IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Izet Dautovic

02/17/2022

Outline

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

Executive Summary

Summary of methodologies

- Data Collection through API
- Data Collection with Web Scraping
- Data Wrangling
- Exploratory Data Analysis with SQL
- Exploratory Data Analysis with Data Visualization
- Interactive Visual Analytics with Folium
- Machine Learning Prediction

Summary of all results

- Exploratory Data Analysis result
- Interactive analytics in screenshots
- Predictive Analytics result

Introduction

Project background and context

Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch.

What is the problem, what we trying to find out?

- What factors determine if the rocket will land successfully?
- The interaction amongst various features that determine the success rate of a successful landing.
- What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

Data collection methodology:

- Data collected utilizing SpaceX API and web scraping from Wikipedia.

Perform data wrangling

- One-hot encoding applied to categorical features and dropping irrelevant columns

Perform exploratory data analysis (EDA) using visualization and SQL

- Utilizing Scatter and Bar Charts to show relationships and patterns between variables.

Perform interactive visual analytics using Folium and Plotly Dash

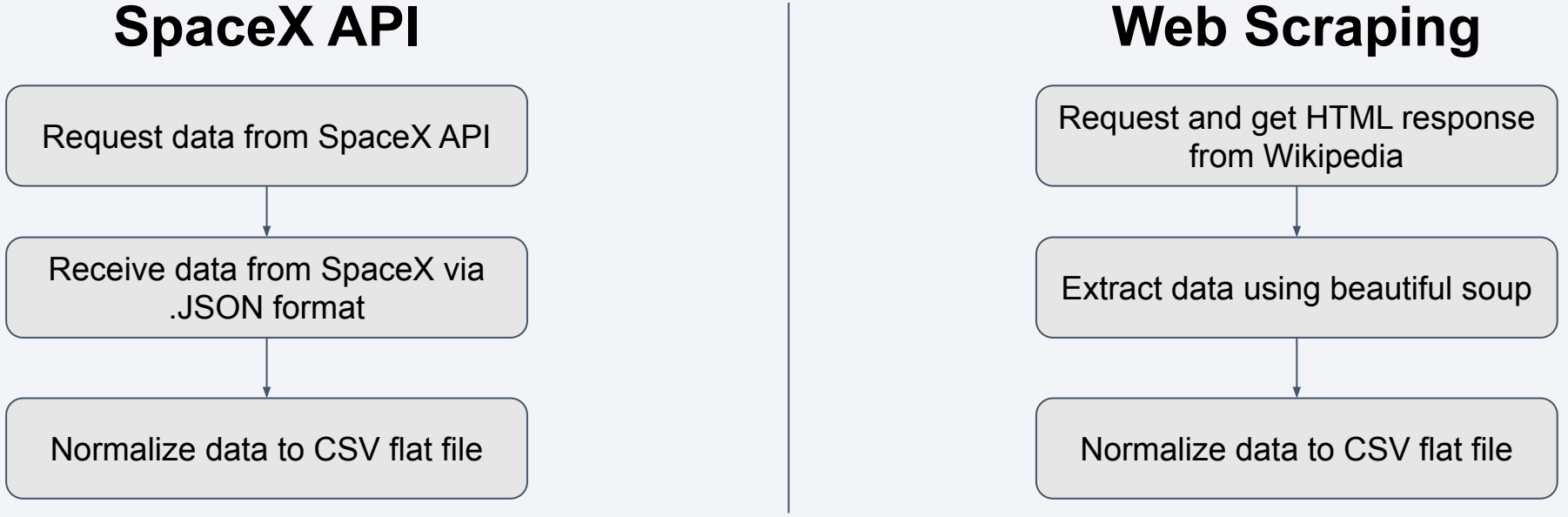
Perform predictive analysis using classification models

Data Collection

For our project, we utilize two data set;

1. SpaceX REST API
2. Wikipedia table by web scraping method to extract data from the web.

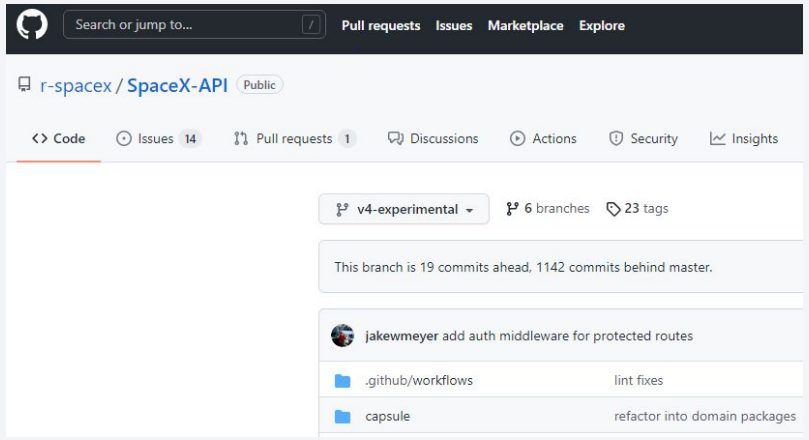
Process used to obtain data set as follow:



Data Collection – SpaceX API

We used the GET request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.

SpaceX API



[GitHub URL]

[Gists URL]

1. Load request library and request rocket data from SpaceX API

```
# Requests allows us to make HTTP requests
import requests
```

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

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

2. Converting response to a JSON format

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

3. Apply predefined helper functions to clean data

def getBoosterVersion(data) | def getLaunchSite(data) | def getPayloadData(data) | def getCoreData(data):

4. Assign list to dictionary and create new data frame

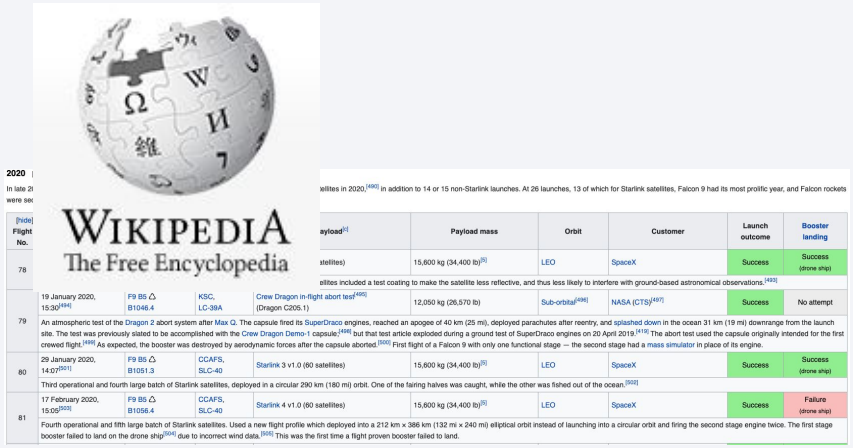
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,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

5. Filter, replace missing data and export final .CSV flat file

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = Launch_data.loc[(Launch_data['BoosterVersion'] == 'Falcon 9')]

data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

We applied web scraping technique to webscrap Falcon 9 launch records from Wikipedia and pursed with BeautifulSoup into the table and then converted to a pandas dataframe.



1. **Request the HTML page from the URL and get a response object**

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

HTTP_Response = requests.get(static_url)
```

2. **Create BeautifulSoup Object and find table**

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(HTTP_Response.content, 'html.parser')

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

3. **Getting Columns Names and creation of dictionary**

```
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

```
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. **Append Data to keys and convert to dataframe**

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

5. **Extract data from datafram to .CSV flat file**

```
df = pd.DataFrame.from_dict(launch_dict)
df.to_csv('spacex_web_scraped.csv', index=False)
```

[GitHub URL]
[Gists URL]

Data Wrangling

We perform Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example:

Outcomes:

- True Ocean = successfully landed to a specific region of the ocean
- False Ocean = unsuccessfully landed to a specific region of the ocean
- True RTLS = successfully landed to a ground pad
- False RTLS = unsuccessfully landed to a ground pad
- True ASDS = successfully landed on a drone ship
- False ASDS = unsuccessfully landed on a drone ship

In this lab we will convert outcomes into Training Labels as:

1 = successfully landed
0 = unsuccessful landing

[GitHub URL]
[Gists URL]

1. **EDA: Load Data, identify data type and if any data missing**

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.
```

```
df.isnull().sum()/df.count()*100
```

```
df.dtypes
```

2. **Calculate the number of launches at each site**

```
df.LaunchSite.value_counts()
```

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

3. **Calculate the number of occurrence of each orbit**

```
df.Orbit.value_counts()
```

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

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

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

4. **Create a landing outcome label from outcome column**

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

5. **Calculate success rate and export dataframe to .CSV**

```
df["Class"].mean()
```

```
df.to_csv("dataset_part_2.csv", index=False)
```

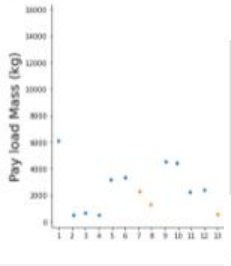
EDA with Data Visualization - Scatter Chart

We explored the data by visualizing the relationship between:

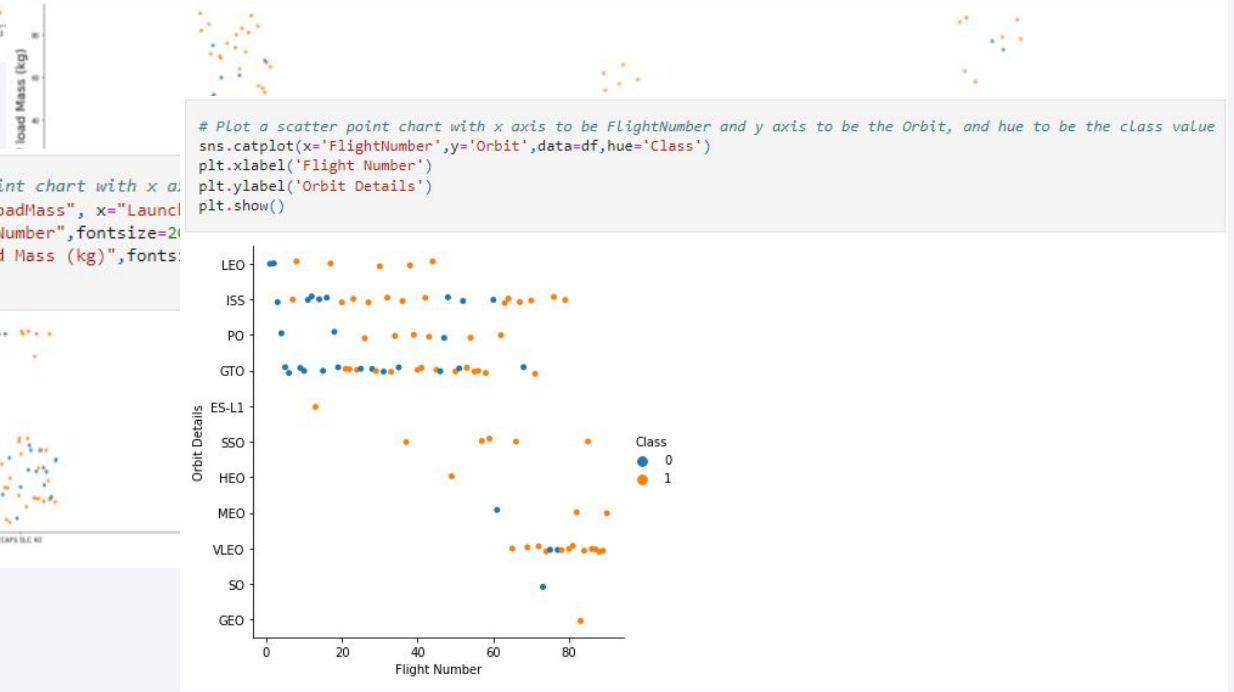
- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Flight Number vs. Orbit Type
- Payload vs. Orbit Type

