Space X Falcon 9 Analysis

IBM Data Science Professional Certificate Capstone Project Presentation

Karman Bhinder
22 November 2022

Table Of Contents

A. Executive Summary

B. Introduction

C. Methodology

D. Results

E. Conclusion

F. Appendix

Executive Summary

Summary of Methodologies

The following steps were used in making this project:

1. Data Collection through SpaceX API and Webscraping
2. Data Wrangling
3. Exploratory Data Analysis via sql and data visualisation
4. Interactive Visual Analysis by a Dashboard
5. Predictive Analysis from various classification models.

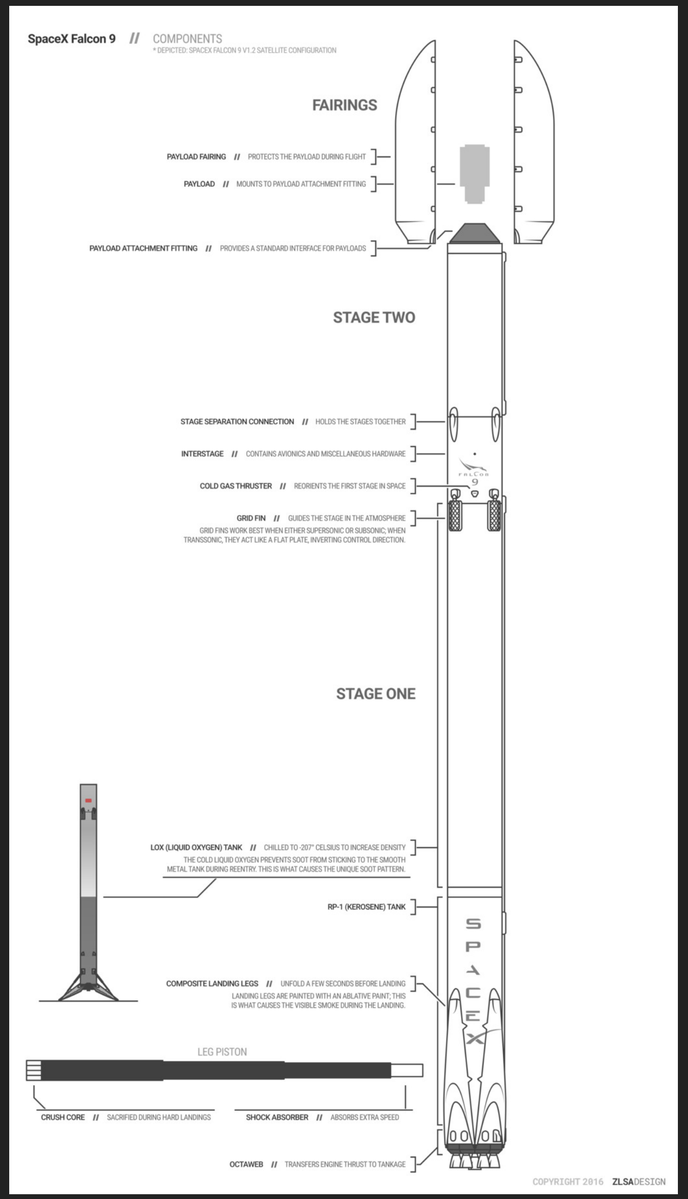
Summary of Results

We trained four different classification models to predict whether or not the Stage 1 of Falcon 9 will land or not to help us determine the cost of our launch. We used features of the likes of Booster Version, Launch Date and Payload Mass to predict the Launch Outcome. We found that the accuracy of our models was comparable. We need more data to account for overfitting in our models

Introduction

Background

- There is a boom in the Commercial Space Travel Industry
- SpaceX is the frontrunner in the industry and has the lowest cost per launch at $62m compared to the market standard of $165m
- Key for SpaceX's Success is its ability to recover and re-use Stage 1 of a rocket during launch.
- This saves them both Time and Money spent in building a New Stage 1 for each rocket which is the largest component of the rocket during launch



Forest Kastch zlsadesign.com

Our mission is to predict whether or not stage 1 of a rocket will land successfully

Methodology

Data Collection

In the Data Collection stage we gathered data by using beautiful soup to webscrape data from Falcon 9's wikipedia page and using GET requests to get data from the SpaceX API.

We retrieved data for the following features:

- Using SpaceX API : FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude, Latitude

- Using Wikipedia for Web scraping: Flight Number, Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Booster Version, Booster landing, Date, Time

Data Collection

SpaceX API

1. Create a GET response and convert it into a json file eventually creating a data frame object

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

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

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

2. Clean the Data and store them in a Dictionary using User defined functions and a user created feature List

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

3. Create a pandas Data Frame using the Dictionary created in step 2.

```
# Create a data from launch_dict
data = pd.DataFrame(launch_dict)
```

4. Deal with missing values for payload mass, filter data for only falcon 9 landings and reset the flight number.

```
data_falcon9 = data [data['BoosterVersion']=='Falcon 9']
```

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
```

```
# Calculate the mean value of PayloadMass column
payloadmean = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass']=data_falcon9['PayloadMass'].replace(np.nan,payloadmean)
```


Github URL

Data Collection

Web scraping From Wikipedia

1. Create a GET response object from static url and use it to create a Beautiful Soup object. Then find all the tables present in the HTML page.

```
page = requests.get(static_url).text
```

```
soup = BeautifulSoup(page,'html.parser')
```

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

2. Extract Column Headers from the soup object for the table.

```
column_names = []

# Apply find_all() function with `th` element on firs
# Iterate each th element and apply the provided extr
# Append the Non-empty column name (`if name is not N
for header in first_launch_table.find_all('th'):
    columnames = extract_column_from_header(header)
    if (columnames != None and len(columnames) > 0):
        column_names.append(columnames)
```

3.Use the Extracted Column names as keys to create a dictionary and use user defined functions to parse data into the keys

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

4. Convert the Dictionary into a Pandas Data Frame

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


Github URL

Data Wrangling

Using Pandas

1. Explore variables Launch Site, Orbit and Outcome using .value_counts( )

```
df['LaunchSite'].value_counts()

CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
```

```
df['Outcome'].value_counts()

True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
```

```
df['Orbit'].value_counts()

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

3.Create a list where the element is 0 if the corresponding row in Outcome is in bad_outcomes, otherwise, it's 1. Store the list in the data frame in a new column named class

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

```
df['Class']=landing_class
```

2.Create a set of outcomes where the second stage did not land successfully

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


Github URL

Exploratory Data Analysis

Data Visualisation

Scatter Plots are used to discover relationships between two numeric variables. We used scatterplots to explore following relationships:

- Flight Number and Launch Site
- Payload and Launch Site
- Orbit Type and Flight Number
- Payload and Orbit Type

Bar Graphs are used to discover relationships between a numeric and a categorical variable. We used a Bar Graph to explore the relationship between Success Rate and Orbit Type

Line Plots are used to discover relationships between a numeric variable over a period of time. We used a line plot to explore the relationship between Success Rate trend over the Years.

