IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Olea Roy
21st February, 2023

Outline

EXECUTIVE SUMMARY

INTRODUCTION

METHODOLOGY

RESULTS

CONCLUSION

APPENDIX

Executive Summary

Summary of Methodology

- The data is collected through the SpaceX API and through webscraping a wikipedia page on Falcon 9 launches.
- The collected data is wrangled to arrive at the 'label' which will be used to train ML models to predict launch outcomes.
- Exploratory data analysis through data visualization and SQL is used to find patterns in data.
- An interactive map is build in Folium and a dashboard on Plotly Dash to perform interactive analytics.
- Finally, the predictive analysis involves finding the best model for predicting launch outcome. As such, we train a Logistic Regression, a SVM, a Decision Tree, and a K Nearest Neighbour model.

Summary of Results

- The exploratory data analysis through data visualization and SQL provide insights on the relationships between the different variables, and these are explained in the Results section of this presentation.
- The interactive dashboard enables real time data visualization and comparison.
- The four ML models trained to predict launch outcome all have the same accuracy rate of 83.33%.

Project Background and Context

Introduction

SpaceX is a private American space exploration company whose aim is to reduce space transportation costs. They have made notable achievements in the production of reusable spacecrafts.

SpaceX advertises their Falcon 9 rocket launches at 62 million dollars, while with their competitors, these cost upward of 165 million dollars. SpaceX has reduced its costs thanks to the reusable first stage booster.

SpaceY wants to bid against SpaceX and hence, interested in knowing the cost of a launch by SpaceX. Given that the first stage is reusable, whether or not it lands successfully can influence the costs greatly. Thus, this project has the following aim:

To be able to predict whether the first stage of SpaceX Falcon 9 rockets will be landing successfully.

Section 1

Methodology

The methods used to achieve the aim of the project are:

1. Data Collection through SpaceX API and Web Scraping
2. Data Wrangling to make drawing meaningful insights easier
3. Exploratory Data Analysis through Data Visualization and SQL
4. Interactive Visual Analytics using Folium and Plotly Dash
5. Predictive Analysis with Machine Learning Classification Models

1. Data Collection

- **The data is collected in two ways:**

  - **Using the SpaceX REST API:** The requested data is parsed, decoded as a Json and normalized into a pandas dataframe. Only the relevant columns are retained and new variables are created. A dictionary is created, with these new variables as the keys. This dictionary is then transformed into our final dataframe after excluding data unrelated to Falcon 9, and after performing some data wrangling to deal with missing values.
  - **Using Webscraping:** Data is sourced from a Wikipedia webpage using the parser BeautifulSoup. Following this, a dictionary with the relevant variables as keys is created and consequently, transformed into a pandas dataframe.

Data Collection – SpaceX API

Link to Notebook

SpaceX API Flowchart:

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

response = requests.get(spacex_url)
```

Request launch data from SpaceX API

```
# Use json_normalize method to convert the json result into a dataframe

data = pd.json_normalize(response.json())
```

Response converted to a json file and saved as a pandas dataframe

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

The data obtained is stored as variables in lists

```
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}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

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

Assigned the lists to a dictionary, then a pandas dataframe.

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

Filtered the data to include only Falcon 9 rocket launches

```
data_falcon9.isnull().sum()
```

```
# Calculate the mean value of PayloadMass column

payloadmassavg = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value

data_falcon9['PayloadMass'].replace(np.nan, payloadmassavg, inplace=True)
```

Checked for missing values, then replaced them with the mean for the variable

Export the data to a CSV

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

Data Collection – Scraping

[Link to Notebook]

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

```python
response = requests.get(static_url).text
html_data.status_code
```

Request the HTML page as a HTTP response

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

Create a BeautifulSoup Object

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

Find all the tables in the wiki page

```python
column_names = []
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
```

Extract the column names

```python
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each value to be an empty list
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']=[]
```

Create a dictionary with the relevant column names as keys

```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:
```