A scatter plot shows how much one variable is affected by another. The relationship between two variables is called a correlation. This type of chart is generally composed of large data set.

```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

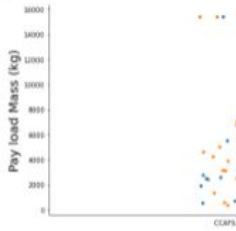



```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class
sns.catplot(y="FlightNumber",x="LaunchSite",hue='Class',data=df, aspect=5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```




```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(x='FlightNumber',y='Orbit',data=df,hue='Class')
plt.xlabel('Flight Number')
plt.ylabel('Orbit Details')
plt.show()
```

```
# Plot a scatter point chart with x a
sns.catplot(y="PayloadMass", x="Launcl
plt.xlabel("Flight Number",fontsize=2
plt.ylabel("Pay load Mass (kg)",fonts
plt.show()
```



[GitHub URL]

[Gists URL]

EDA with Data Visualization - Bar & Line Chart

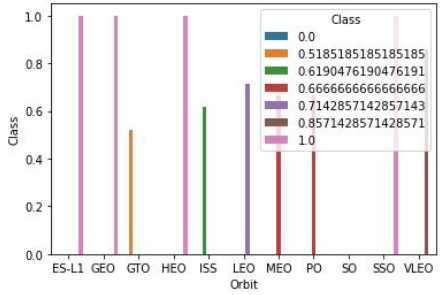
We explored the data by visualizing the relationship between:

- Orbit Type vs. Success Rate

We used a bar chart which makes it easy to compare datasets between multiple groups at a glance. One axis represents a category and the other axis represents a discrete value. The purpose of this chart is to indicate the relationship between the two axes.

```
# HINT use groupby method on Orbit column and get the mean of Class column
orbit_success = df.groupby('Orbit').mean()
orbit_success.reset_index(inplace=True)
sns.barplot(x="Orbit",y="Class",data=orbit_success,hue='Class')
```

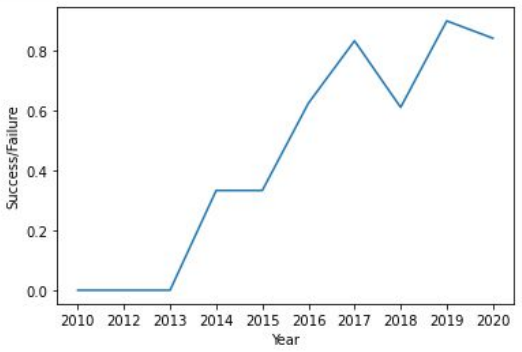
```
<AxesSubplot:xlabel='Orbit', ylabel='Class'>
```



- Year vs Success Rate

For this analysis we use a line chart which shows data variables and trends very clearly and helps predict the results of data that has not yet been recorded.

```
# Plot a line chart with x axis to be the extracted year and
plt.plot(average_by_year["Year"],average_by_year["Class"])
plt.xlabel("Year")
plt.ylabel("Success/Failure")
plt.show()
```



GitHub URL
Gists URL

EDA with SQL

First, we created table within Db2 database and then we loaded the SpaceX dataset.

Second, we applied EDA with SQL to get insight from the data. We completed following queries:

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC'
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date when the first successful landing outcome in drone ship was achieved
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- List the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

GitHub URL
Gits URL

Build an Interactive Map with Folium

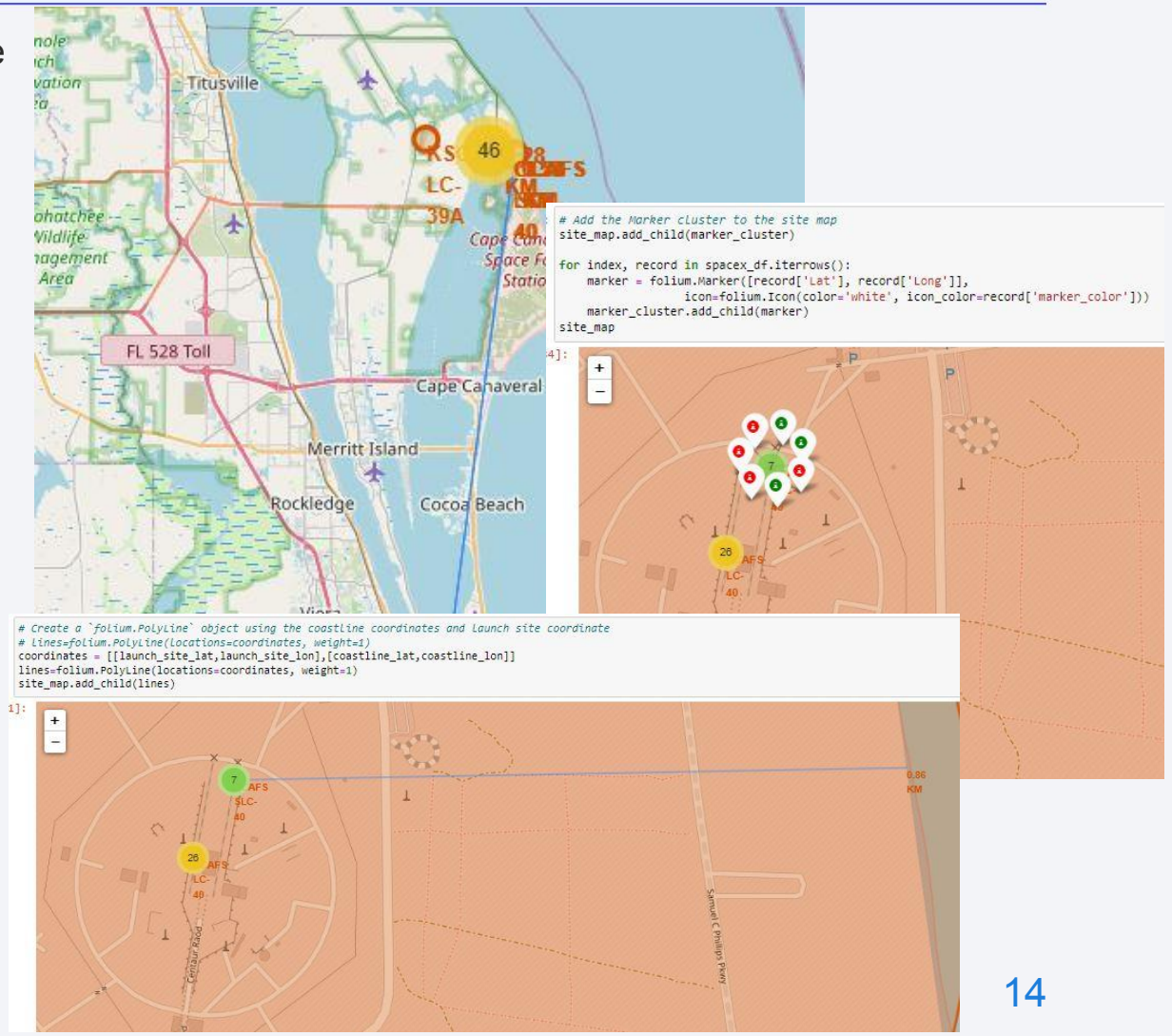
We took the Latitude and Longitude coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.

We created and added following object to a folium map:

- Markers that show all launch sites on a map
- Success/failed launches markers for each site as (1,0) or Green and Red markers utilizing MarkerCluster()
- Lines that show the distances between a launch site

By adding these objects we were able identify following geographical patterns about launch sites:

- Are launch sites in close proximity to railways? Yes
- Are launch sites in close proximity to highways? Yes
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

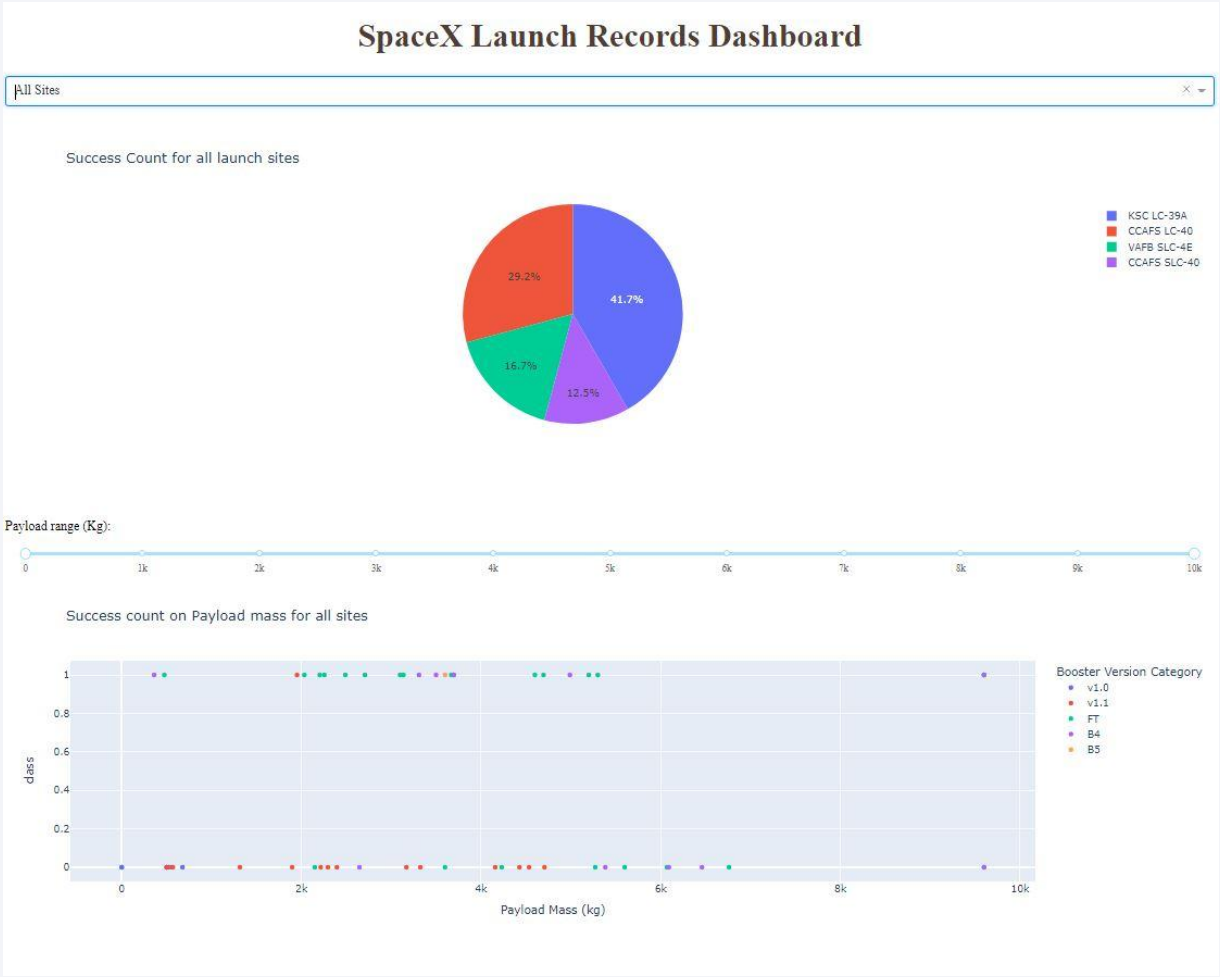


[GitHub URL]
[Gits URL]

Build a Dashboard with Plotly Dash

We built an interactive dashboard with Plotly dash

- Pie charts showing the total launches by a certain sites
- Scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.