Github URL

Exploratory Data Analysis

SQL

We performed the following SQL queries on the data set:

- 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 the average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome on a ground pad was achieved
- List the names of the boosters which had success on a drone ship and a payload mass between 4000 and 6000kg
- List the total number of successful and failed mission outcomes
- List the names of the booster versions which have carried the maximum payload mass
- List the failed landing outcomes on drone ships, their booster versions, and launch site names for 2015
- Rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order

Github URL

Building an Interactive Map

Using Folium

In this stage we created an Interactive map to conduct Visual Analysis:

1. Mark All Launch Sites on a Map

Create a Folium Map Object and mark the Launch sites on the map using both Folium.Marker and Folium.Circle

2. Mark the Success/Fail Launches for each site on the Map

Assign a marker colour as success=green and failure=red to each launch and then cluster them together by adding a Folium.Marker for each launch in MarkerCluster

3. Calculate the Distances between a launch site and its proximities

Create a folium.marker for each proximity then draw a line between the launch site and proximities using Folium.Polyline and add it to the map.

Github URL

Building an Interactive Dashboard

Using Plotly Dash

In this stage we added two plots to an Interactive Dashboard for our presentation:

1. Pie Chart

Used to visualise the success rate for each launch site and also see the distribution of successes across launch sites to determine which launch site is the most successful.

2. Scatter Graph

Used to visualise the relationship between Payload Mass in Kg and the Outcome of a Launch. This could be done over a certain range of the Payload Mass using a slider. The data is also filtered further by the booster version.

Github URL

Predictive Analysis

Using Classification Algorithms from scikitlearn

In this stage we learned different models and evaluated them to find the best one:

1. Model Development

Prepare the data for model deployment by splitting the data into a test and a train set. Then Choose Appropriate Learning Algorithm for our problem. Then we create a GridSearchCV object and a dictionary of parameters for each chosen model, fit the GridSearchCV object to our parameters and use the Training data set to train each model.

2. Model Evaluation

For each algorithm's GridSearchCV object, use best_params_ and best_score_ to find the best parameters and check the accuracy score respectively

3. Finding the Best Model

Compare the accuracy scores for our models and select the one with the highest accuracy score.

Github URL

Results

Exploratory Data Analysis

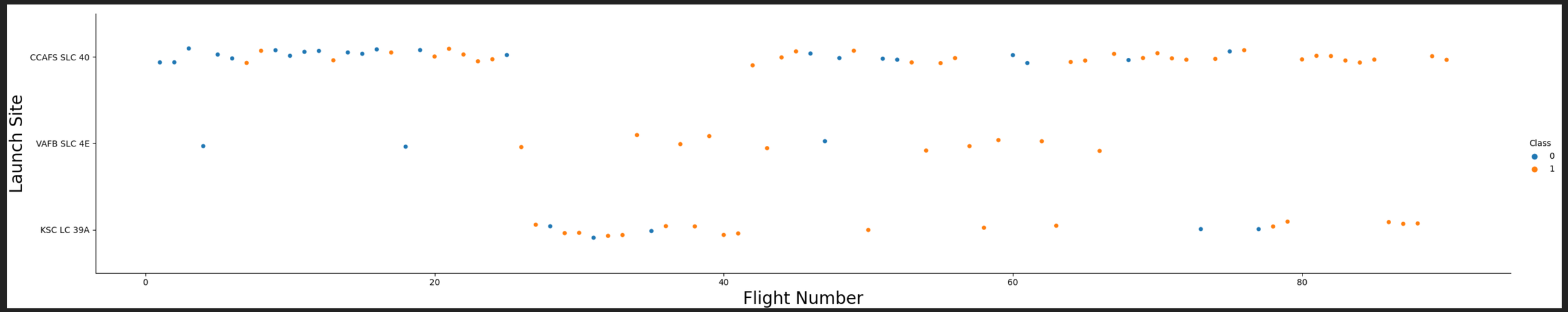
Data Visualisation

- Launch Site Vs Flight Number
- Launch Site Vs Payload Mass
- Success Rate Vs Orbit
- Orbit Vs Flight Number
- Orbit Vs Payload Mass
- Yearly Average Success Rate

Launch Site Vs. Flight Number

I observed the following trends:

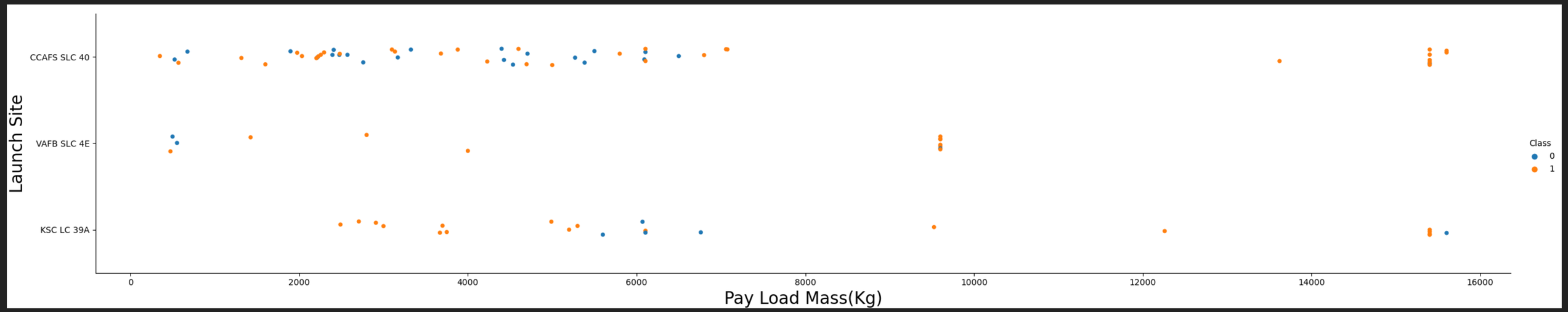
1. Launch Success increases with the number of Flights
2. The KSC LC 39A Launch Site was the last one to be initiated which explains the increase in its success rate
3. VAFB SLC 4E has the least amount of launches.
4. After the 30th Flight, The launch success rate increased notably.