Append data to the keys (Refer to notebook)

```python
headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if  ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

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

Convert the dictionary into a dataframe

```python
df.to_csv('spacex_web_scraped.csv', index=False)
```

Export data to a CSV

2. Data Wrangling

- The data collected so far through SpaceX API and Webscraping has already gone through some stages of data wrangling when we performed data reduction to include only relevant data in the dataset and replaced missing values with the attribute means.
- We now further perform some exploratory data analysis(EDA) to arrive at the class variable or label which enables us to train supervised machine learning models to predict launch outcome.
- A variable 'landing_outcome' is created based on the 'Outcome' column, and from this variable, a set of 'bad_outcomes' is created. Now, a variable is created, '**landing_class**,' based on if the outcome falls in the 'bad_outcomes' set or not, assigning a value 0 if it does, and 1 otherwise. 'Landing_class' is the classification variable that will represent the outcome of each launch. This is the required label.
- **Link to Notebook**

Data Wrangling Flowchart

[Link to Notebook]

```
df.isnull().sum()/df.shape[0]*100
```

Identify and calculate the percentage of missing values in each attribute

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

Determine the number of launches from each site

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

Determine the number and occurrence of each orbit

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

Determine the number of landing outcomes and assign it to variable 'Landing_outcomes'

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

Create a set of outcomes where the second stage did not land successfully, 'bad_outcomes'

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

Create the classification variable 'landing_class'

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

Export to CSV

3. EDA with Data Visualization

Link to Notebook

Summary of Charts plotted:

1. Scatter Point Chart for FlightNumber Vs. PayloadMass
2. Scatter Point Chart for FlightNumber vs. LaunchSite
3. Scatter Point Chart for LaunchSite vs. PayloadMass

(All with landing_outcome overlaid)

- To observe how these variables affect landing_outcome

1. Bar Chart for Success Rate of Each Orbit - To visualize the success rate of each orbit type

2. Scatter Point Chart for FlightNumber vs. Orbit
3. Scatter Point Chart for Payload vs. Orbit type

- To see if there is any relationship between these variables

4. Line Chart for Success Rate - To visualize the yearly trend in launch success

3. EDA with SQL

Link to Notebook

Summary of SQL Queries Performed:

1. To display the names of the unique launch sites in the space mission

```
sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1;
```

2. To display 5 records where the launch sites begin with the string 'CCA'

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

3. To display the total payload mass carried by boosters launched via NASA(CRS)

```
sql SELECT SUM (PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER='NASA (CRS)'
```

4. To display average payload mass carried by booster version 'F9 v1.1%'

```
sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE booster_version LIKE 'F9 v1.1%'
```

5. To list the date of the first successful landing on ground pad

```
sql select min(DATE) from SPACEXTBL;
```

EDA with SQL

Link to Notebook

Summary of SQL Queries Performed:

6. To list boosters that have achieved success in drone ship landing and have a payload mass between 4000 and 6000

```
df_suc_drone_Mass4000_6000=df[(df['Landing _Outcome'] == 'Success (drone ship)') & (df['PAYLOAD_MASS__KG_'] < 6000) & (df['PAYLOAD_MASS__KG_'] > 4
booster=df_suc_drone_Mass4000_6000["Booster_Version"].to_list()
print(f"Booster versions with the description {booster} have masses between 4000 and 6000 and succeded in landing by drone ship ")
```

7. To list the total number of successful and failed mission outcomes

```
%sql select count(MISSION_OUTCOME) as "Successful mission" from SPACEXTBL where MISSION_OUTCOME like 'Success%';
```

```
%sql select count(MISSION_OUTCOME) as "Failed mission" from SPACEXTBL where MISSION_OUTCOME like 'Failure%';
```

8. To list the booster versions that have carried the maximum payload mass

```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

EDA with SQL

Link to Notebook

Summary of SQL Queries Performed:

9. To list the records which will display month names, failure landing outcomes in drone ship, booster versions, launch site for the months in 2015

```
%%sql
select substr(Date, 4, 2) as 'month names', 'landing_outcome', booster_version, launch_site from SPACEXTBL
where substr(Date, 7, 4)='2015' and 'Landing_Outcome'='Failure(drone ship)'
```

10. To rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order

```
lower_date=df[df["Date"]=="04-06-2010"]
higher_date=df[df["Date"]=="20-03-2017"]
df_timed=df.iloc[0:31,]
df_timed['Landing _Outcome'].value_counts(ascending=False)
```

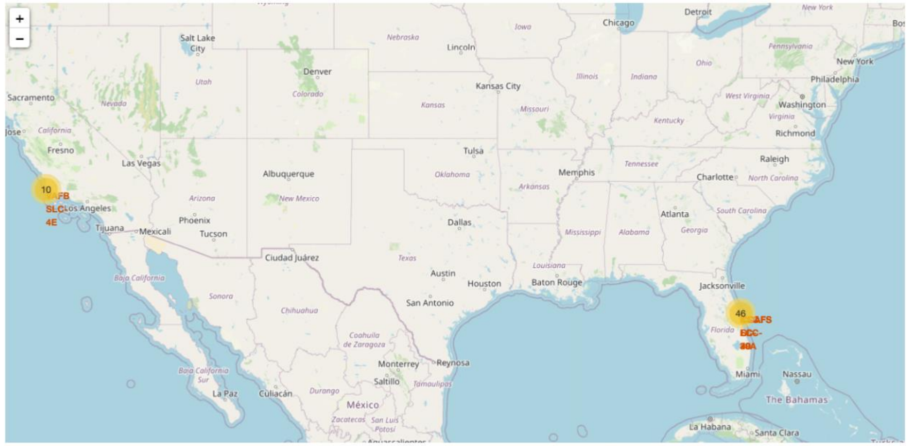
4.

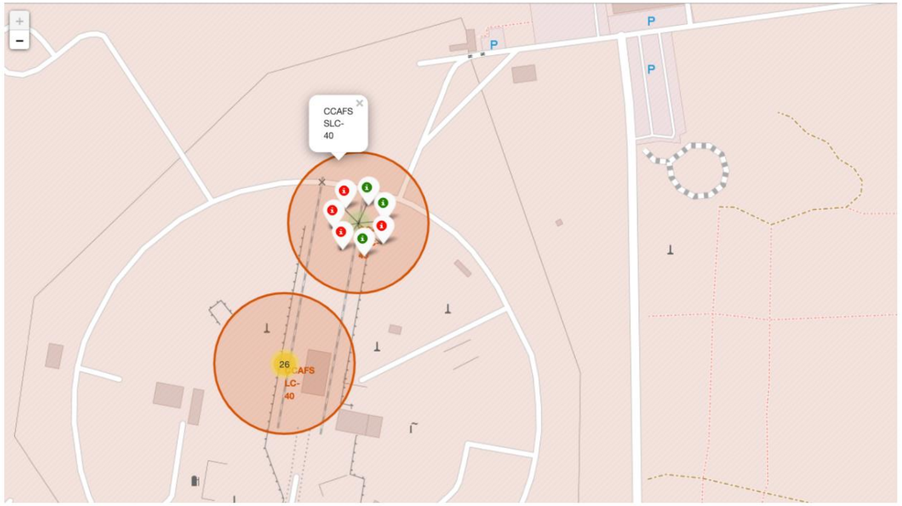
Build an Interactive Map with Folium

[Link to Notebook]

- Map objects created and added to the map include:
  - folium.Circle to add a highlighted circle with a text label on a specific coordinate
  - folium.Marker to represent a point of interest or location where a popup appears if someone drags the mouse over it and further information can be obtained. A marker was added for all launch records. Green markers represent successful launches, and red markers represent failure.
  - MarkerCluster to simplify a map where several markers have the same coordinates
  - MousePosition to obtain the coordinates of any point on the map by hovering the mouse over it
  - Polylines to represent routes between launchsites and the nearest coastline, highway, railroad, city

The objective of this exercise was to find the optimal location for building a launchsite.