Code Only:
[GitHub URL]
[Gists URL]

Predictive Analysis (Classification)

Building Model

- Load dataset into NumPy and Pandas
- Standardize and transform Data
- Split data into training and test data sets
- Check how many test samples we have
- Set regression object, then create GridSearchCV object
- Fit datasets into the GridSearchCV objects and train our dataset.

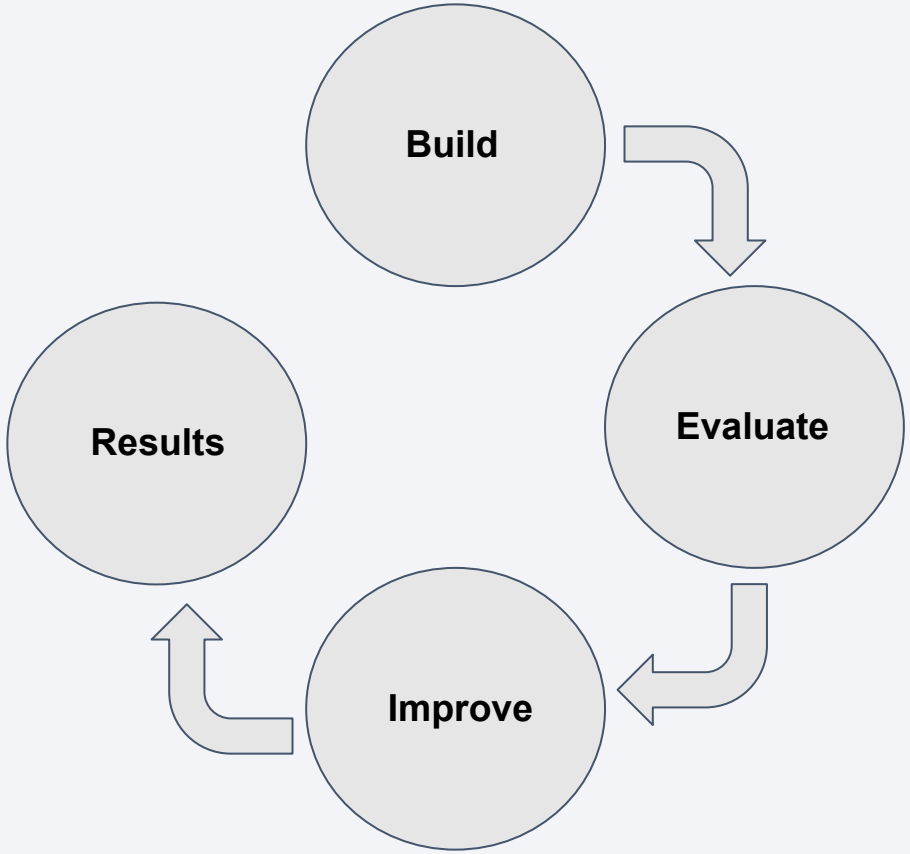
Evaluating Model

- Check accuracy for each model
- Plot Confusion Matrix

Improve Model

- Algorithm Tuning

Finding the best performing model
The best accuracy score wins the best performing model



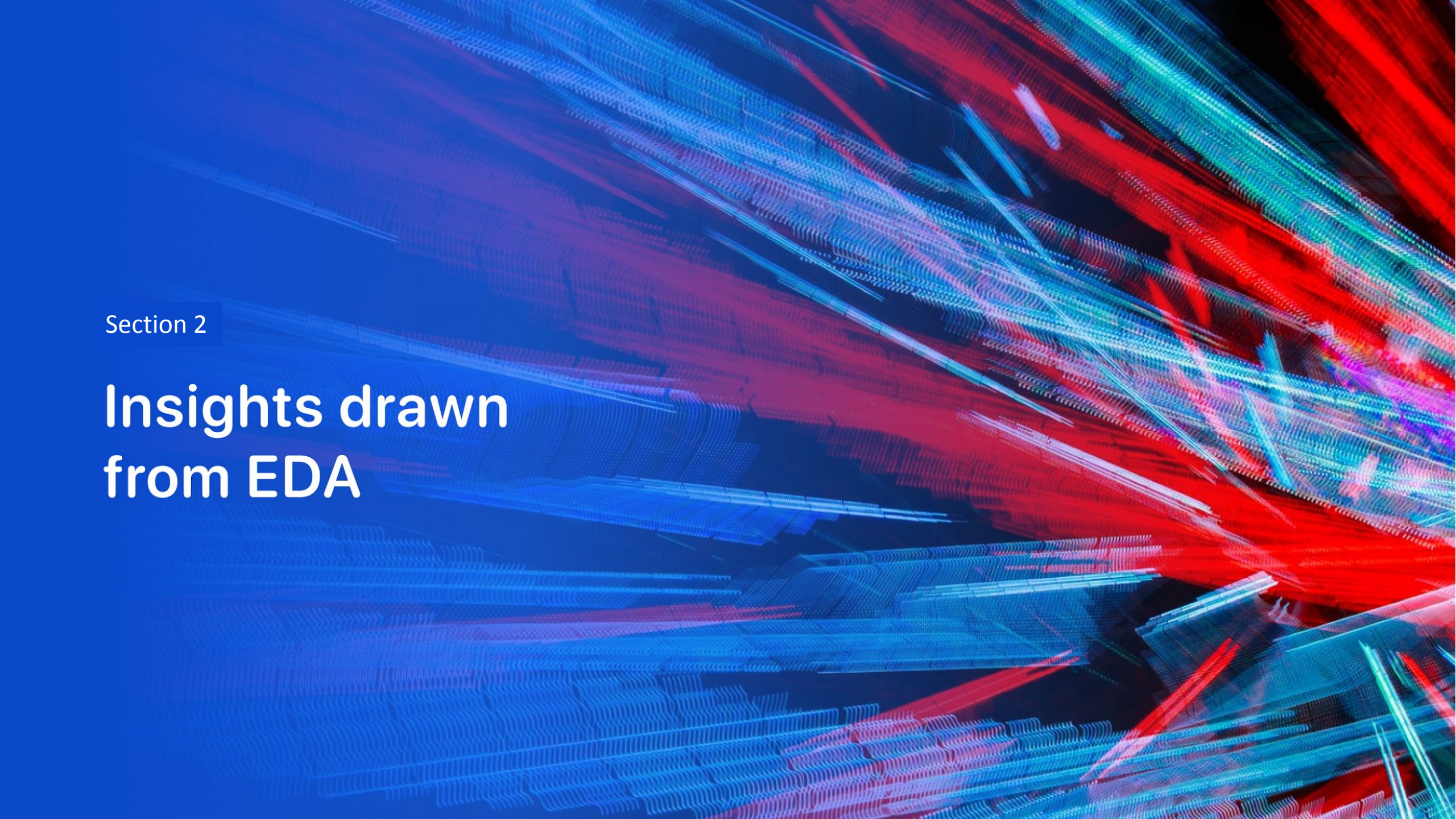
GitHub URL
Gists URL

Results

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

Section 2

Insights drawn from EDA

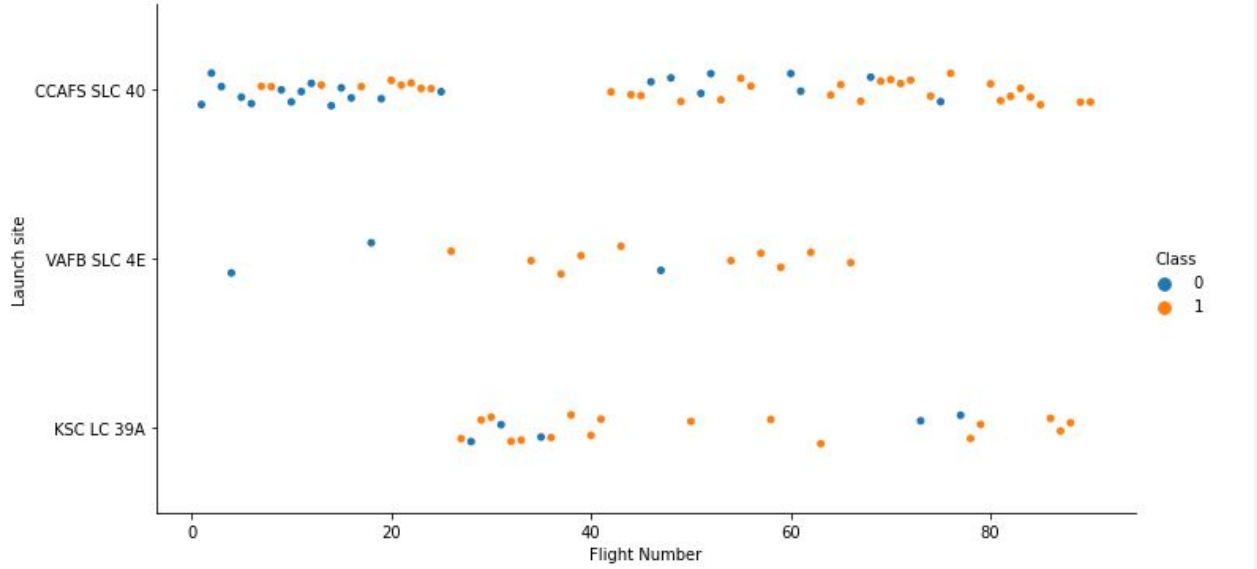


Flight Number vs. Launch Site

This figure shows that the success rate increased as the number of flights increased for each site. Also, there seem to be increase in success rate after 20th flight.

Legend:
Class 0 (**blue**) = unsuccessful launch
Class 1 (**orange**) = successful launch.

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be t
sns.catplot(x="FlightNumber",y="LaunchSite",hue='Class',data=df, aspect=2)
plt.xlabel("Flight Number")
plt.ylabel("Launch site")
plt.show()
```

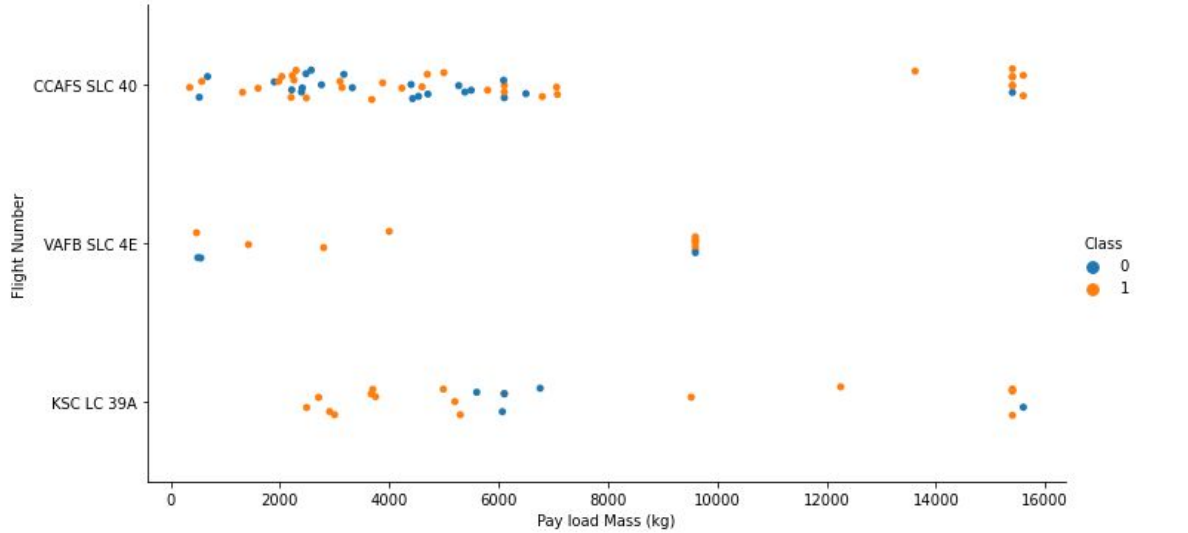


Payload vs. Launch Site

We can see that for the VAFB-SLC launchsite there are no rockets launched for heavy payload mass >10,000 (kg). Also we can see there is no significant or clear pattern between Launch Site and Payload Mass for a successful launch.

Legend:
Class 0 (**blue**) = unsuccessful launch
Class 1 (**orange**) = successful launch.

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to
#sns.catplot(x="PayloadMass", y="LaunchSite", hue="Class", data=df, aspect = 5)
sns.catplot(x="PayloadMass", y="LaunchSite", hue="Class", data=df, aspect=2)
plt.ylabel("Flight Number")
plt.xlabel("Pay load Mass (kg)")
plt.show()
```



