IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Alcides Neto
March 13, 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection (SpaceX API)
  - Data Collection (Web Scraping)
  - Data Wrangling
  - Exploratory Data Analysis (EDA) with Data Visualization
  - EDA with SQL
  - Building an Interactive Map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive Analysis (Classification)
- Summary of all results
  - Exploratory Data Analysis Results
  - Interactive Analytics Demo in Screenshots
  - Predictive Analysis Results

Introduction

- Project background and context
  - SpaceX 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 SpaceX 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 SpaceX for a rocket launch. The goal of this project is to predict if Falcon 9 first stage will land successfully by using a Machine Learning approach (model).
- Problems you want to find answers
  - What factors are behind the success of a landing?
  - What combination of features will most impact the success rate of a landing?
  - What conditions does SpaceX need to achieve to establish a successful landing program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - With SpaceX Rest API and Web Scrapping from Wikipedia
- Perform data wrangling
  - One-hot Encoding was applied to categorical data (features)
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Data sets were collected as shown bellow:
  - Collected rocket launch data with SpaceX API call;
  - Converted the response content to a Pandas data frame with help of JSON;
  - Filtered the data to include only Falcon 9 launches;
  - Checked for missing values and replaced them where necessary;
  - Falcon 9 launch records were collected by web scraping Wikipedia using BeautifulSoup;
  - The launch HTML tables were parsed into a Pandas data frame.

Data Collection – SpaceX API

GitHub URL to notebook

1. Collect data with API call

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

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

2. Convert do Pandas data frame

```
# Use json_normalize meethod to convert the json result into a dataframe
response = requests.get(static_json_url)
if response.status_code == 200:
    data = pd.json_normalize(response.json())
```

3. Data cleaning (pre-processing)

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

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

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

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

4. Filter data

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data_launch[data_launch['BoosterVersion']=='Falcon 9']
```

5. Fill missing values

```
# Calculate the mean value of PayloadMass column
payloadmass_mean = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].fillna(payloadmass_mean, inplace=True)
```

Data Collection – Web Scraping

[GitHub URL to notebook]

1. Creating the BeautifulSoup object

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
if response.status_code == 200:
    soup = BeautifulSoup(response.text, 'html.parser')
```

2. Finding tables

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

3. Getting column names

```
column_names = []
```

```
rows = first_launch_table.find_all('th')
for row in rows:
    print(row)
    header = extract_column_from_header(row)
    if header != None:
        if len(header) > 0:
            column_names.append(header)
```

4. Creating dictionary from 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
            # 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)
```

5. Converting dictionary to Pandas data frame

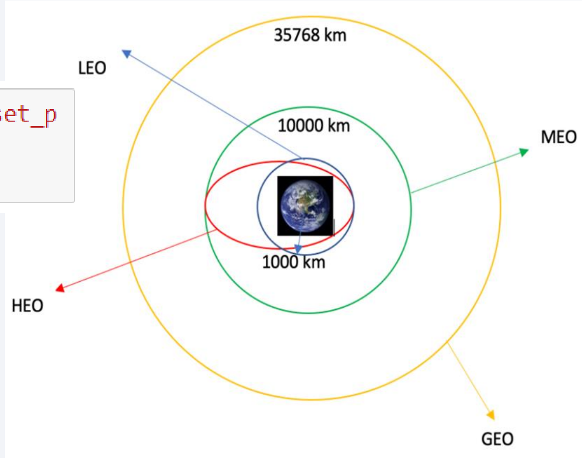
```
df=pd.DataFrame(launch_dict)
df.head()
```

Data Wrangling

GitHub URL to notebook

1. Loading data set

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```



2. Calculate the number and occurrence of mission outcome per orbit type

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

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

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

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

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

4. Create a landing outcome label from Outcome column