Launch Site Vs. Payload Mass

I observed the following trends:

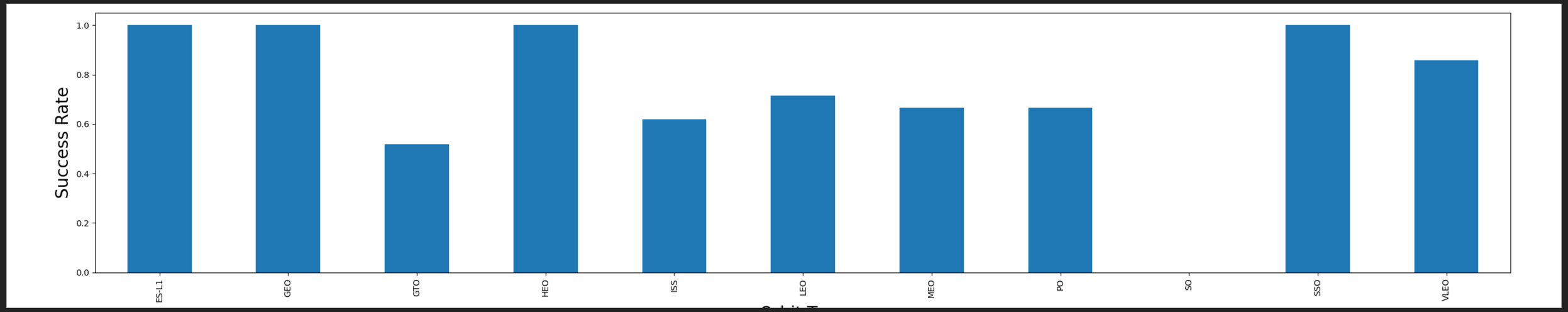
1. I did not observe a relationship between payload mass and launch success rate.
2. After a Payload Mass of 8000 Kg we only had one unsuccessful landing, but that could be attributed to a lack of data points in that region.
3. CCAFS SLC 40 had the most amount of launches.



Success Rate Vs. Orbit

I observed the following trends:

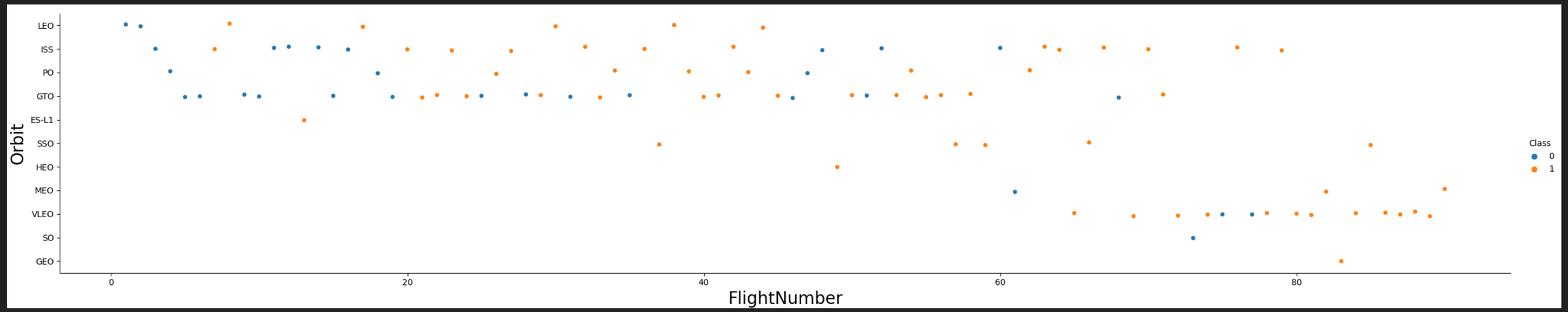
1. I observed that the orbits ES-L1,GEO,HEO and SSO all have a 100% Success Rate.
2. I also noted the the SO orbit had no successful launches



Orbit Vs. Flight Number

I observed the following trends:

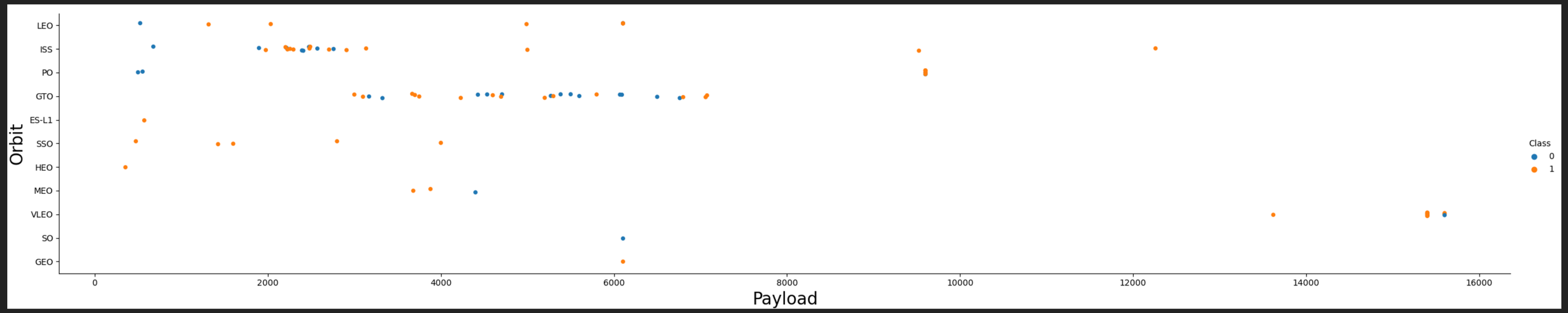
1. The only 100% success rate orbits only had 1 or 2 flights with the exception of SSO with 5 successful flights.
2. If we isolate the GTO and LEO orbits we can observe a linear relationship between flight number and launch outcome.



Orbit Vs. Payload Mass

I observed the following trends:

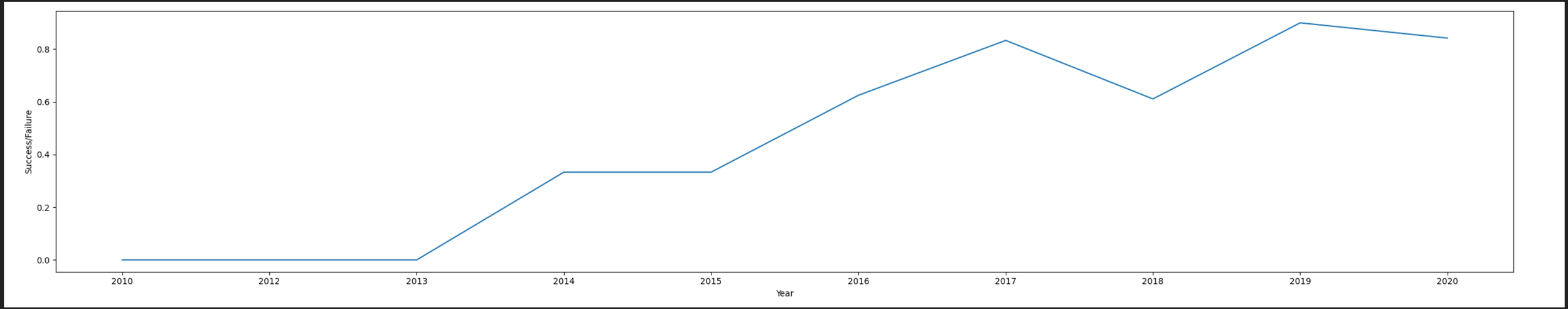
1. VLEO orbits have a very heavy payload.
2. If we isolate the PO,ISS and LEO orbits we can observe a positive linear relationship between payload mass and launch outcome.