Success Rate vs. Orbit Type

Highest Success Rate

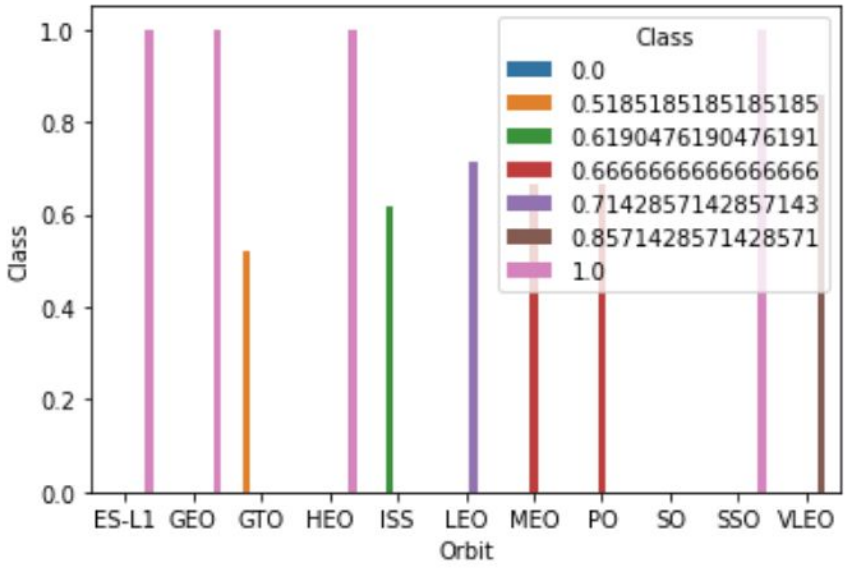
Orbit types SSO, HEO, GEO, and ES-L1 have the highest success rates (100%).

Lowest Success Rate

Orbit types GTO, ISS, MEO (<70%).

```
# HINT use groupby method on Orbit column and get the mean of Class column
orbit_success = df.groupby('Orbit').mean()
orbit_success.reset_index(inplace=True)
sns.barplot(x="Orbit",y="Class",data=orbit_success,hue='Class')
```

```
<AxesSubplot:xlabel='Orbit', ylabel='Class'>
```

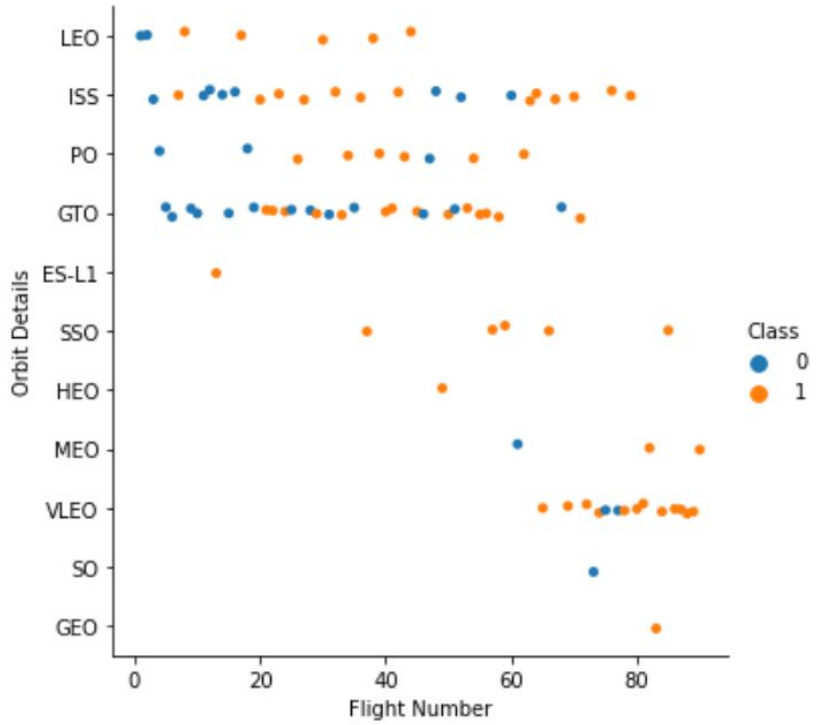


Flight Number vs. Orbit Type

- It appears SpaceX started with LEO orbit with moderate success rate.
- SSO orbit seems to be successful, although number of light seems to be low.
- GTO and ISS orbit, there seems to be no relationship between flight numbers and success rate.
- VLEO has a high success rate, is used the most in recent launches.

Legend:
Class 0 (**blue**) = unsuccessful launch
Class 1 (**orange**) = successful launch.

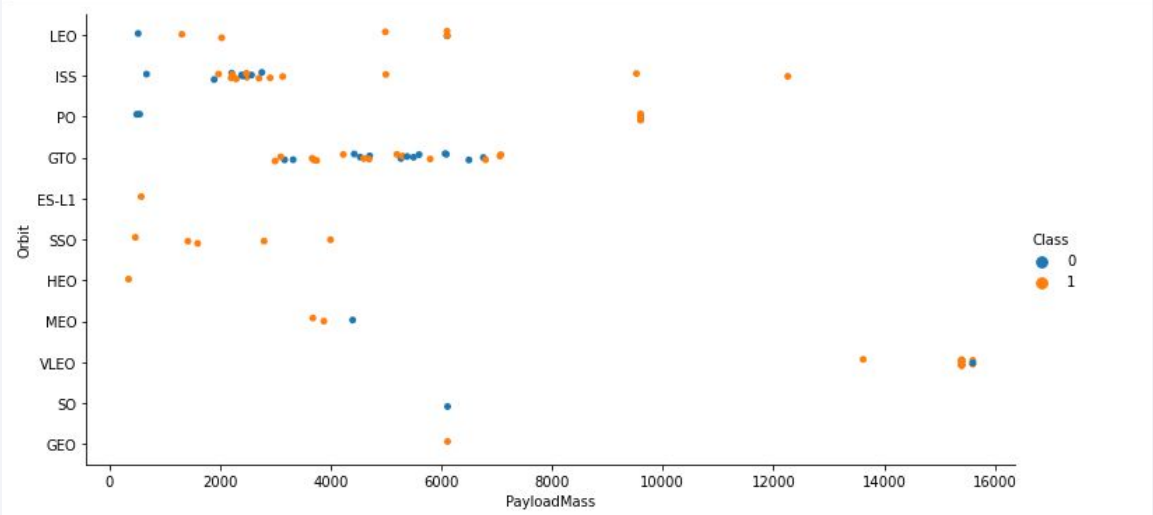
```
# Plot a scatter point chart with x axis to be FlightNumber
sns.catplot(x='FlightNumber',y='Orbit',data=df,hue='Class')
plt.xlabel('Flight Number')
plt.ylabel('Orbit Details')
plt.show()
```



Payload vs. Orbit Type

- LEO orbit has moderate success rate for payload less than 6K kg
- SSO orbit seems to be successful for lower payload, we can observe that no payload is sent greater than 5K kg.
- GTO and ISS orbit, there seems to be no relationship between flight numbers and success rate.
- VLEO has a high success rate, is used the most in recent launches.

```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(x='PayloadMass',y='Orbit', hue='Class', data=df, aspect=2)
plt.xlabel('PayloadMass')
plt.ylabel('Orbit')
plt.show()
```

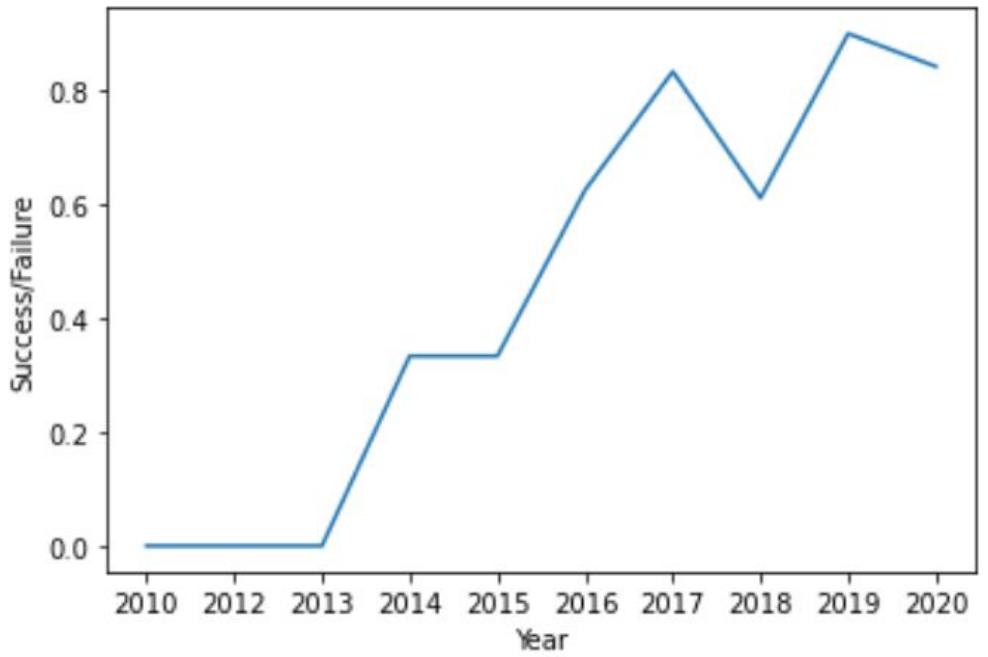


Legend:
Class 0 (**blue**) = unsuccessful launch
Class 1 (**orange**) = successful launch.

Launch Success Yearly Trend

We can observe that success rate since 2013 kept increasing till 2020, this is good new for future flights.

```
# Plot a line chart with x axis to be the extracted year and
plt.plot(average_by_year["Year"],average_by_year["Class"])
plt.xlabel("Year")
plt.ylabel("Success/Failure")
plt.show()
```



All Launch Site Names

To identify unique launch site, we use sql **DISTINCT** clause in the query.

```
%sql select distinct(LAUNCH_SITE) from SPACEXTBL
```

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

Launch Site Names Begin with 'KSC'

To find records where launch sites' names start with `KSC` we use LIKE operator and percent sign (%) after keyword, for example; "**like 'KSC%'**" This will give us all records where "launch site" starts with "KSC"

To limit number of records in the query we use "LIMIT" operator, for example "**limit 5**"

```
%sql select * from SPACEXTBL where LAUNCH_SITE like 'KSC%' limit 5
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2017-02-19	0001-01-01 14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	0001-01-01 06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2017-03-30	0001-01-01 22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-01-05	0001-01-01 11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	0001-01-01 23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Total Payload Mass

To calculate total payload carried by boosters from NASA we use SUM() function.

We have to include WHERE clause, this will calculate total payload only from NASA

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where CUSTOMER = 'NASA (CRS)'
```

Payload
45596

Average Payload Mass by F9 v1.1

To calculate average payload mass carried by booster version F9 v1.1 we use AVG() function and WHERE clause to identify Booster version.