```
landing_class = df['Outcome'].map(lambda x: 0 if x in bad_outcomes else 1)
landing_class.value_counts()
```

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

5. Determining the mean success rate

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

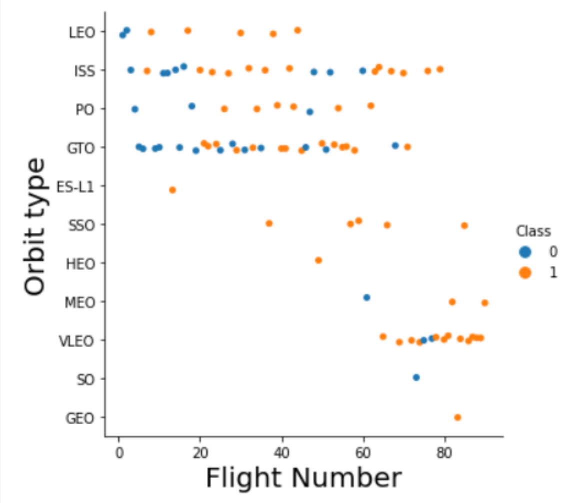
```
0.6666666666666666
```

EDA with Data Visualization

[GitHub URL to notebook]

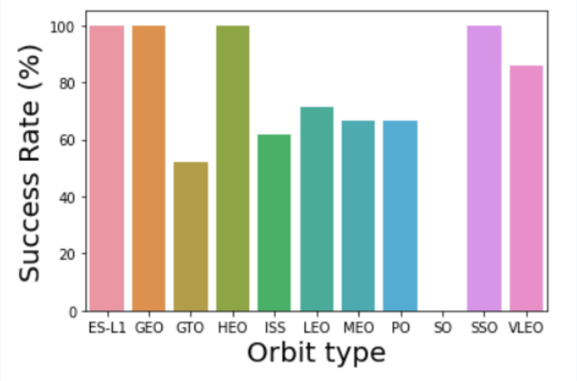
1. Scatter Plots

Scatter Plots show how the relationship between two variables (correlation).



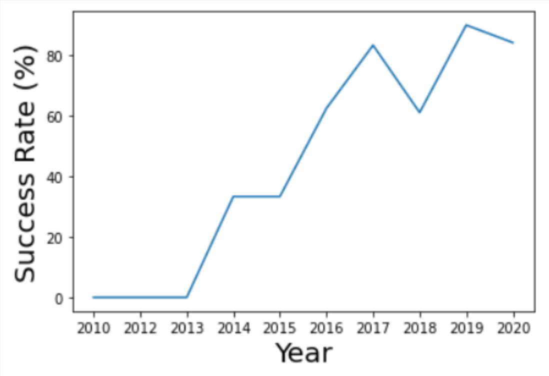
2. Bar Graph

A bar diagram make easier to compare sets of data between different groups.



3. Line Plot

Show data variable and trends in a clear way. Can help to making predictions.



EDA with SQL

GitHub URL to notebook

- We applied EDA with SQL to get insight from the data. Queries written:
  - Display the names of 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 average payload mass carried by booster version F9 v1.1;
  - List the date when the first successful landing outcome in ground pad was achieved;
  - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000;
  - List the total number of successful and failure mission outcomes;
  - List the names of the booster_versions which have carried the maximum payload mass;
  - List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015;
  - 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.

Build an Interactive Map with Folium [GitHub URL to notebook]

- To visualize the launch data into an interactive map, we marked all launch sites using latitude and longitude coordinates, and added map objects such as markers, circles, and labels to identify each site on the folium map.
- We assigned the data frame column launch_outcomes to class 0 (failure) and 1 (success).
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- Using Haversine's formula we calculated the distances between a launch site to various landmarks. We answered the following questions:
  - Are launch sites near railways, highways and coastlines?
  - Do launch sites keep certain distance away from cities?