Yearly Average Success Rate

I observed the following trends:

1. SpaceX had no successful landings prior to 2013
2. The Success rate generally increased after 2013
3. There were dips in the Success Rate in 2018 and 2020
4. SpaceX maintained a 50% Success Rate after 2016



Exploratory Data Analysis

SQL

- All Site Launch Names
- Launch Site Names beginning with CCA
- Total Payload Mass
- Average Payload Mass
- First Successful Ground Landing Date
- Successful Drone Ship Landing with payload between 4000 and 6000 Kg
- Total Failure/Successful Mission outcomes
- Boosters Carrying Maximum Payload
- 2015 Launch Records
- Rank Landing Outcomes Between 2010-06-04 And 2017-03-20

All Site Launch Names

Find the names of the unique launch sites

The Query uses the Distinct function to return unique values of the column Launch Site from the table SPACEXTBL

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

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

Launch Site Names beginning with ‘CCA’

Find 5 records where launch sites begin with `CCA`

The Query uses the Limit function to limit the output table to just 5 rows and uses the like function to return rows where the string values of the column Launch Site from the table SPACEXTBL begin with ‘CCA’. In our query the % sign is a wild card and is used when we do not know the complete string value

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

Date	Time (UTC)	Booster_Version	Launch_Site
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40

Total Payload Mass

Calculate the Total Payload carried by boosters from NASA

The Query uses the Sum function to return the sum of the values of the column payload mass from the table SPACEXTBL where the condition is met that the customer is NASA

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "TOTAL PAYLOAD MASS" FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';
```

TOTAL PAYLOAD MASS
45596

Average Payload Mass

Calculate the Average Payload carried by boosters from NASA where booster version is F9 V1.1

The Query uses the Avg function to return the average of the values of the column payload mass from the table SPACEXTBL where the condition is met that the the booster version is F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "AVERAGE PAYLOAD MASS" FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.1%';
```

AVERAGE PAYLOAD MASS
2534.6666666666665

First Successful Ground Landing Date

Find the first day when the launch outcome was successful on ground pad

The Query uses the Min function to return the minimum value of Date column from the table SPACEXTBL where the condition is met that the launch outcome is 'Successful (ground pad)'

```
%sql SELECT MIN(Date) FROM SPACEXTBL WHERE "Landing _Outcome" = "Success (ground pad)";
```

MIN(Date)
01-05-2017

Successful Drone Ship Landing with payload between 4000 and 6000 Kg

List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000.

The Query uses the AND clause in the Where condition to apply multiple conditions on our query.

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE "Landing _Outcome" = 'Success (drone ship)' AND 4000 < PAYLOAD_MASS__KG_ < 6000;
```

Booster_Version
F9 FT B1021.1
F9 FT B1022
F9 FT B1023.1
F9 FT B1026
F9 FT B1029.1
F9 FT B1021.2
F9 FT B1029.2
F9 FT B1036.1
F9 FT B1038.1
F9 B4 B1041.1
F9 FT B1031.2
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

Total Failure/Successful Mission outcomes

Calculate the total number of successful and failure mission outcome.

The Query uses the Count function to return the number of each values of the column Mission Outcome from the table SPACEXTBL and it uses group by function to combine the different outcomes.

```
%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

Boosters Carrying Maximum Payload

List the names of boosters which have carried the maximum payload mass

The Query uses a sub query to get the required conditions for our where clause.

```
%sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = ( SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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 failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015.

We use the as clause to specify column name and substr(Date,4,2) and substr(Date,7,4) to extract month and year respectively