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where BOOSTER_VERSION = 'F9 v1.1'
```

```
 * ibm_db_sa://xzb03393:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00
ludb
Done.
```

Payload

```
2928
```

First Successful landing outcome in drone ship

To find first successful landing outcome on drone ship we use MIN() function to find the earliest date in the column DATE and used WHERE clause to find landing outcome is “Drone Ship”

```
%sql select min(DATE) from SPACEXTBL where Landing__Outcome = 'Success (drone ship)'
```

```
 * ibm_db_sa://xzb03393:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

Min Date

```
2016-05-27
```

Successful Ground Pad with Payload between 4000 and 6000

To find the list of the names of boosters which have successfully landed on ground pad and had payload mass greater than 4000 but less than 6000 we use WHERE clause to find lading outcome is success for Ground Pad and we use AND and greater than “>” and Less Than (<) operator to identify payload mass.

```
%sql select BOOSTER_VERSION from SPACEXTBL where Landing__Outcome = 'Success (ground pad)' and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000
```

```
 * ibm_db_sa://xzb03393:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

booster_version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

To calculate the total number of successful and failure mission outcomes we use COUNT() function and we GROUP BY clause to group by mission outcome status.

```
%sql SELECT MISSION_OUTCOME, COUNT(*) AS "Total" from SPACEXTBL Group by MISSION_OUTCOME
```

```
 * ibm_db_sa://xzb03393:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.datab
ludb
Done.
```

mission_outcome	Total
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

To find Max Payload, we use a subquery, first, find the maximum value of the payload by using MAX() function, and second, filter the dataset to perform a search if PAYLOAD_MASS__KG_ is the maximum value of the payload. According to the result, version F9 B5 B10xx.x boosters could carried the maximum payload.

```
%sql select distinct(BOOSTER_VERSION), PAYLOAD_MASS__KG_  from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

```
 * ibm_db_sa://xzb03393:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1048.5	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1049.7	15600
F9 B5 B1051.3	15600
F9 B5 B1051.4	15600
F9 B5 B1051.6	15600
F9 B5 B1056.4	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600
F9 B5 B1060.3	15600

2017 Launch Records

To find list of successful landing_outcomes in ground pad ,booster versions, launch_site for the year 2017, we need to use EXTRACT clause to convert date to Year and filter landing outcome for successful landing for ground pad. Also we converted date to month as name.

```
%sql SELECT Year(Date) AS "Year", MONTHNAME(Date) AS "Month", Booster_Version,Launch_site, Landing__Outcome FROM SPACEXTBL where EXTRACT(YEAR FROM DATE)='2017'and Landing__Outcome = 'Success (ground pad)'
```

```
 * ibm_db_sa://xzb03393:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

Year	Month	booster_version	launch_site	landing__outcome
2017	February	F9 FT B1031.1	KSC LC-39A	Success (ground pad)
2017	January	F9 FT B1032.1	KSC LC-39A	Success (ground pad)
2017	March	F9 FT B1035.1	KSC LC-39A	Success (ground pad)
2017	August	F9 B4 B1039.1	KSC LC-39A	Success (ground pad)
2017	July	F9 B4 B1040.1	KSC LC-39A	Success (ground pad)
2017	December	F9 FT B1035.2	CCAFS SLC-40	Success (ground pad)

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

To find total rank for landing outcome we use WHERE clause, filter the dataset to perform a search if the date is between 2010-06-04 and 2017-03-20. Then we Using the ORDER BY clause to sort the records by total number of landing, and using DESC keyword to sort the records in descending order.

```
%sql select Landing__Outcome, COUNT(Landing__Outcome) AS "Total" from SPACEXTBL where (DATE between '2010-06-04' and '2017-03-20') group by Landing__Outcome Order by "Total" desc
```

```
 * ibm_db_sa://xzb03393:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

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

To find rank and count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order we use the same query but this time we added additional WHERE clause for Success.

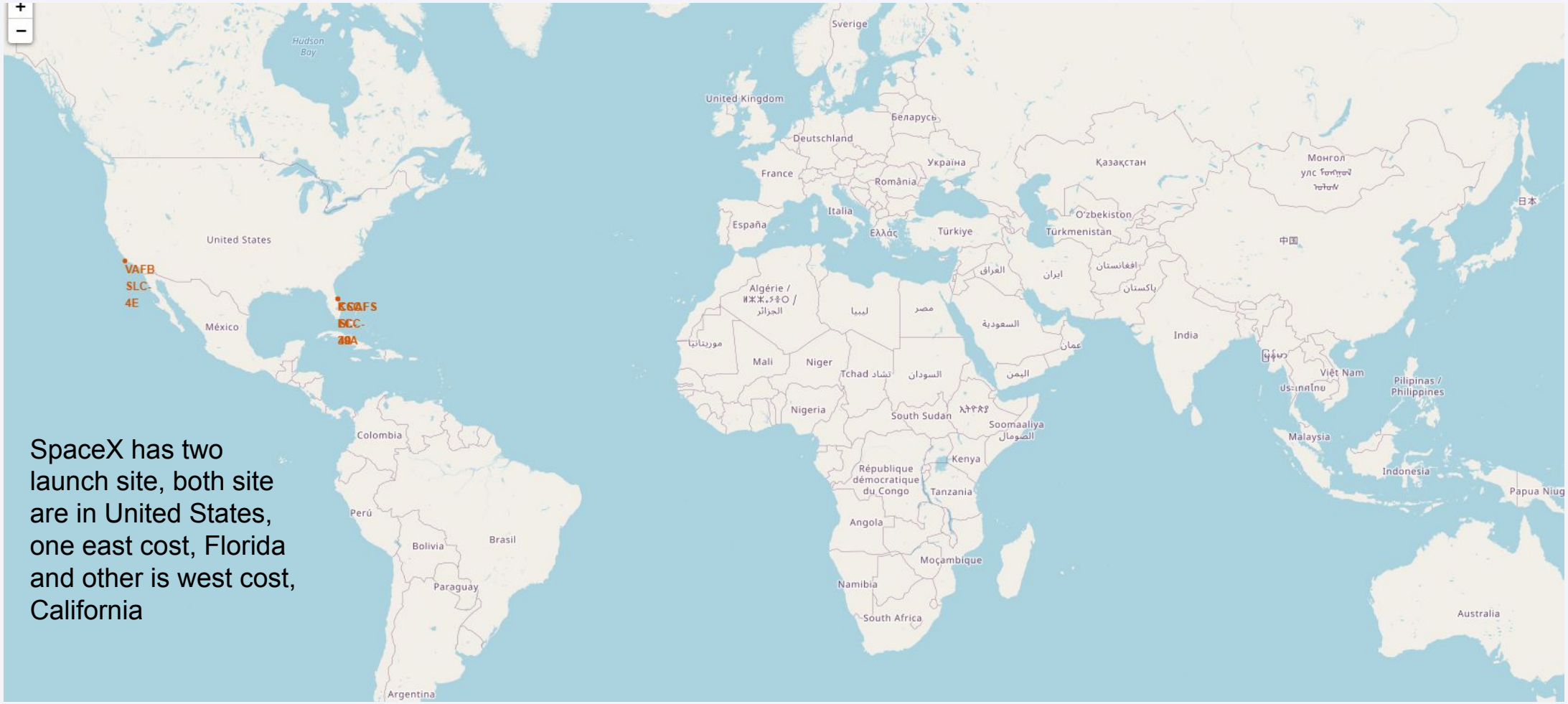
```
%sql select Landing__Outcome, COUNT(Landing__Outcome) AS "Total" from SPACEXTBL where Landing__Outcome like 'Success%' and (DATE between '2010-06-04' and '2017-03-20') group by Landing__Outcome
```

```
 * ibm_db_sa://xzb03393:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

landing__outcome	Total
Success (drone ship)	5
Success (ground pad)	5

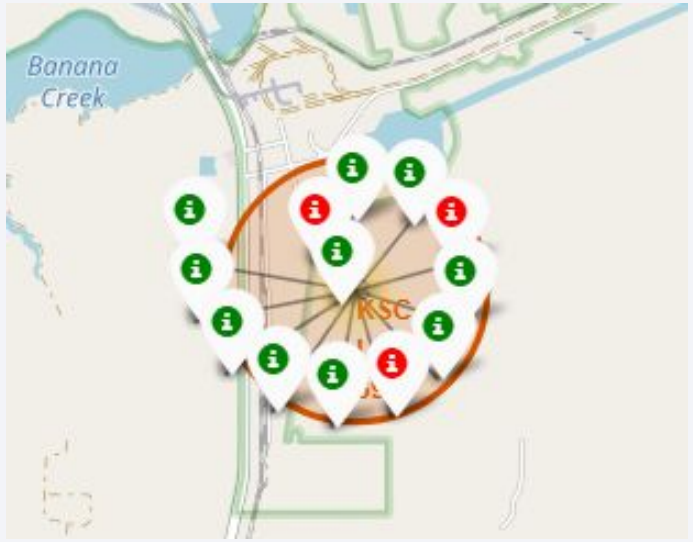

Section 3
Launch Sites
Proximities Analysis

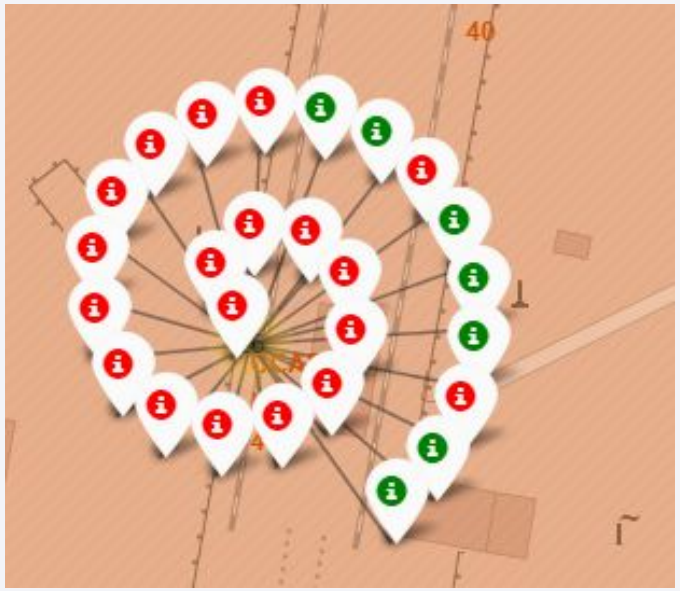
All Launch Site - Global Map



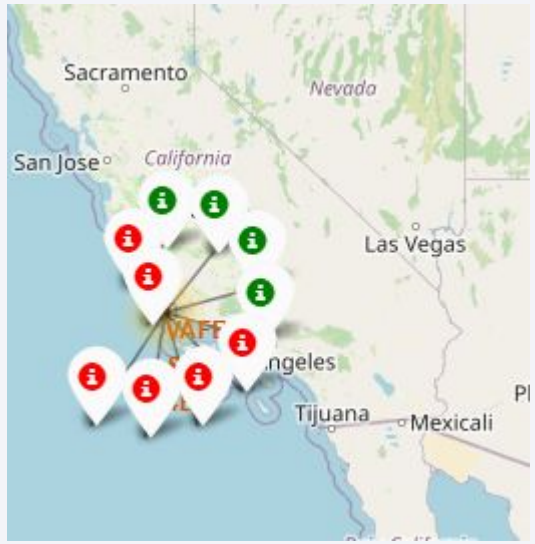
Markers showing launch site with color labels