- Lines are drawn on the map to support the answers.

Build a Dashboard with Plotly Dash

GitHub URL to code

- An interactive dashboard with Plotly Dash was built
- We plotted pie charts showing the total launches by a certain sites or by all sites, displaying relative proportions of multiple classes of data.
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version, revealing the relationship between both variables.

Predictive Analysis (Classification)

[GitHub URL to notebook]

1. Building the model
    - Creating column for the Class (what we want to predict)
    - Data standardization
    - Splitting data into train/test sets
    - Building GridSearchCV model and fit the data
2. Evaluating the model
    - Calculating accuracies and confusion matrix
    - Plotting the results
3. Finding the best model
    - Find the best hyperparameters for the models
    - Find the model with best accuracy

Results

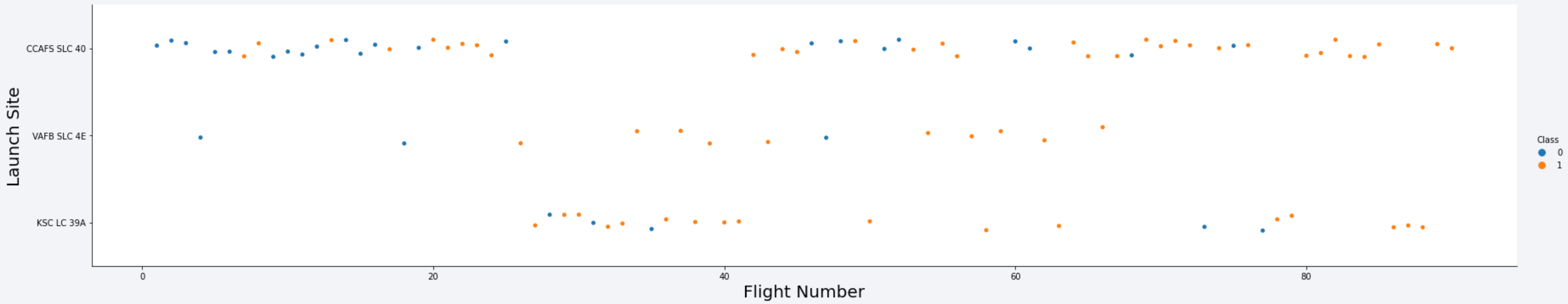
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

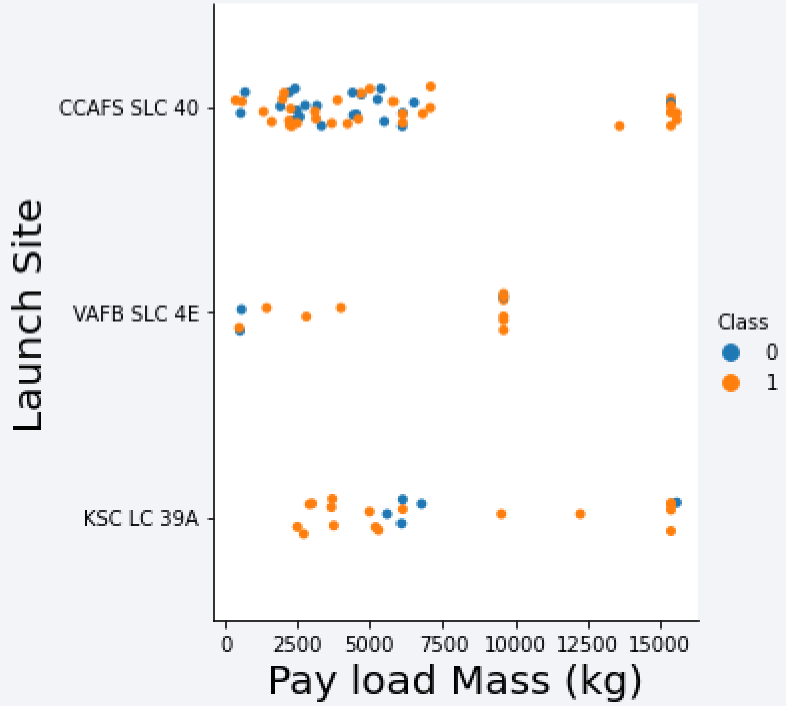
Flight Number vs. Launch Site

- With the increase of flight amount at a launch site, the success rate is increasing as well at that launch site.



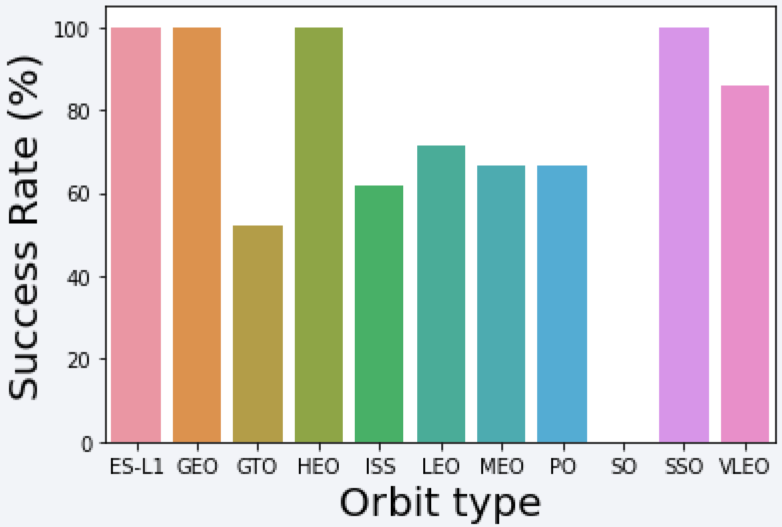
Payload vs. Launch Site

- The greater the payload mass for Launch Site CCAFS SLC40 the higher the success rate for the rocket.



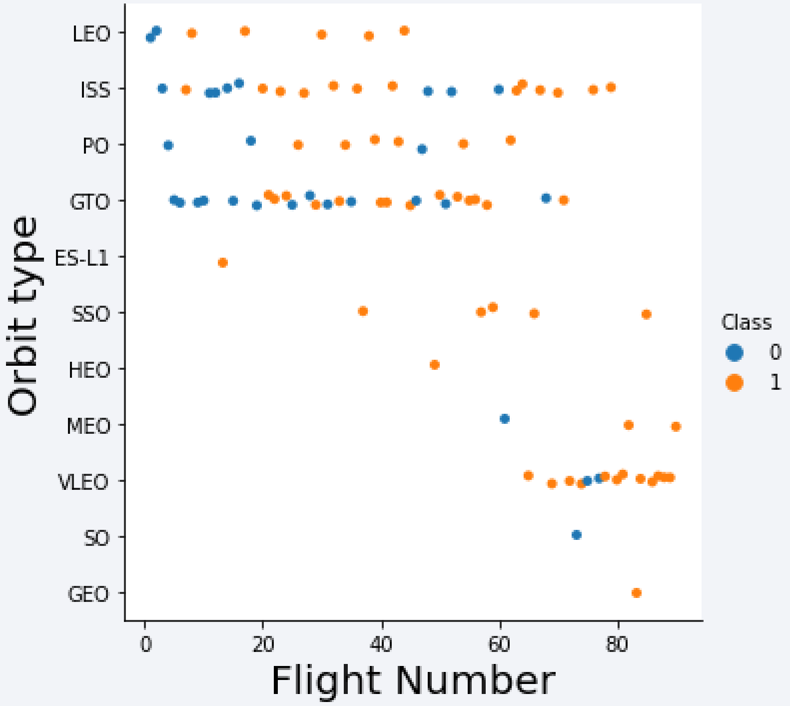
Success Rate vs. Orbit Type

- Orbits ES-L1, GEO , HEO and SSO had the best success rate.



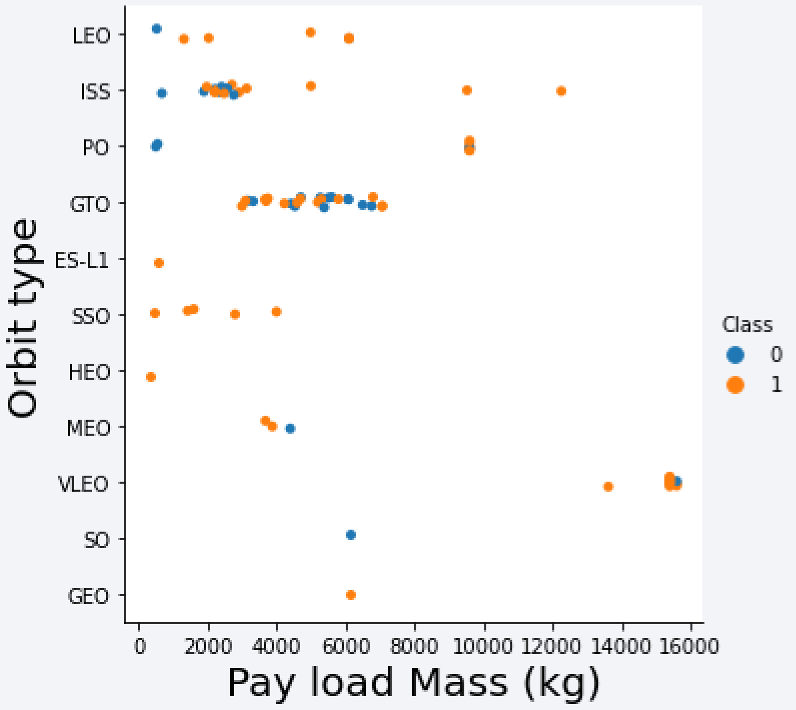
Flight Number vs. Orbit Type

- LEO orbit the success appears related to the number of flights;
- There seems to be no relationship between flight number when in GTO orbit.



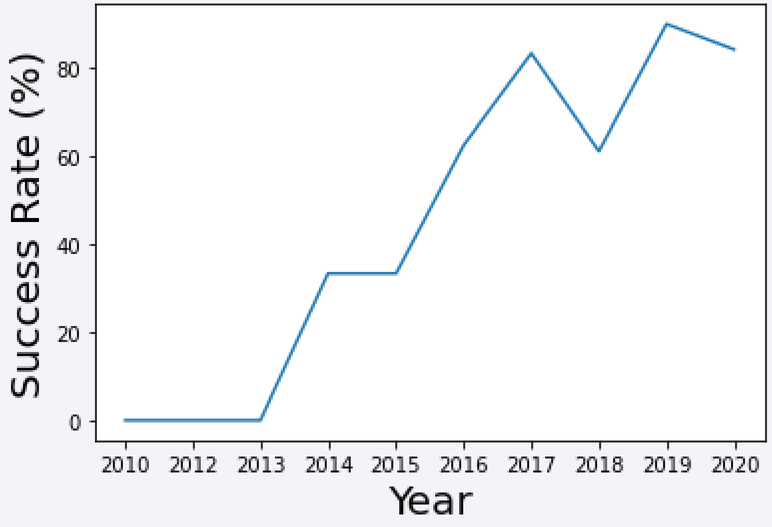
Payload vs. Orbit Type

- Heavy payloads have a negative influence on GTO orbits and positive effect on PO, ISS and LEO orbits.



Launch Success Yearly Trend

- You can observe that the success rate since 2013 kept increasing till 2020.



All Launch Site Names

- We can get unique values using key word “DISTINCT”.

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