```
%%sql
SELECT substr(Date, 4, 2) AS Month,"Landing _Outcome", BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXTBL
WHERE "Landing _Outcome" = 'Failure (drone ship)'
    AND substr(Date,7,4) = '2015';
```

Month	Landing _Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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.

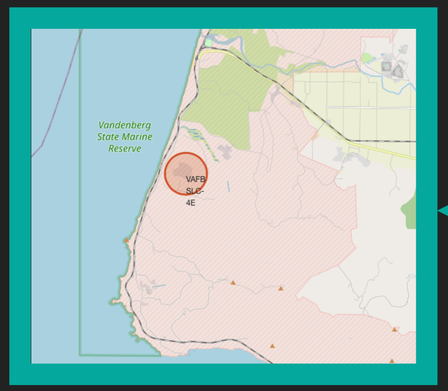
We use the between clause to specify a range of date values as a condition for our where clause.

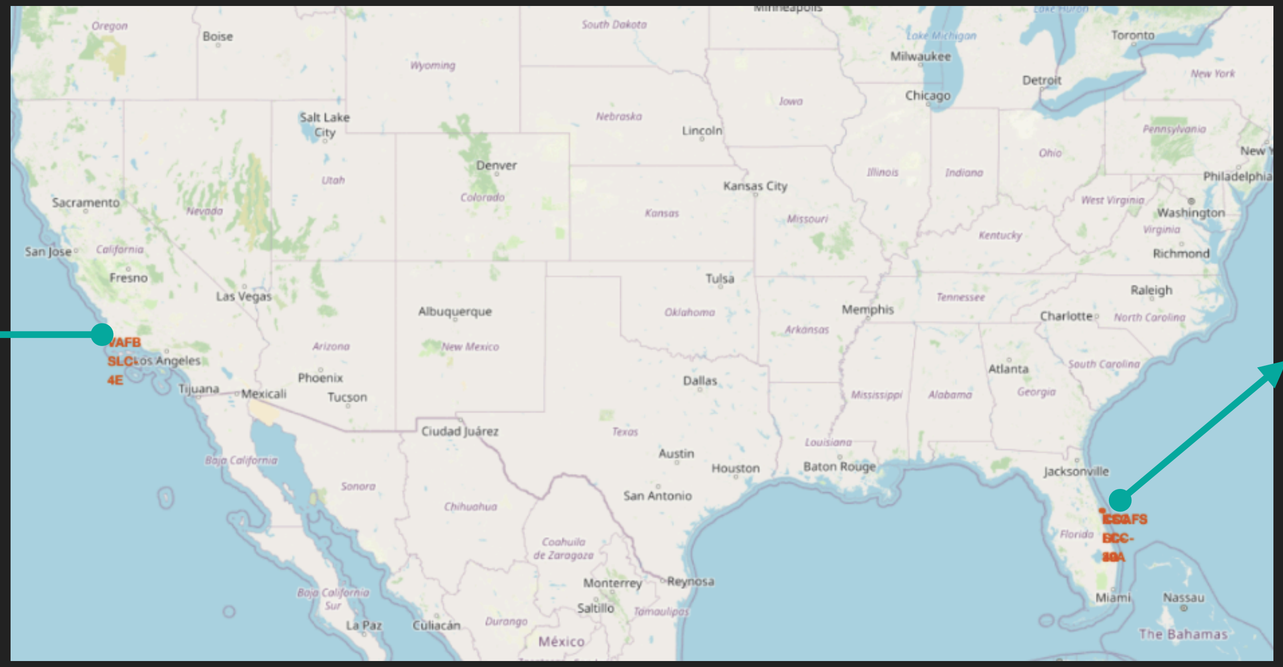
```
%sql SELECT "Landing _Outcome", COUNT("Landing _Outcome") AS TOTAL_NUMBER FROM SPACEXTBL WHERE DATE BETWEEN '04-06-2010' and '20-03-2017'
```

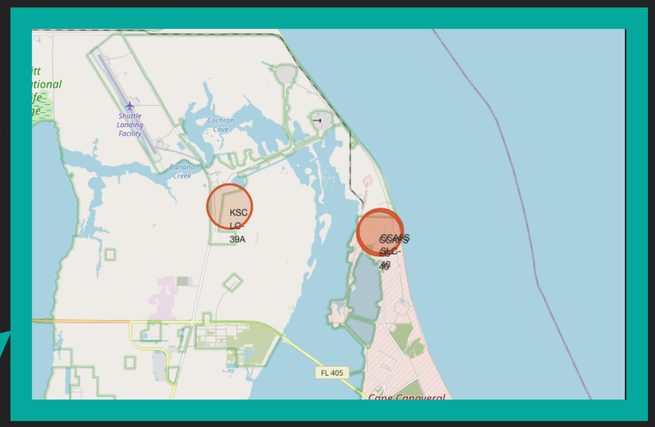
Landing _Outcome	TOTAL_NUMBER
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

Launch Sites Proximity Analysis

All Launch Sites on a Map



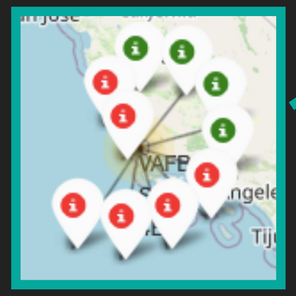


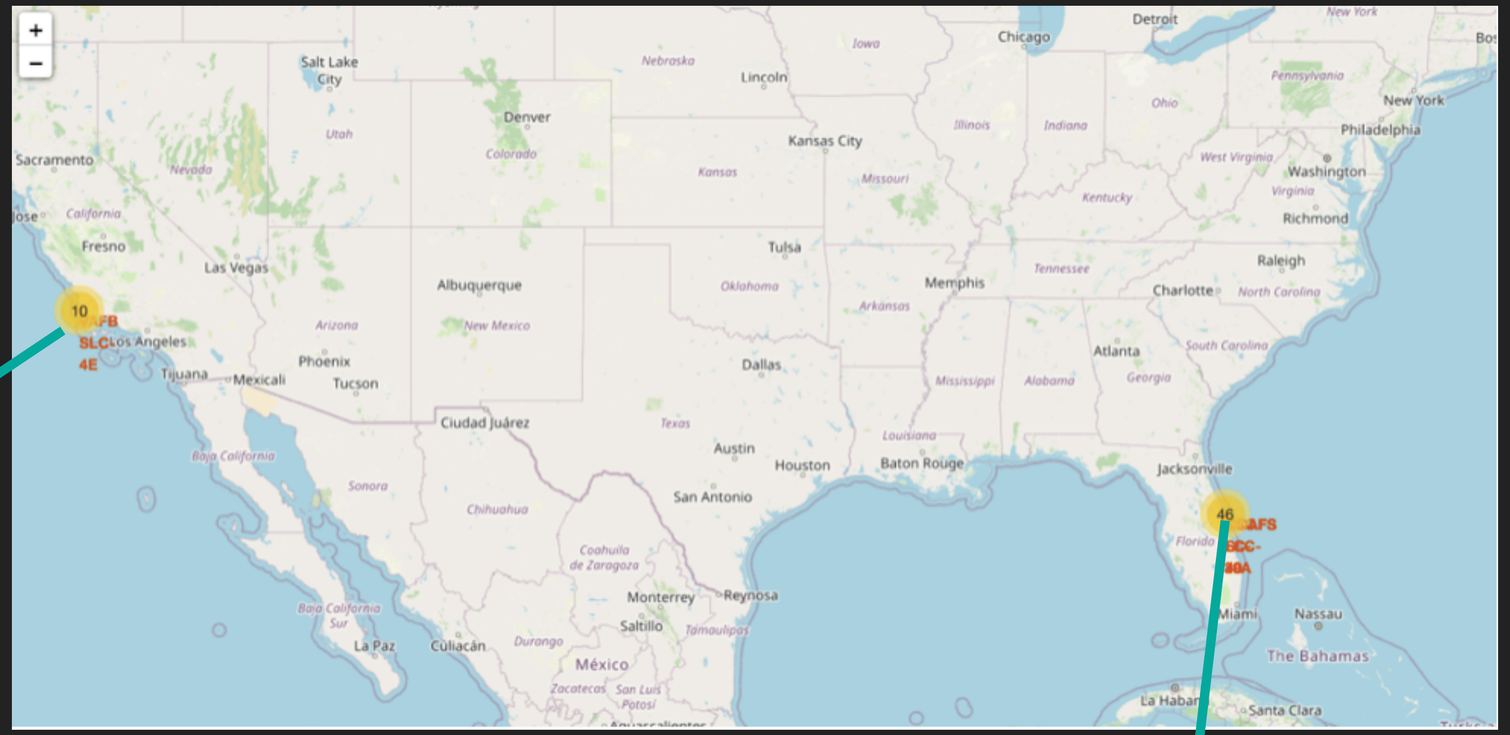


The Launch Sites are in Coastal regions most likely due to suitable climate to support and increase the chances of a successful launch

Success/Failure for Launch Sites

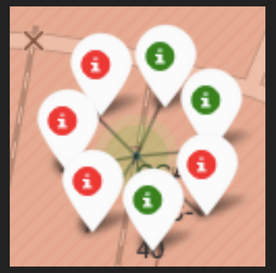
VAFB SLC 4E



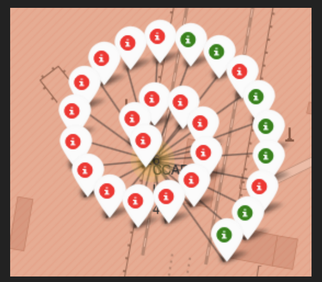


KSC LC 39A

CCAFS SLC 40

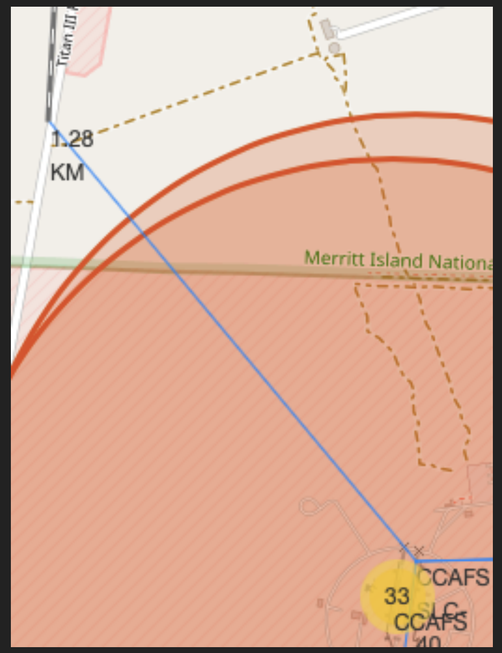


CCAFS LC 40

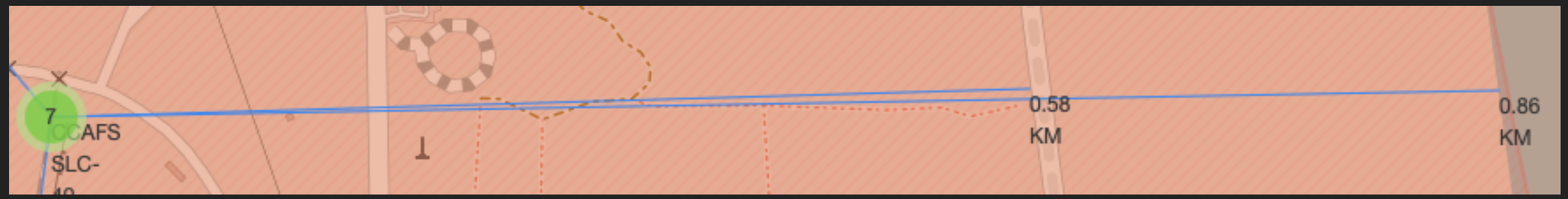


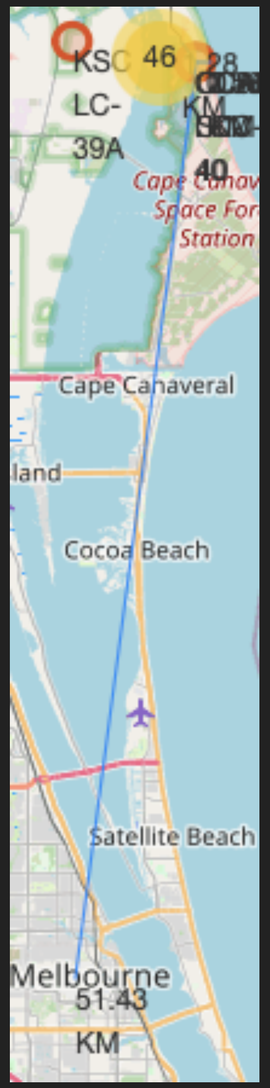
Launches have been grouped into clusters by launch sites, and marked with green icons for successful launches, and red icons for failed launches.

Proximity of Launch Site to Points of Interest