Florida



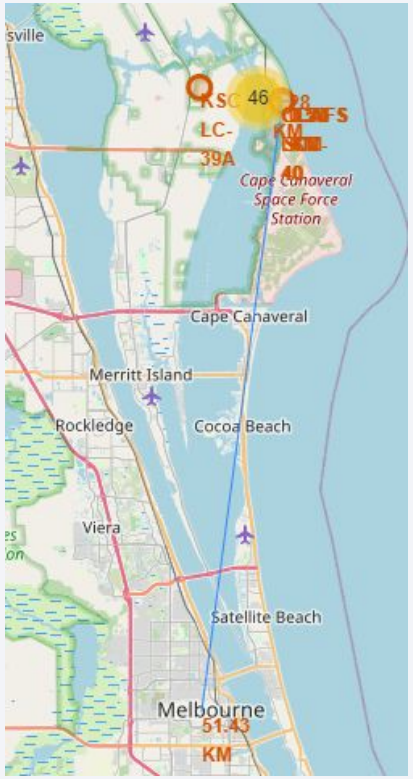


California

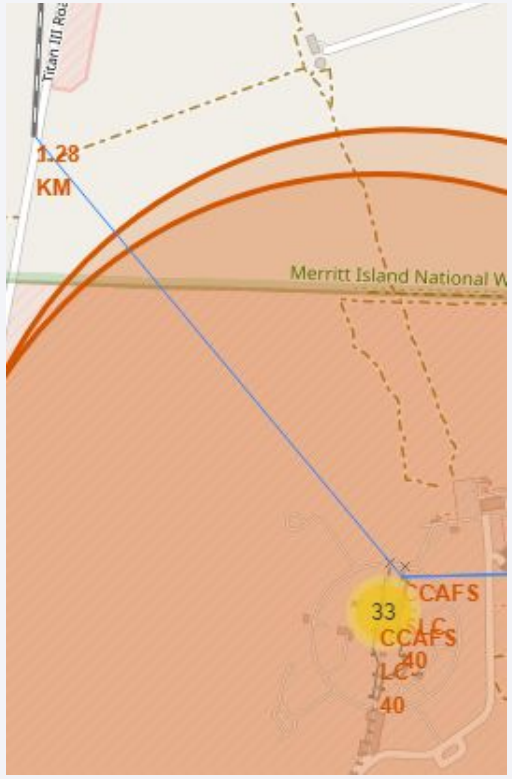


Distance from Launch Site to Landmarks

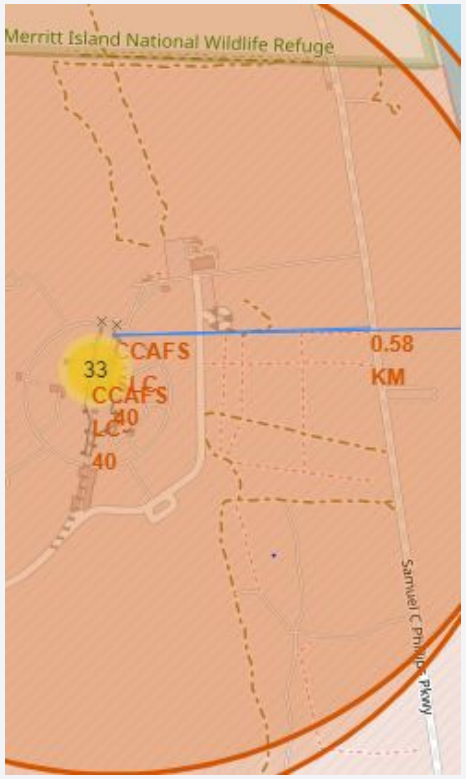
Closest City



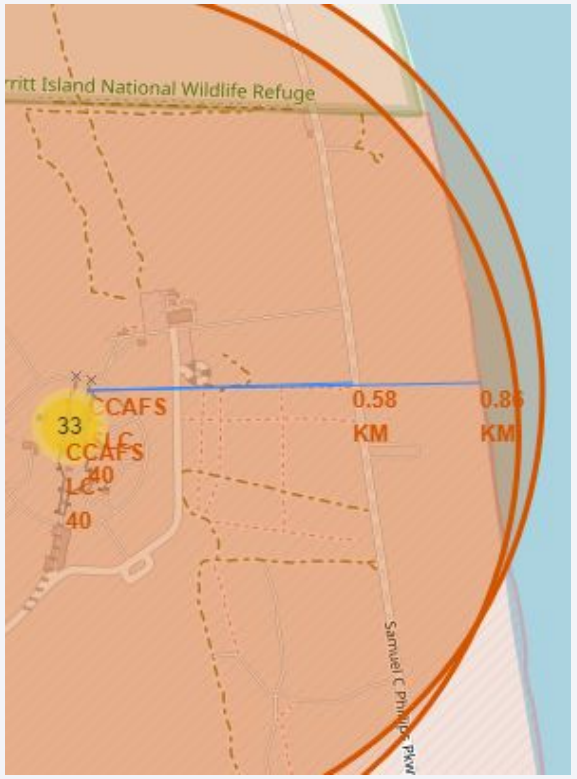
Closest Railway

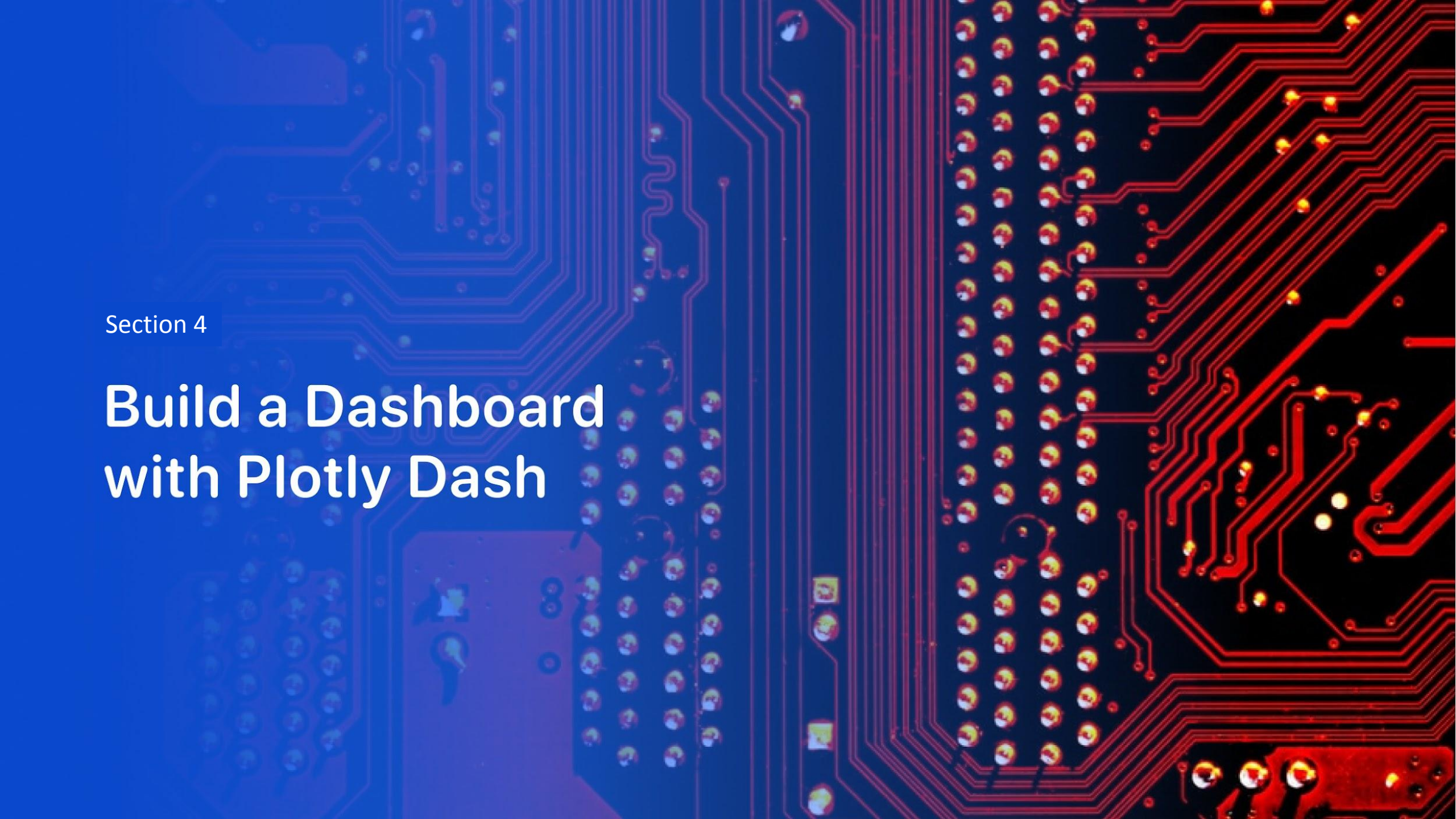


Closest Highway



Closest Coast

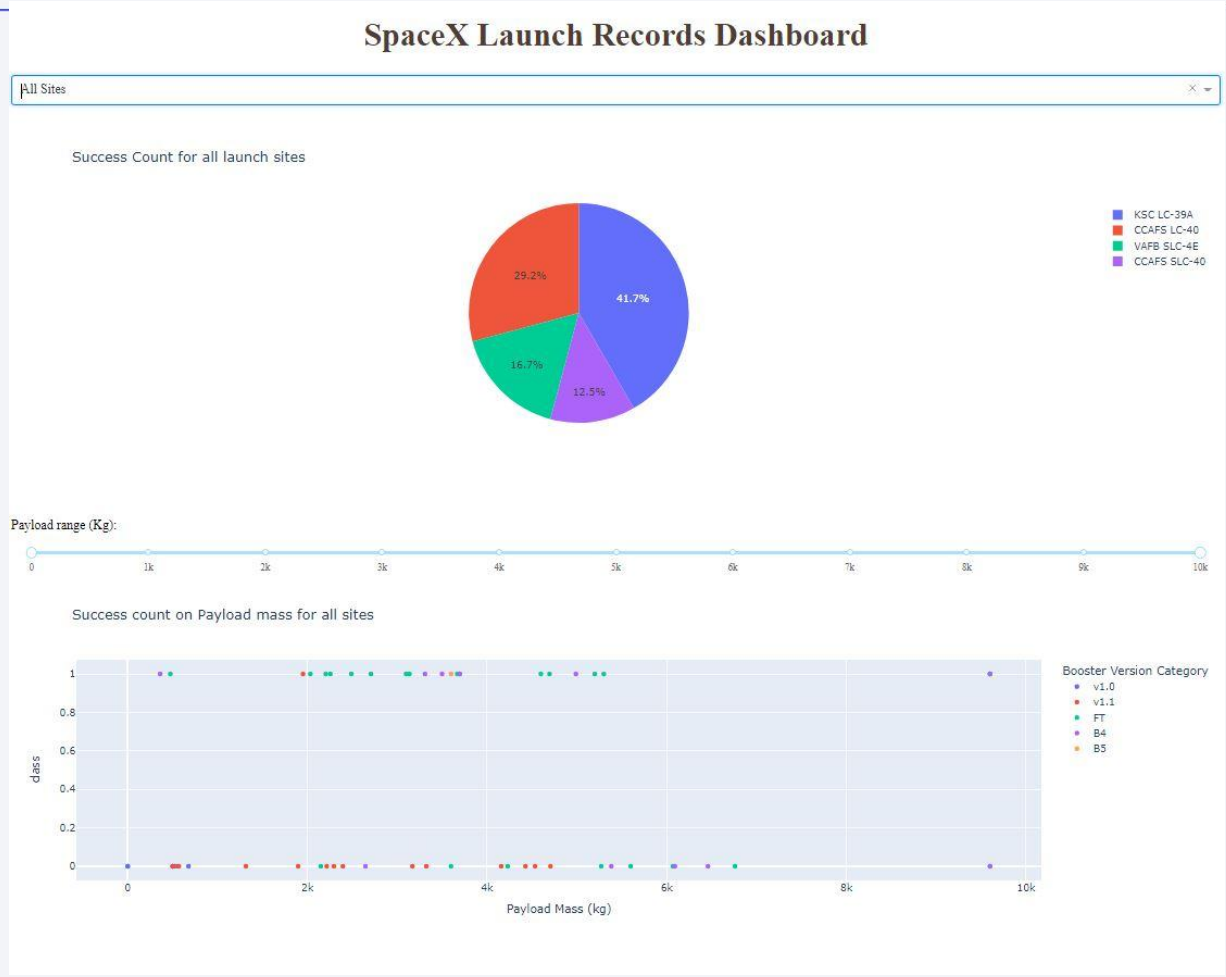



Section 4
Build a Dashboard with Plotly Dash

SpaceX Launch Records Dashboard

SpaceX Launch Dashboard contains following key elements.