```
 * ibm_db_sa://qwr42669:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

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

Launch Site Names Begin with 'CCA'

- The key word LIMIT in the query limits the results to only 5 records.

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

```
 * ibm_db_sa://qwr42669:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- We can get the sum of all values by using the key word SUM.

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

```
 * ibm_db_sa://qwr42669:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

1
45596

Average Payload Mass by F9 v1.1

- We can get the average of all values by using the key word AVG and the WHERE clause.

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

```
 * ibm_db_sa://qwr42669:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

1
2928

First Successful Ground Landing Date

- We can get the first successful record (launch) by using the key word MIN. In this case, the first date is the minimum date.

```
%%sql
select min(DATE)
from SPACEXTBL
where LANDING__OUTCOME = 'Success (ground pad)'
```

```
 * ibm_db_sa://qwr42669:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We can filter the landing outcome and payload mass using the key word WHERE. To get results with payload mass between two values (4000 and 6000) we used the key word BETWEEN.

```
%%sql
select distinct BOOSTER_VERSION
from SPACEXTBL
where (LANDING__OUTCOME = 'Success (drone ship)') and (PAYLOAD_MASS__KG_ between 4000 and 6000)
```

```
 * ibm_db_sa://qwr42669:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

- We used wildcard LIKE (%) to filter using the key word WHERE to determine if a mission outcome was a success (100 launches) or a failure (1 launch).

```
%%sql
select MISSION_OUTCOME, COUNT(MISSION_OUTCOME)
from SPACEXTBL
where MISSION_OUTCOME like 'Success%' or MISSION_OUTCOME like 'Failure%'
group by MISSION_OUTCOME
```

```
 * ibm_db_sa://qwr42669:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

mission_outcome	**2**
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the key word MAX.

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

```
 * ibm_db_sa://qwr42669:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

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

2015 Launch Records

- We can get the year of a record by using year(DATE) in the WHERE clause.

```
%%sql
select LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
from SPACEXTBL
where LANDING__OUTCOME = 'Failure (drone ship)' and year(DATE) = 2015
```

