Booster version v1.0 has the highest success rate.



MACHINE LEARNING PREDICTION ANALYSIS

- Once we find out the accuracy score and best score for each classification algorithm, we get the following:
  - The Decision Tree algorithm has the highest accuracy of 0.94 with a best score of 0.9.
  - If we examine the decision matrix, we see that logistic regression can differentiate between classes. The major problem is that there are false positives.





CONCLUSION

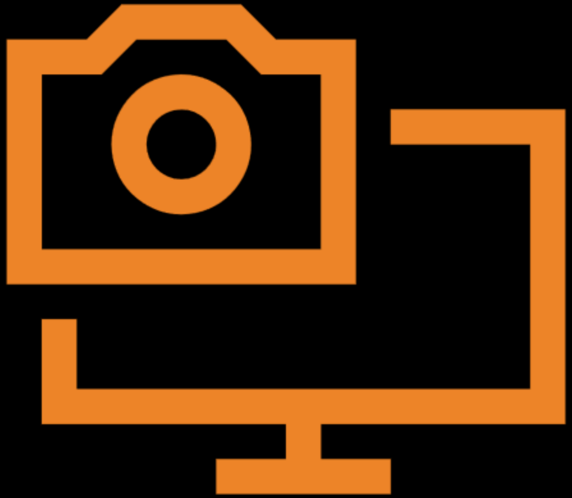
- **Predictive Model Performance:** Our machine learning model earned an accuracy of 90% in predicting Falcon 9 first stage landing success. This cost-effective approach significantly enhances decision-making during launch planning.
- **Cost Savings:** By accurately forecasting landing outcomes, SpaceX can optimize booster reuse, leading to a decrease in overall mission costs. Estimated savings: $$$$$ millions per launch.
- **Future Work:** Refine the model with data that has not been covered (weather conditions, historical launches, etc.). Examine transferability to other rocket systems.
- **Acknowledgments:** Thanks to our team, mentors, and SpaceX for all their support.

APPENDIX

➤ SpaceX Data Collection API Code snippet for IDs given for each launch

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

APPENDIX



➢ Coding logic for parsing and extracting each launch table

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # Append the flight_number into launch_dict with key `Flight No.`
            launch_dict["Flight No."].append(flight_number)

            # Date value
            #Append the date into launch_dict with key `Date`
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            launch_dict["Date"].append(date)

            # Time value
            #Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict["Time"].append(time)

            # Booster version
            #Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            launch_dict["Version Booster"].append(bv)

            # Launch Site
            #Append the bv into launch_dict with key `Launch Site`
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site)

            # Payload
            #Append the payload into launch_dict with key `Payload`
            payload = row[3].a.string
            launch_dict['Payload'].append(payload)

            # Payload Mass
            #Append the payload_mass into launch_dict with key `Payload mass`
            payload_mass = get_mass(row[4])
            launch_dict['Payload mass'].append(payload_mass)

            # Orbit
            #Append the orbit into launch_dict with key `Orbit`
            orbit = row[5].a.string
            launch_dict['Orbit'].append(orbit)

            # Customer
            #Append the customer into launch_dict with key `Customer`
            if row[6].a != None:
                customer = row[6].a.string
            else:
                customer = 'None'

            launch_dict['Customer'].append(customer)

            # Launch outcome
            #Append the launch_outcome into launch_dict with key `Launch outcome`
            launch_outcome = list(row[7].strings)[0]
            launch_dict['Launch outcome'].append(launch_outcome)

            # Booster landing
            #Append the booster_landing into launch_dict with key `Booster landing`
            booster_landing = landing_status(row[8])
            launch_dict['Booster landing'].append(booster_landing)


            print(f"Flight Number: {flight_number}, Date: {date}, Time: {time} \n \
            Booster Version {bv}, Launch Site: {launch_site} \n \
            Payload: {payload}, Orbit: {orbit} \n \
            Customer: {customer}, Launch Outcome: {launch_outcome}\
            Booster Landing: {booster_landing} \n \
            *** ")
```