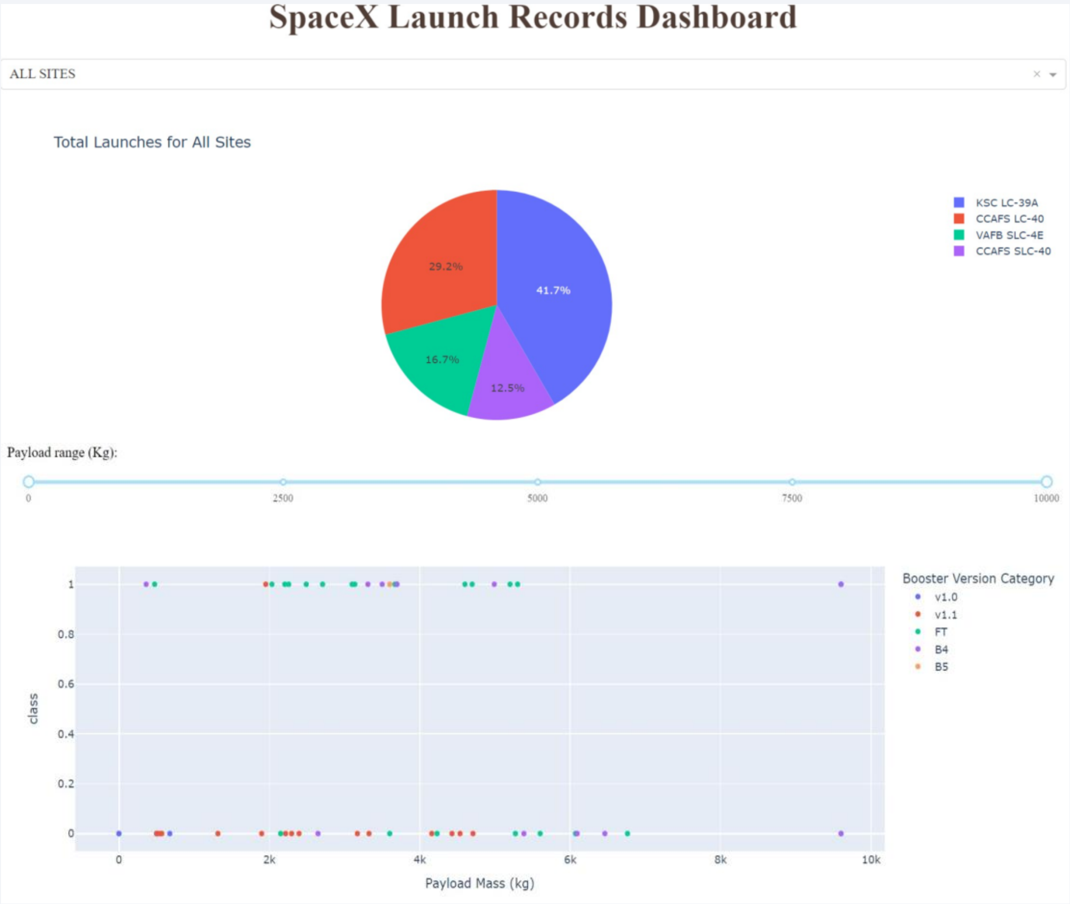


4. Build a Dashboard with Plotly Dash

Link to Notebook

A dashboard was created for users to perform interactive visual analytics on SpaceX launch data in real time. The following objects were added to the dashboard as a result:

- Dropdown list to enable launch site selection
- Pie chart to show total successful launches count for all sites
- Slider to select payload range