We use the CCAFS SLC 40 as an example and learn about points of interest in near proximity of a launch site. We observe that Launch Sites are in close proximity with modes of transport such as railways(1.28km), highways(0.58km) and the coastline(0.86km), but it tends to be farther away from a city, in our example Melbourne the nearest city is 51.43km away.

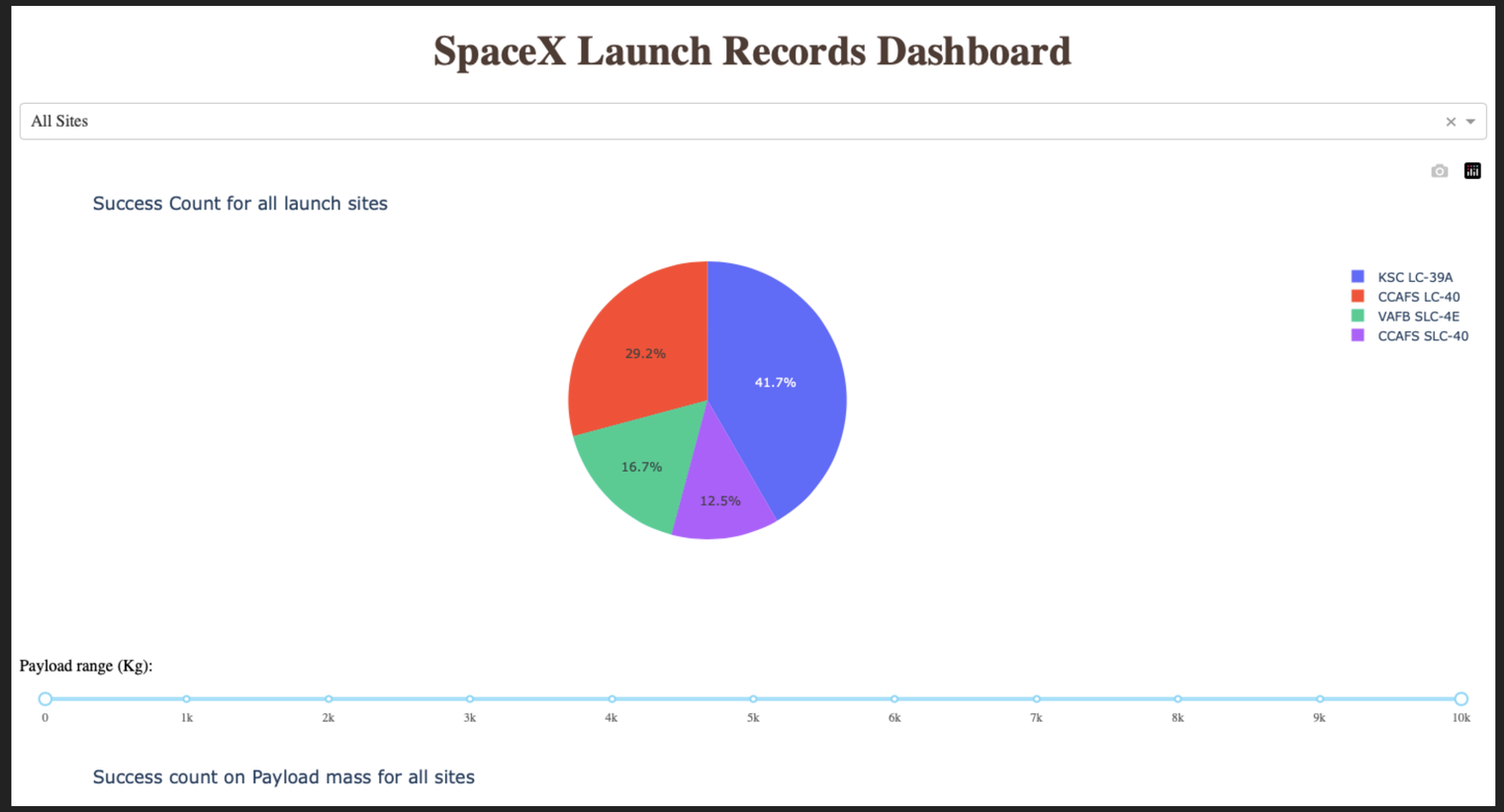




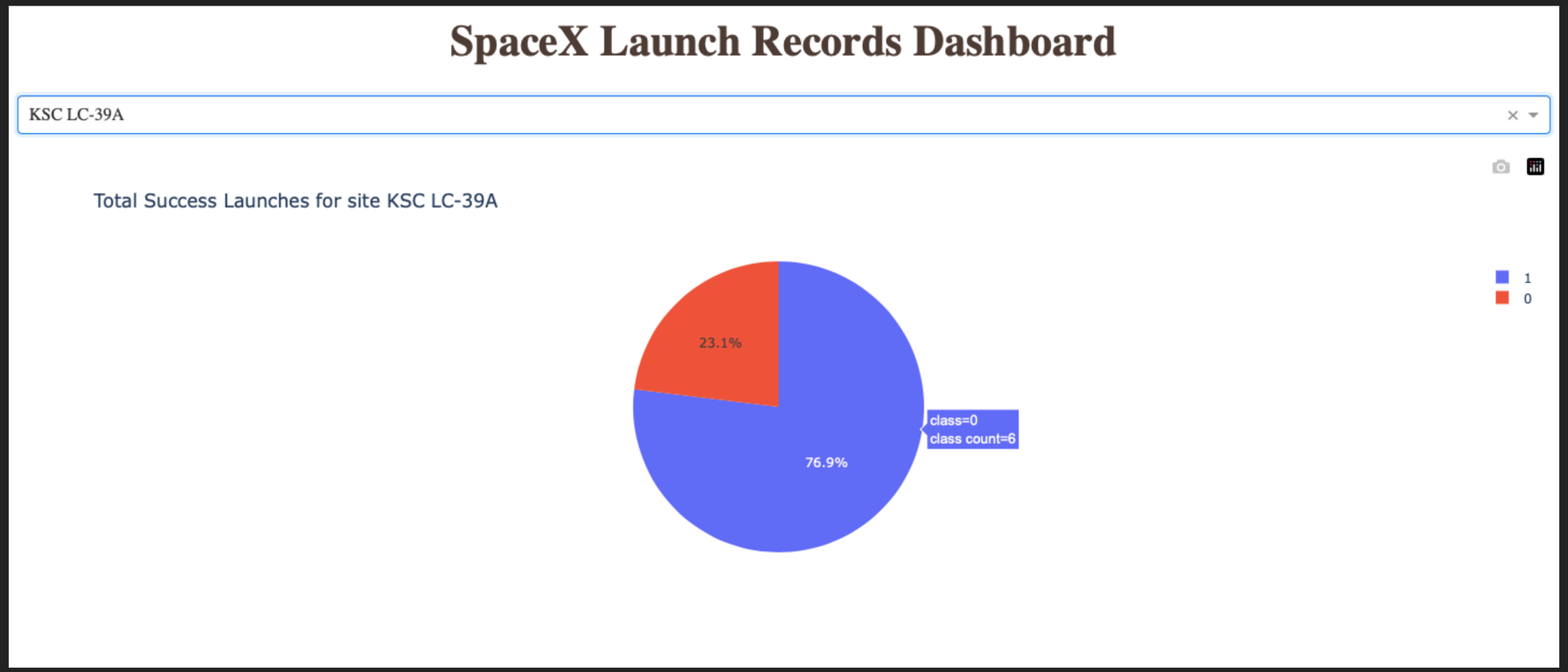
Launch Records Dashboard

Distribution of Launch Successes by Launch Site

We found that the launch site KSC LC-39A had the most number of successful launches.

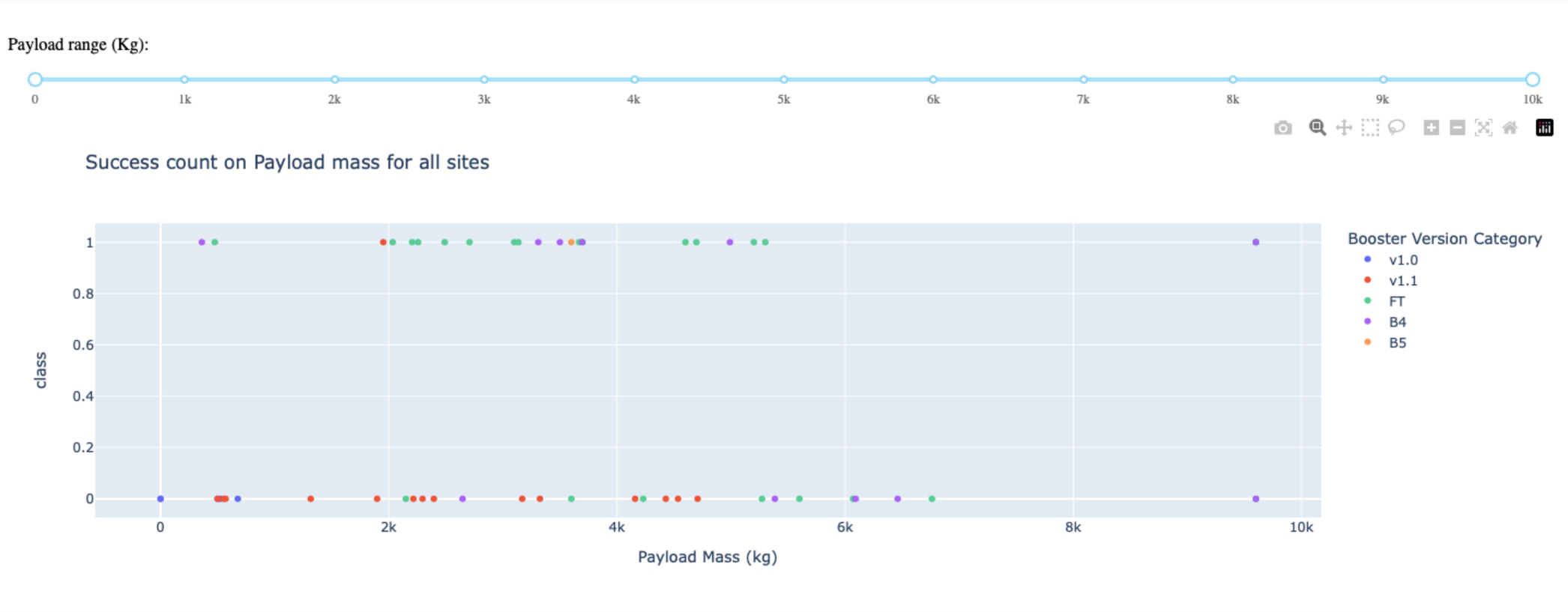


Pie Chart of Launch Site KSC LC-39A



We found that the launch site KSC LC-39A also had the highest success rate at 76.9%

Launch Outcome vs Payload



We found that higher payloads had a lower chance of success and we also noticed that some booster versions only launched with lower payloads.

Predictive Analysis

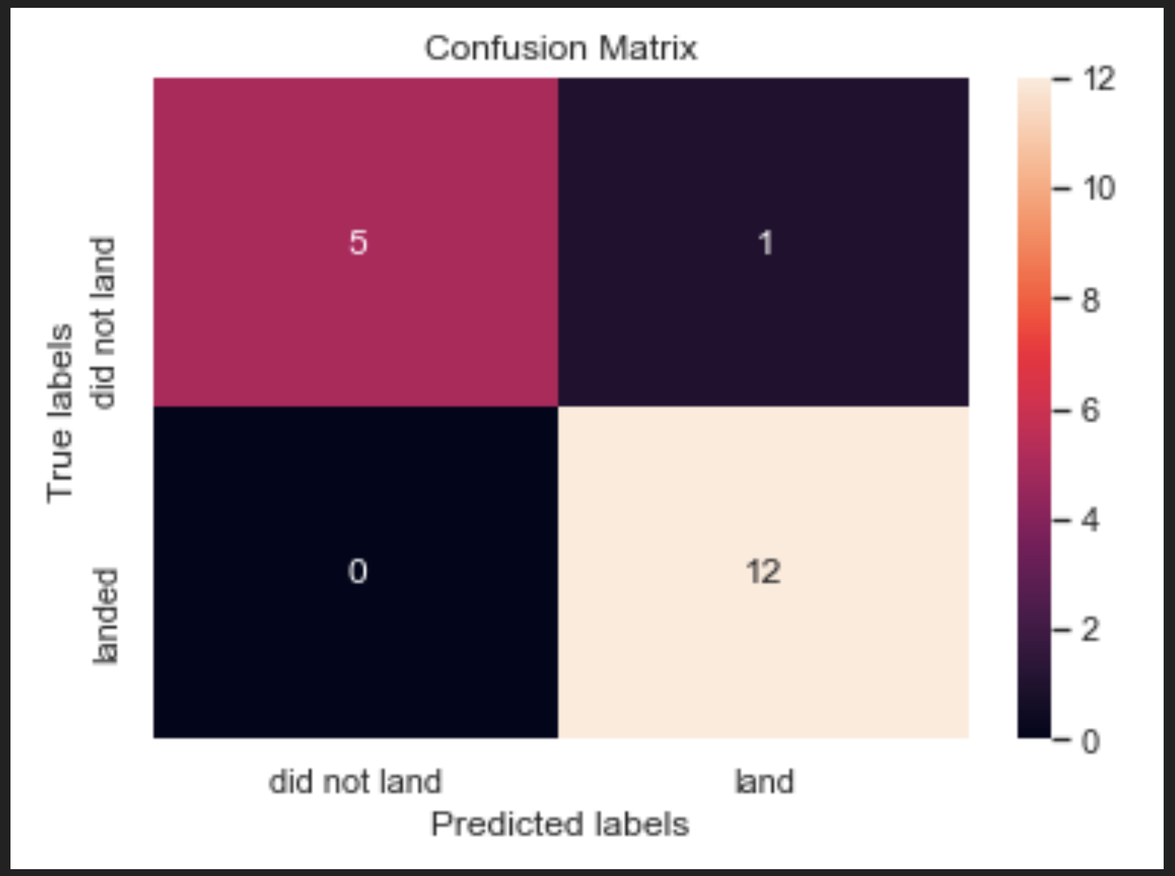
Comparing Accuracy Scores

Algorithm	Accuracy Score	Best Score
Logistic Regression	83.33	0.8464
Support Vector Machine	83.33	0.8482
Decision Tree	94.44	0.9000
K nearest neighbours	83.33	0.8482

We observe that the model with the best score score used the Decision Tree algorithm with an accuracy score of 94.44% and The best score of 0.9000

Distribution of Launch Successes by Launch Site

The confusion matrix supports the accuracy score of 94.44% by correctly classifying 17 (5 + 12) of the 18 total samples in our test set.



Conclusions