```
 * ibm_db_sa://qwr42669:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- With the key word ORDER BY DESC we can order values in descending order and with key words COUNT + GROUP BY we can count all records from each landing outcome.

```
%%sql
select LANDING__OUTCOME, count(LANDING__OUTCOME)
from SPACEXTBL
where (DATE between '2010-06-04' and '2017-03-20')
group by LANDING__OUTCOME
order by count(LANDING__OUTCOME) desc
```

```
 * ibm_db_sa://qwr42669:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

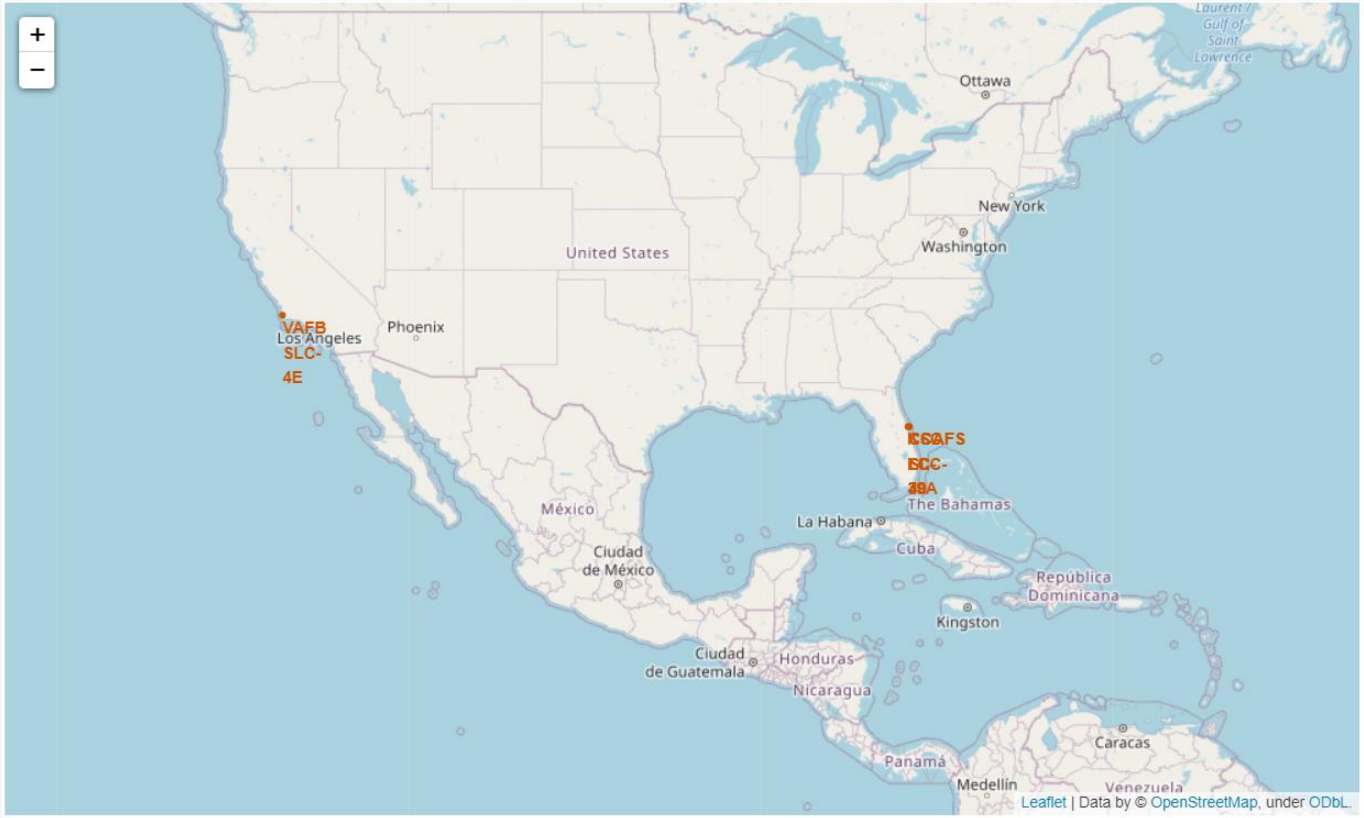
landing__outcome	2
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

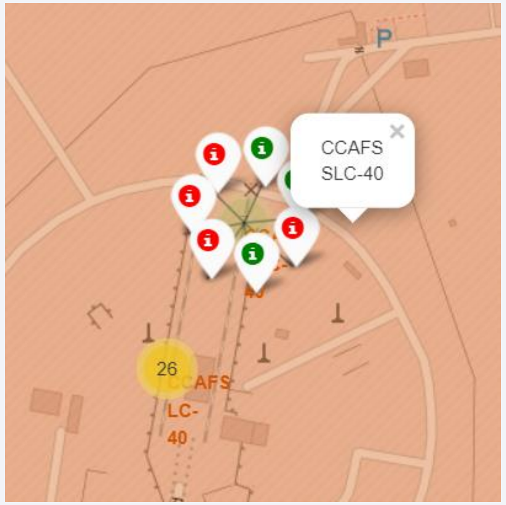
All launch sites global map markers

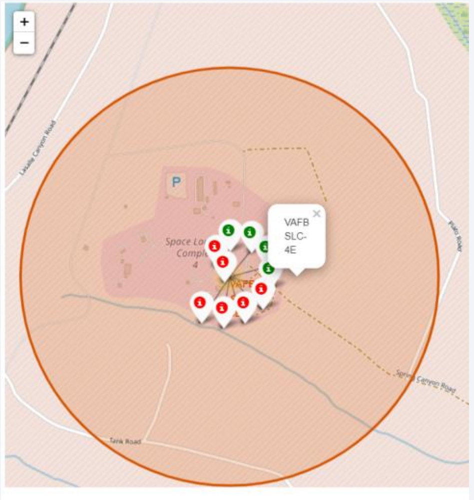