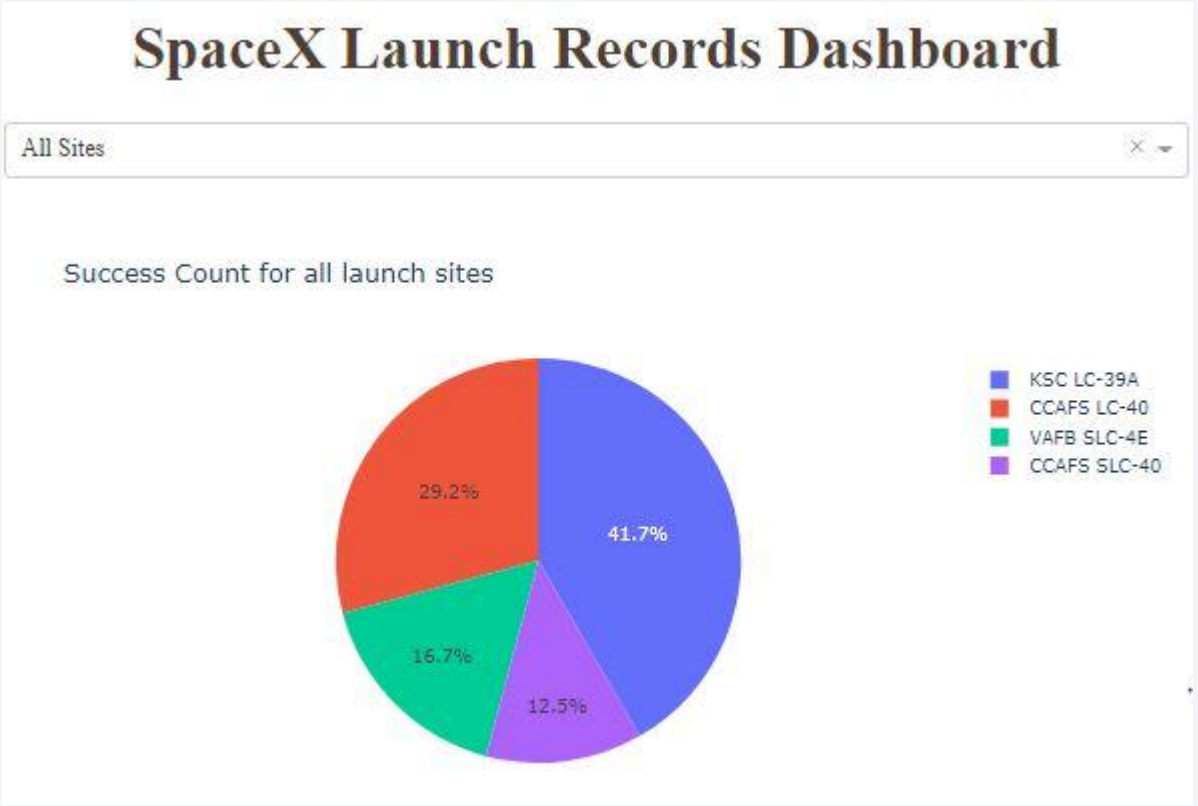
- Dropdown with all launch sites
- Success Rate Pie Chart
- Payload Range Selector
- Count of successful payload for all sites



Launch Sites

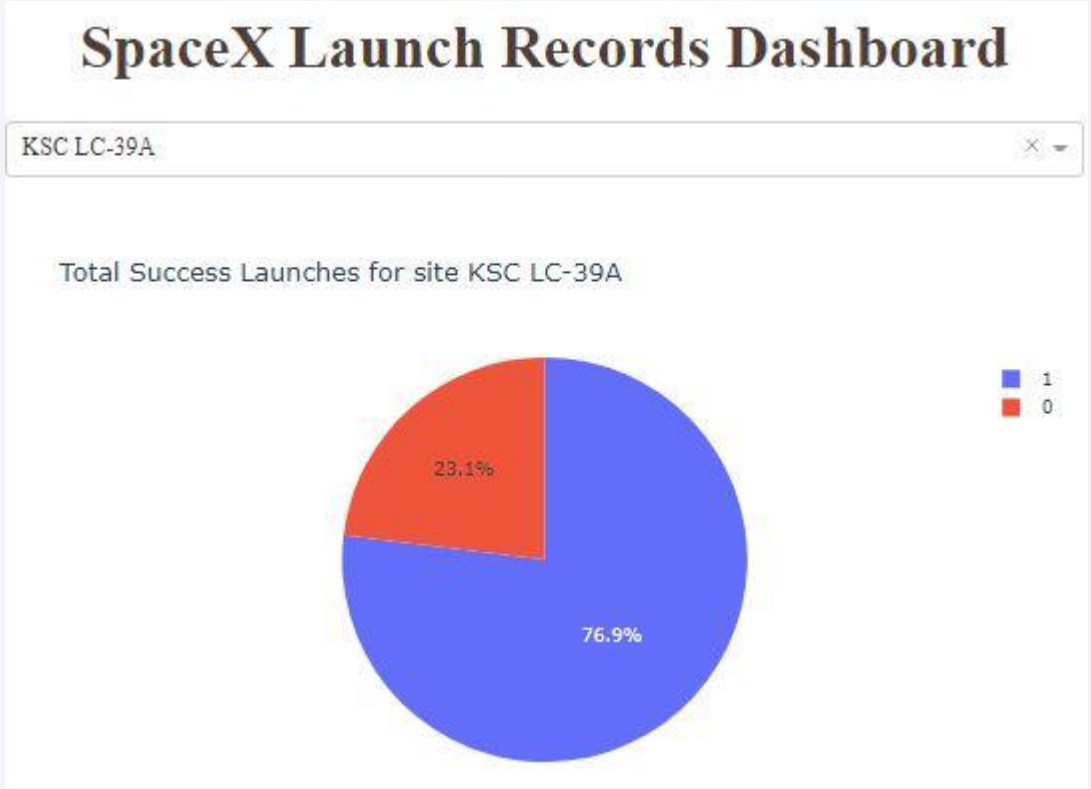
Success rate by all sites

- Top, the most successful launch site is KSC LC-39A
- Second,successful launch site is CCAFS LC-40
- Least successful launch site is CCAFS SLC-40



Highest Launch Success Ratio

- Based on data, highest launch success Ratio is by KSC LC-39A
- ~77% success rate

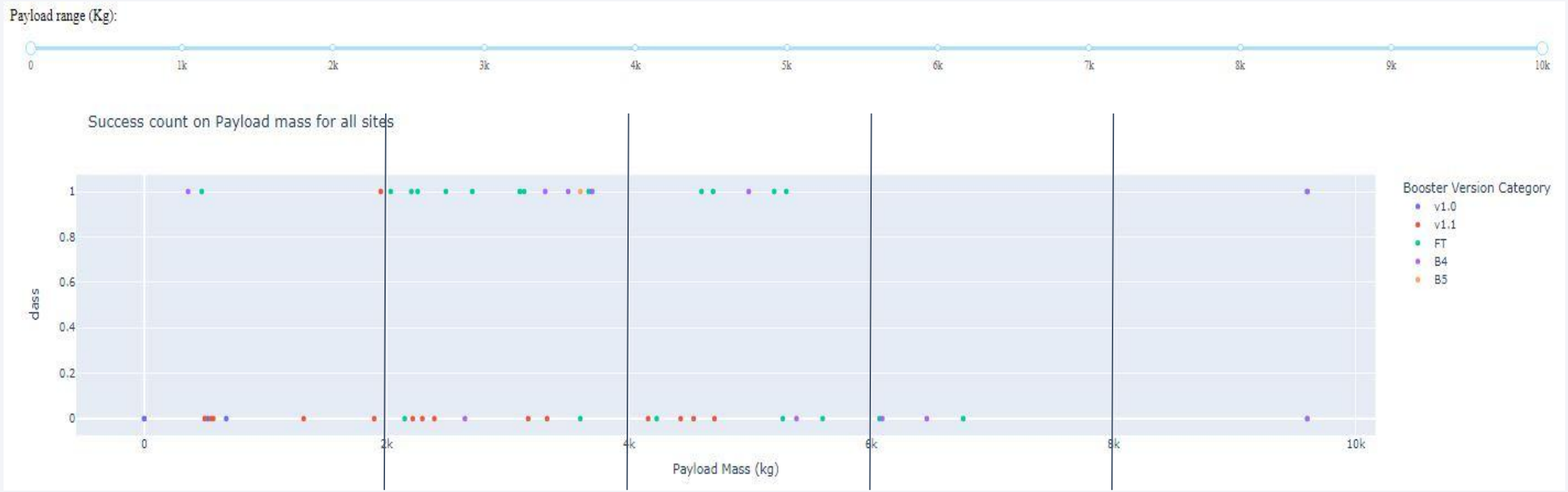


Payload vs Launch Outcome <=10K

Based on data representing payload vs outcome launch, we can see distinct difference in success rate between heavy and light payload. For example:

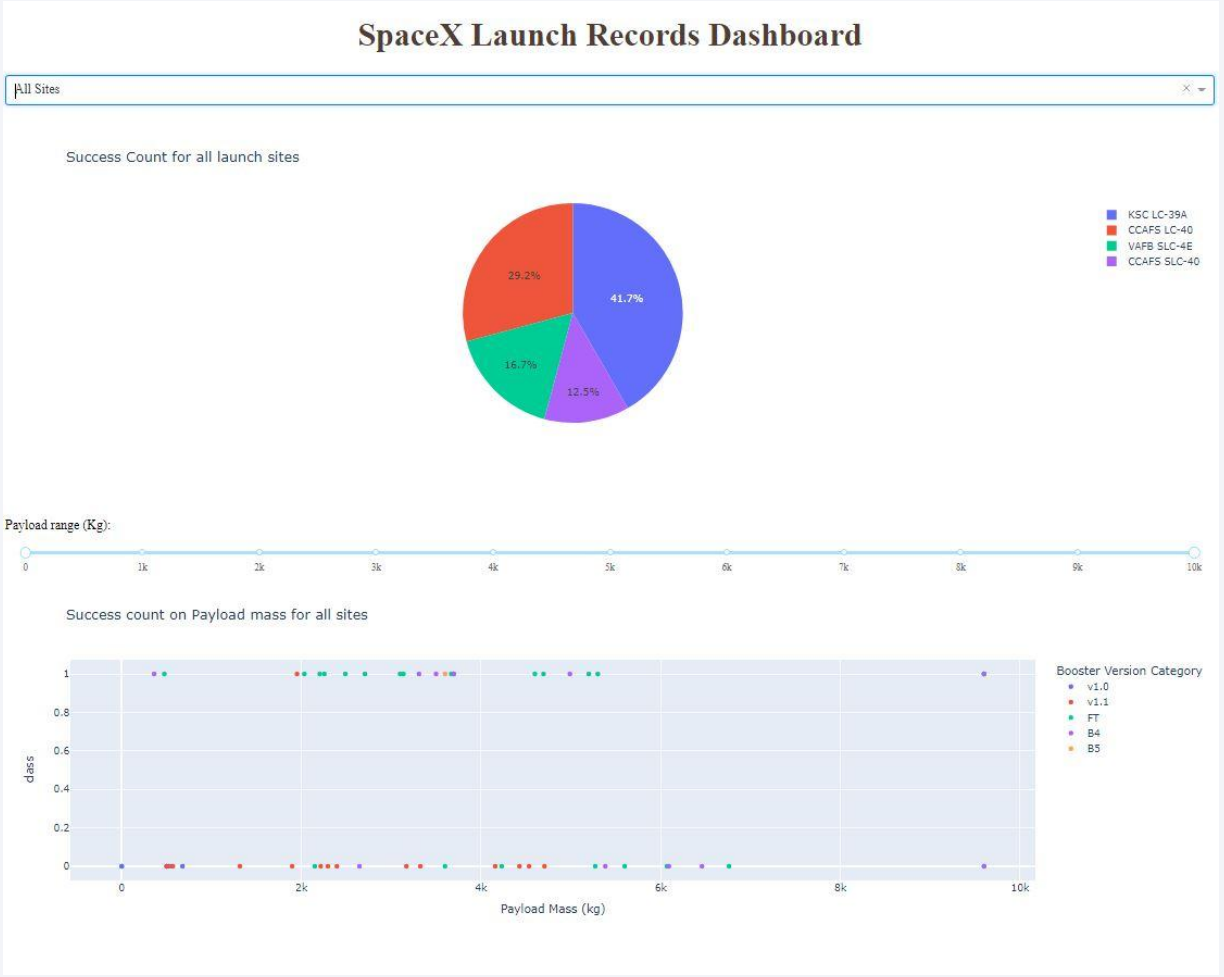
- 0 - 2K 38% success rate, not significant success rate but on right track
- 2K - 4K 60% success rate, the best success rate
- 4K-6K 38% success rate, not significant success rate but on right track
- 6K-8K 0% success rate, although there is few data points, this can be considered as insufficient
- 8K-10K 50% success rate, not enough data, it can be considered outliers

It appear that best success rate for the boost version is FT version.



SpaceX Dashboard Workbook

Dashboard (Code Only)
https://gist.github.com/idautovic/0bc8afcbd9823e58c591f7897d4c1a07



Section 5

Predictive Analysis (Classification)

Classification Accuracy

The decision tree classifier is the model with the highest classification accuracy

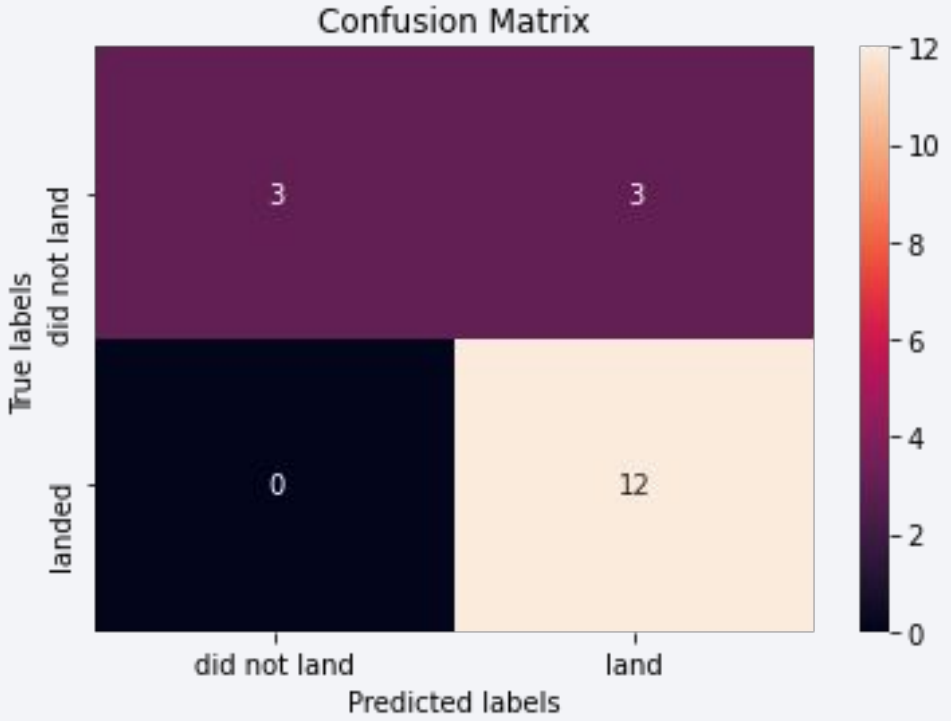
```
#scores = [lr_score,svm_score,tree_score,knn_score]
#print(scores)
#print(scores.index(max(scores)))
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8892857142857145
Best params is : {'criterion': 'gini', 'max_depth': 12, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_spl
it': 10, 'splitter': 'random'}
```

Confusion Matrix

The confusion matrix is the same for all models because all models performed the same for the test set.

Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the major problem is false positives.



Conclusions

Based on SpaceX data analysis we can conclude that.

- Success rate is correlated to the number of flights and time. As the number of flights increased, the success rate increased. This indicates that is constant improvement and eventually spacex will get to six sigma quality and ~99.99% success rate.
- Orbital types = SSO, HEO, GEO, ES-L1 = highest success rate (100%).
- KSLC-39A site has highest number of successful launch.
- Payload between 2K-6K under “FK” Booster performed the best.
- Tested model, all models have the same accuracy (83.33%).
- The launch site is close to railways, highways, and coastline, but far from cities.
- The Decision tree classifier is the best machine learning algorithm for analysis.

Data Sources

Data Collection API
spacex_url="https://api.spacexdata.com/v4/launches/past"

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

EDA Data Wrangling
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")

EDA with Data Visualization
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv")

EDA with SQL
https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_2/data/Spacex.csv?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDS0321ENSkillsNetwork26802033-2021-01-01

Machine Learning
data = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv")

Launch Sites Locations Analysis with Folium
wget.download('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv')

Launch Site Dashboard
wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_dash.csv"
wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_3/spacex_dash_app.py"

Workbooks

Data Collection API
https://gist.github.com/idautovic/f5beef6ebdb4af3d9280aa4902f556bf

Data Collection Web Scraping
https://gist.github.com/idautovic/6b7696d97396f03b8c51d686bafe6028

EDA Data Wrangling
https://gist.github.com/idautovic/49cacfc34619f8ae35b54e9dda1cbee6

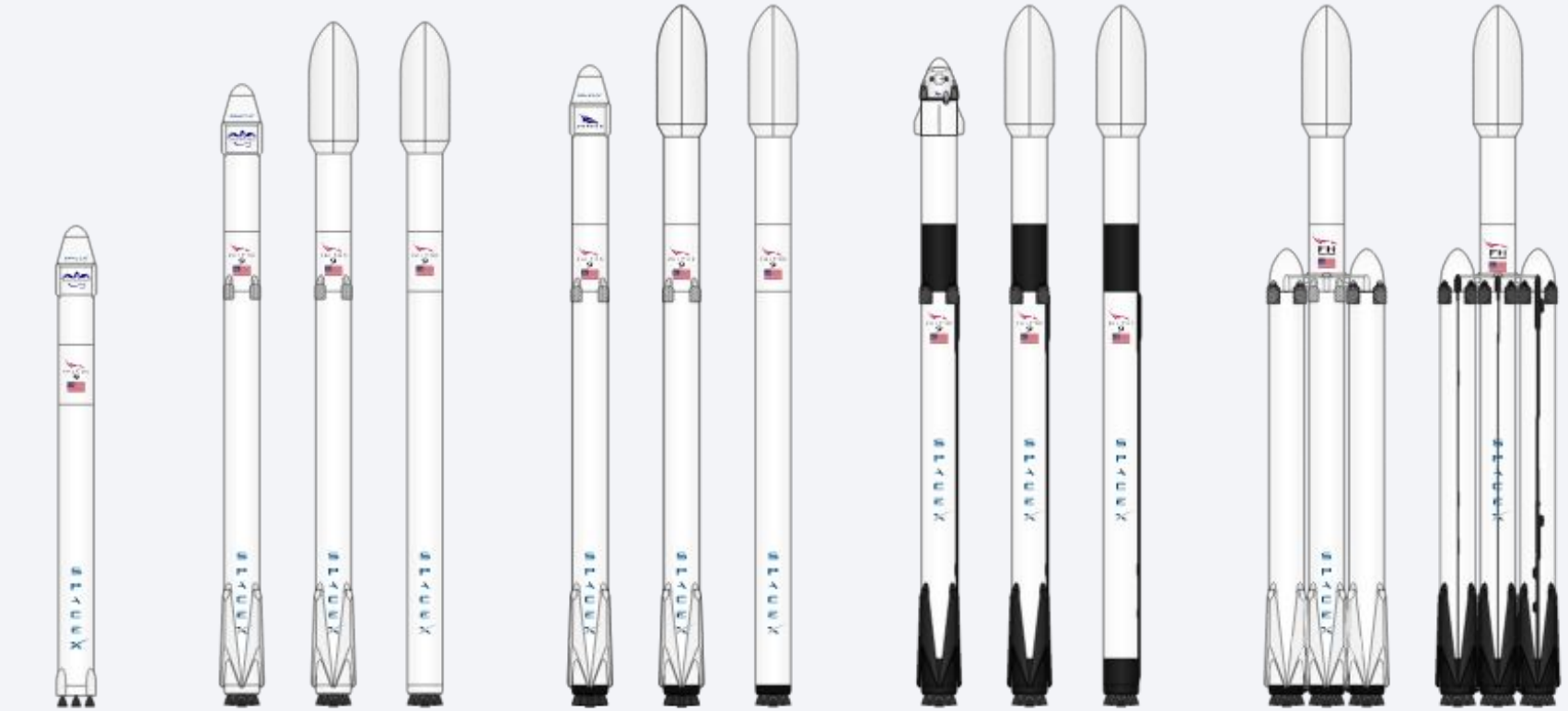
EDA with Data Visualization
https://gist.github.com/idautovic/285838d3366a8844621e5237ec2b2243

EDA with SQL
https://gist.github.com/idautovic/40a8f1cbc7e2a2c5af1a31efba734036

Machine Learning
https://gist.github.com/idautovic/2202a4c30dcb3863f71eaaf14160a434

SpaceX Dashboard (Code Only)
https://gist.github.com/idautovic/0bc8afcbd9823e58c591f7897d4c1a07

Appendix


Falcon 9 v1.0
Falcon 9 v1.1
Falcon 9 v1.2 (FT)
Falcon 9 Block 5
Falcon Heavy
FH B5

Thank you!