After our analysis, we are able to conclude the following:

1. As the flight number Increases, so does the chances of a launch being successful

- We observed a positive relationship between flight number and Successes in the Launch Site vs Flight Number scatter plot.
- We also saw a general growth trend in the yearly average success rate line plot.

2.There is a relationship between Orbit Type and Launch Outcome, Payload Mass

- We saw ES-L1,GEO,HEO and SSO all had a 100% success outcome but all of them had only one launch apart from SSO which stands out because it had 5 successful launches.
- Orbits PO,ISS and LEO increase their chances of success with heavier payloads
- VLEO orbits have a very heavy payload.

3. Launch Site KSC LC-39A is the most successful launch site with 41.7% of total successful launches being launched here and it also has a launch success rate of 76.9% which is the highest for a launch site.

4. Launch Sites generally located in the coastal areas and tend to be in close proximity to modes of transport such as railways, highways, and coastlines but far away from densely populated areas such as cities.

5. There is a relationship observed between Launch Outcome and Payload Mass. As the payload mass increases, chances of a successful launch decreases.

6. The best algorithm to train a model to predict whether a launch will have a success or failure outcome is the Decision Tree Algorithm.

Appendix

Data Collecting
SpaceX API
Additional Functions Defined
to clean and extract data

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.

```
# 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'])
```

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

```
# 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 mul
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)]
```

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

```
# 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.

```
# 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.

```
# 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'])
```

Data Collecting

Wikipedia Web scraping

Method to parse data and fill the launch data dictionary

```python
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
            # TODO: Append the flight_number into launch_dict with key `Flight No.
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            #print(date)

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

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

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

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

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

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

            # Customer
            # TODO: Append the customer into launch_dict with key `Customer`
            customer = row[6].text.strip()
            launch_dict['Customer'].append(customer)
            #print(customer)

            # Launch outcome
            # TODO: 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)
            #print(launch_outcome)

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

Thank You!