- All launch sites are in the USA coasts (Florida and California).



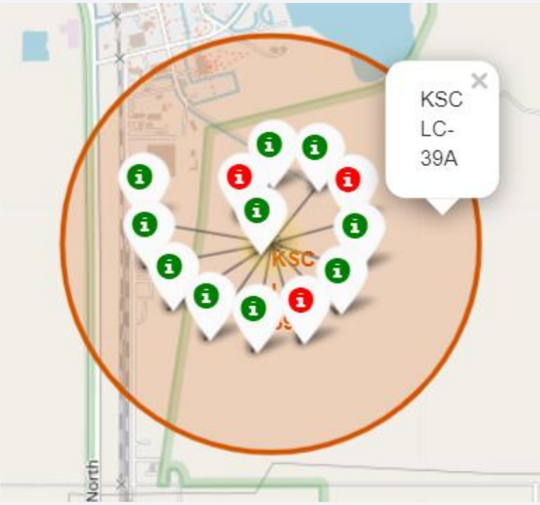
Markers showing launch sites with color labels

- Florida launch sites



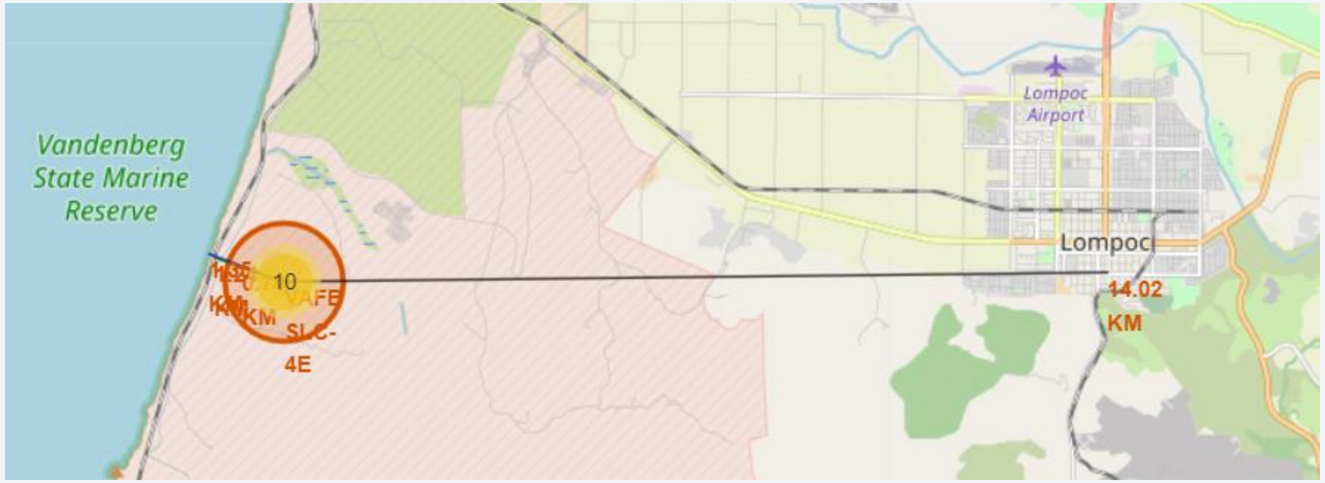


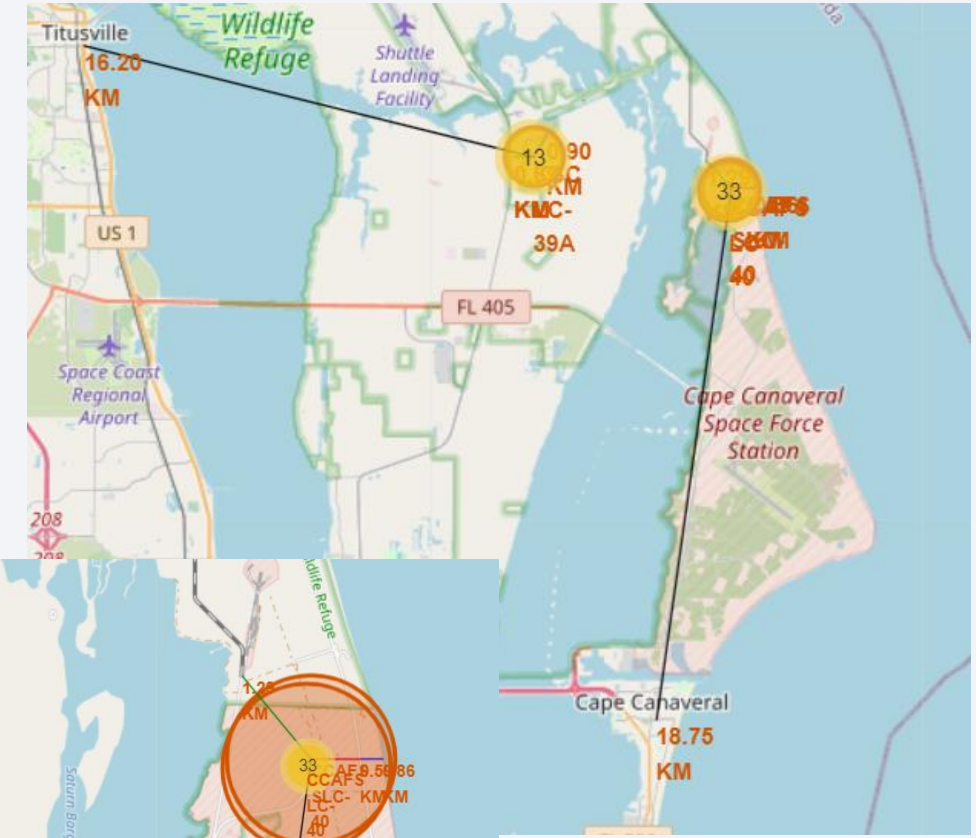
- California launch site



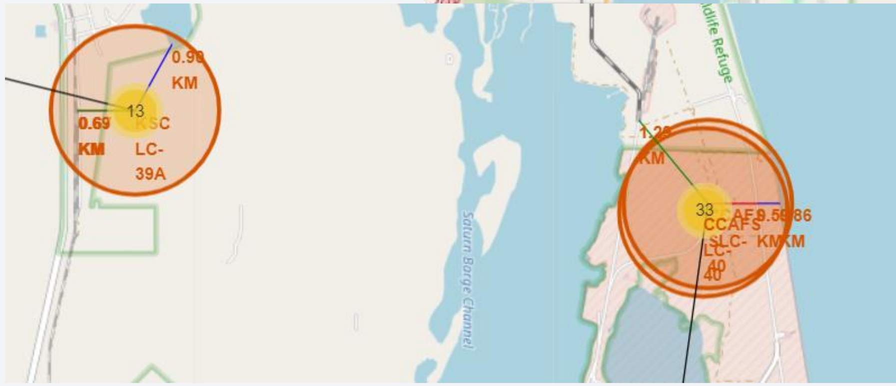
- Green marker: successful landing
- Red marker: failure landing

Launch Site distance to landmarks





- All distances from launch sites to its proximities. They weren’t far away from coastline, railways nor highways; but all sites are far away from cities.

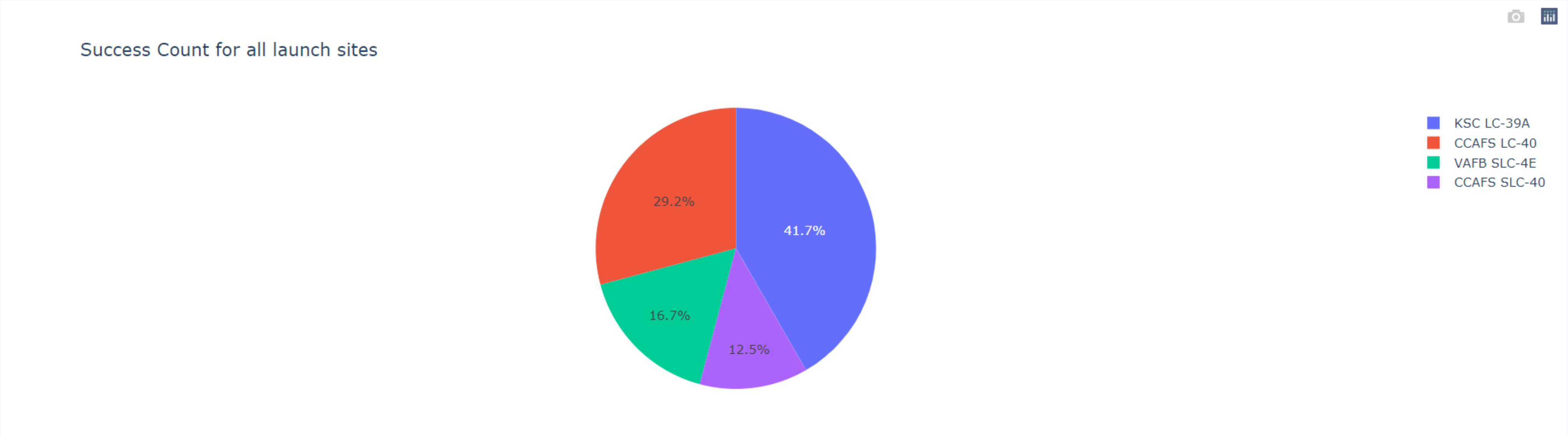


Section 4

Build a Dashboard with Plotly Dash

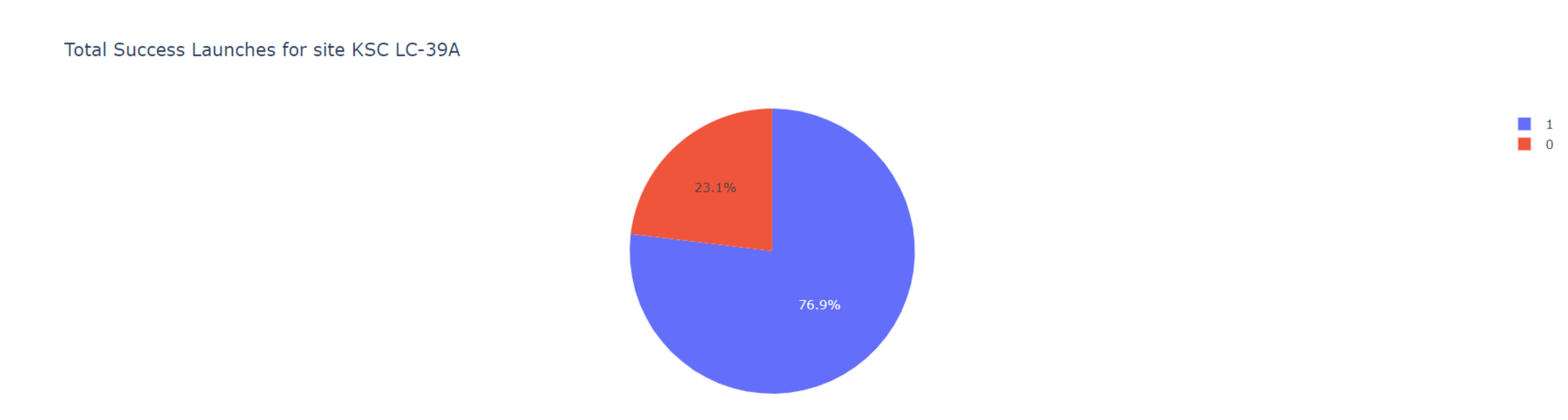
Pie chart showing the success percentage achieved by each launch site

- KSC LC-39A has the highest success score.



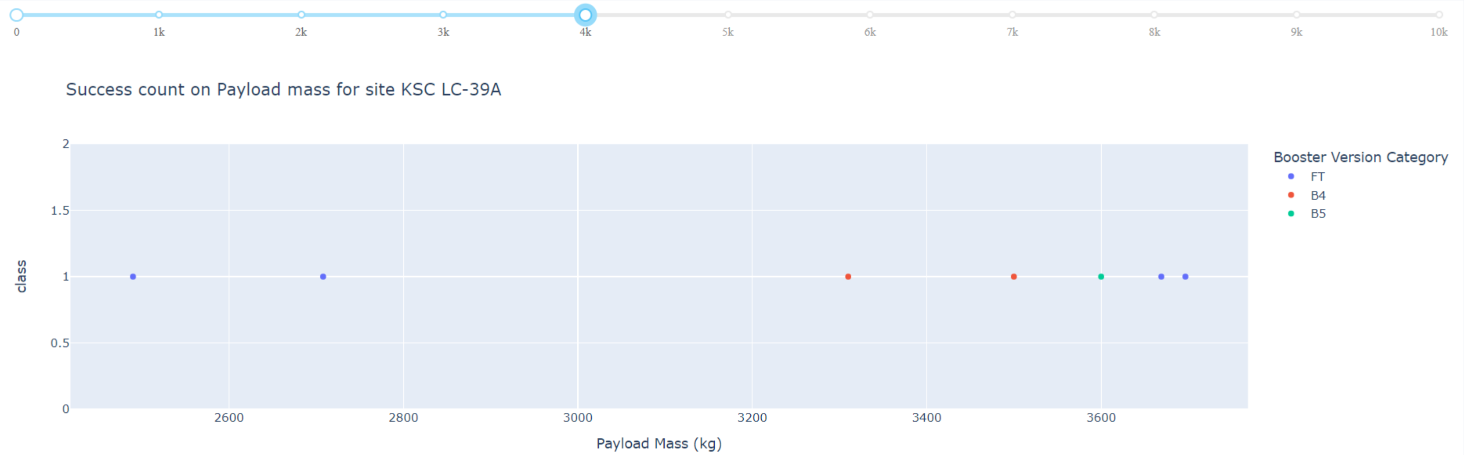
Pie chart showing the Launch site with the highest launch success ratio

- KSC LC-39A achieved a success rate of almost 77%.

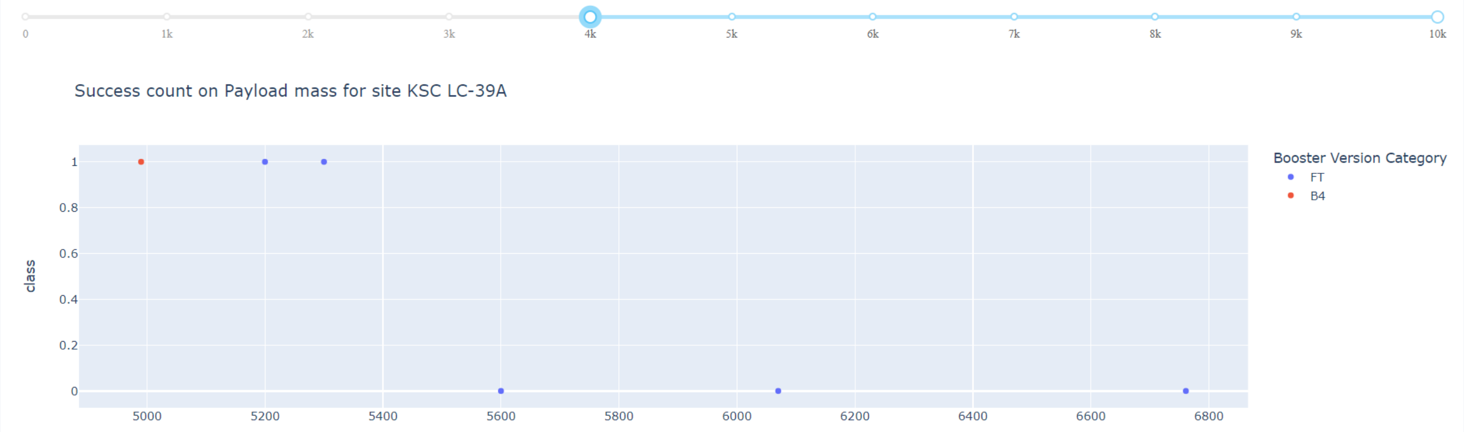


Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

- Low weighted payload (0 to 4000 kg):



- Heavy weighted payload (4000 to 10000 kg):



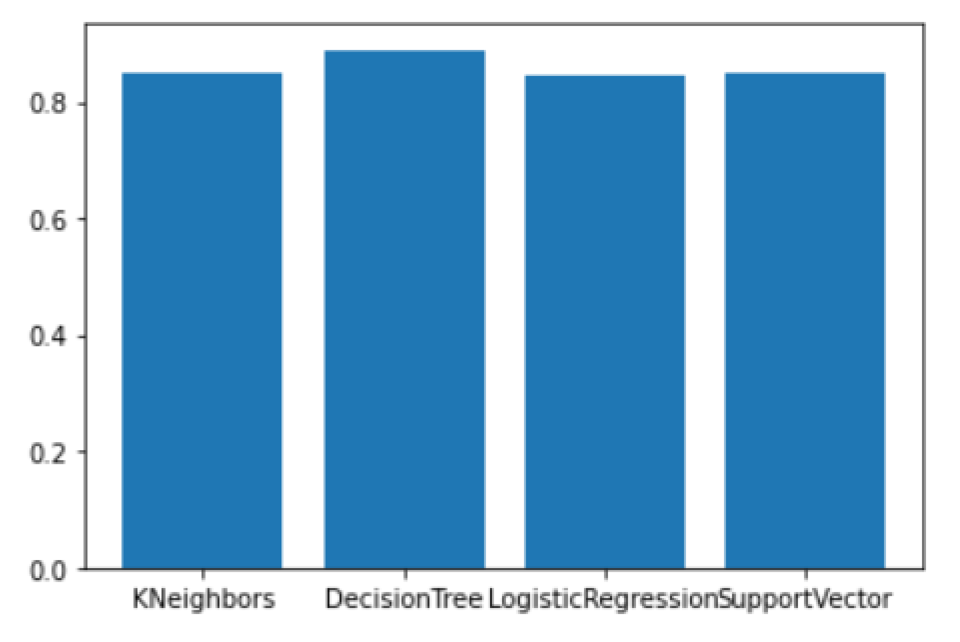
- The success rate for low weighted payloads is higher.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

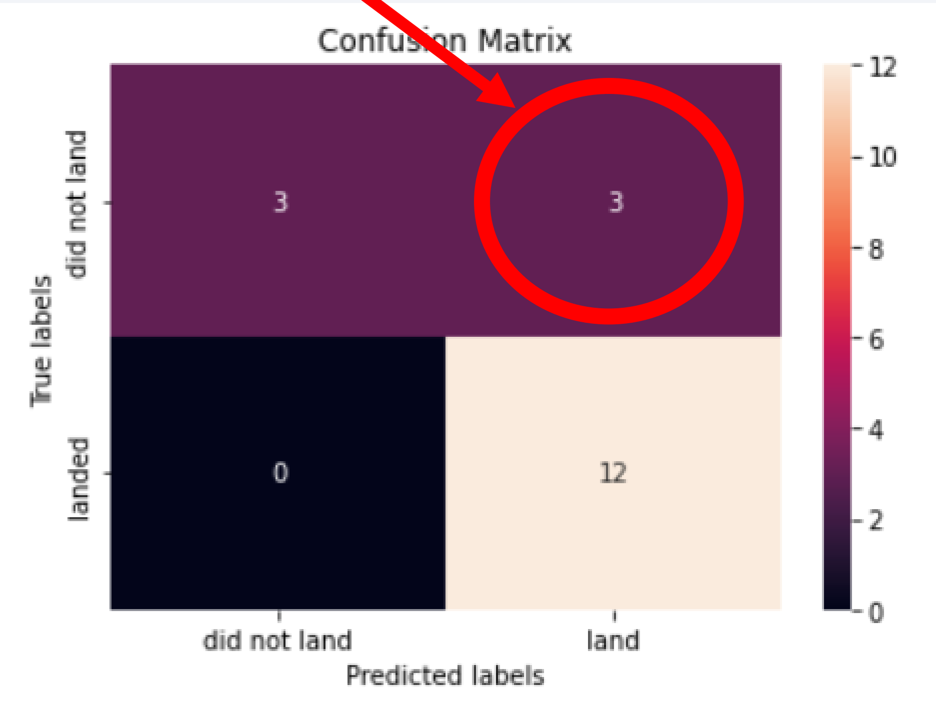
- Decision Tree has the highest accuracy with almost 0.89.




```
Best model is DecisionTree with a score of 0.8892857142857142
Best parameters are: {'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
```

Confusion Matrix

- The confusion matrix shows that the classifier can distinguish between the different classes. The major problem is the false positives (unsuccessful landing marked as successful landing by the decision tree classifier).



Conclusions

- We found that KSC LC-39A is the launch site with the highest success rate;
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate;
- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020;
- Decision Tree was the optimal model with accuracy of almost 0.89;

Appendix

- All codes can be found in my GitHub repo: link

Thank you!