5. Predictive Analysis (Classification)

Link to Notebook

```
Y = data["Class"].to_numpy()
Y
```

Create a numpy array from the column of the classification variable

```
X.mean(axis=0)
X.std(axis=0)
```

```
transform = preprocessing.StandardScaler().fit(X)
```

```
X=transform.transform(X)
```

```
X.std(axis=0)
```

Create the array X from the data

```
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

Split the data into training and test data

```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}
lr=LogisticRegression()
```

```
logreg_cv = GridSearchCV(lr, parameters, cv=10)
```

```
logreg_cv.fit(X_train, Y_train)
```

Train a logistic regression model and select the hyperparameters

```
yhat=logreg_cv.predict(X_test)
```

```
logreg_cv.score(X_test, Y_test)
```

```
plot_confusion_matrix(Y_test,yhat)
```

Calculate the test accuracy and plot the confusion matrix

Similarly, train and test the accuracy of the SVM, Decision Tree, KNN models (Check notebook for code)

```
print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
print( 'Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
print('Accuracy for K nearsdt neighbors method:', knn_cv.score(X_test, Y_test))
```

Check which method has the highest accuracy

Results

The results obtained thus far are summarized in:

- **Section 2**: Insights drawn from EDA (Data Visualization and SQL)
- **Section 3**: Interactive Analytics: Launch Sites Proximities Analysis
- **Section 4**: Interactive Analytics: Dashboard with Plotly Dash
- **Section 5**: Results of Predictive Analysis

Topic Summary

- EDA with Data Visualization
- EDA with SQL

Section 2

Insights drawn from EDA

Section 2

Insights drawn from EDA

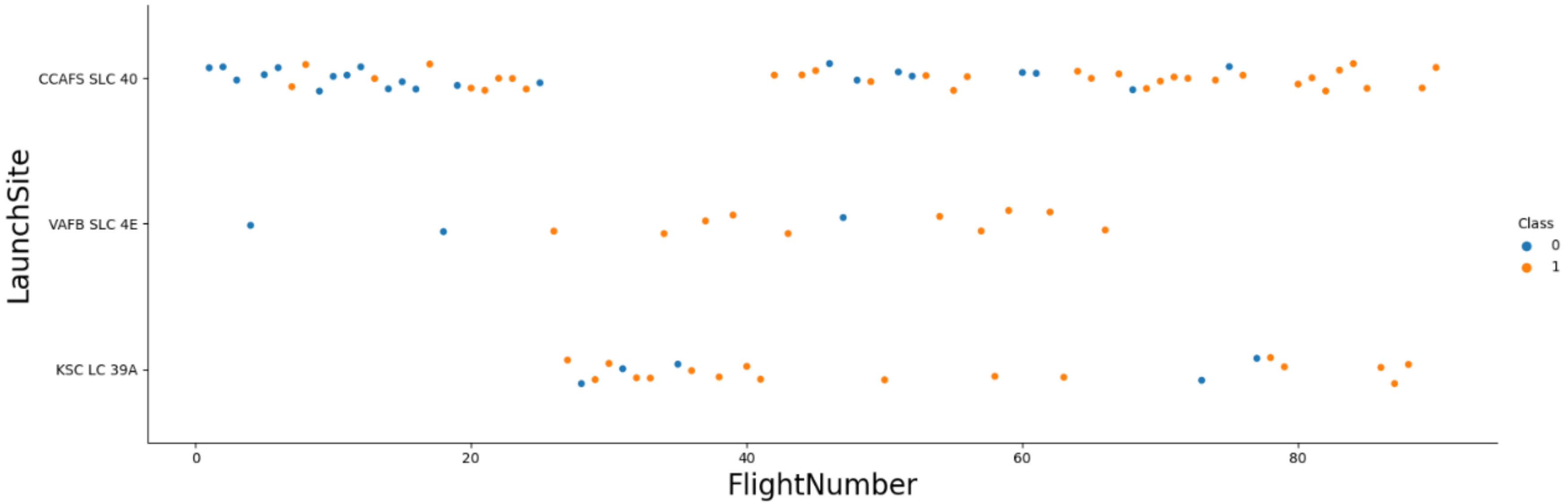
Topic Summary

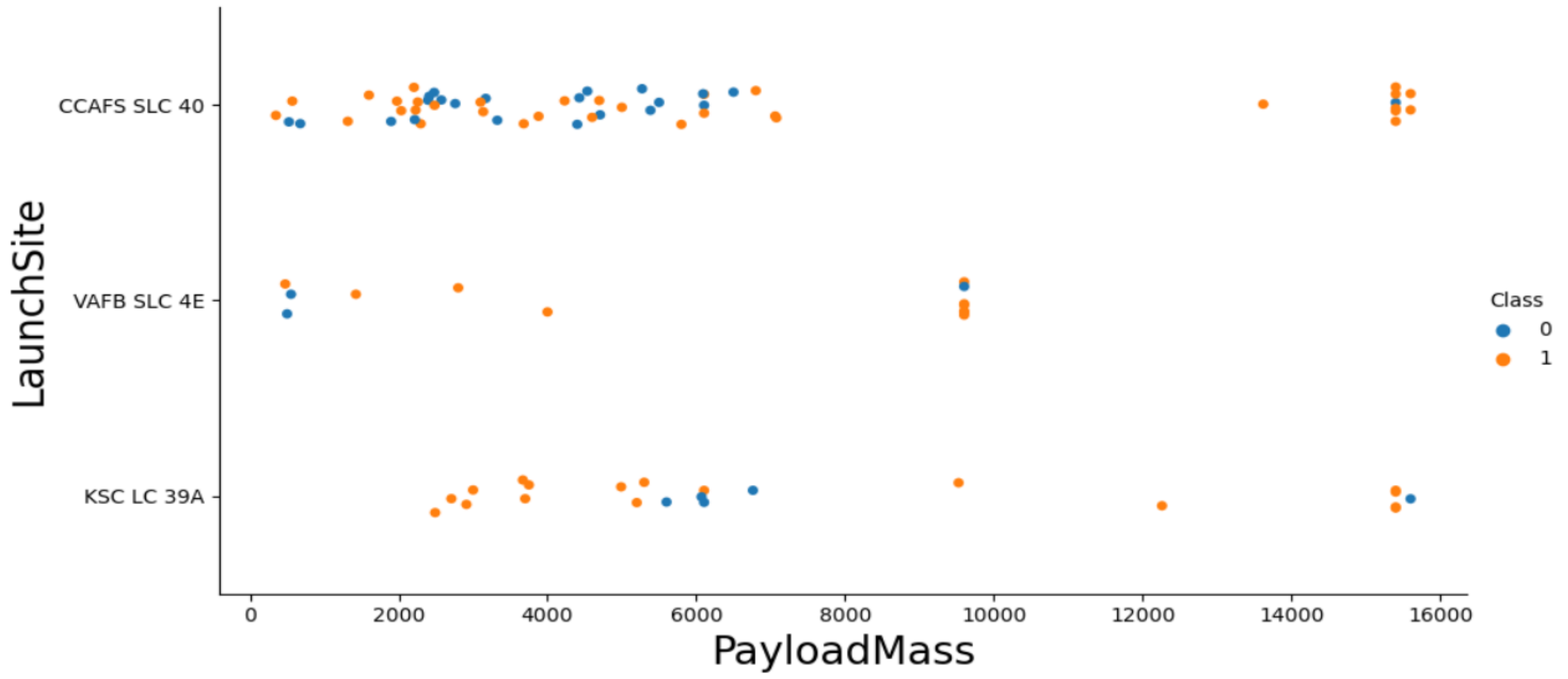
- EDA with Data Visualization
- EDA with SQL

Insights from EDA with Data Visualization

Flight Number vs. Launch Site

- CCAFS SLC 40 has the highest number of launches.
- Flight number seems to be positively correlated with launch site success.



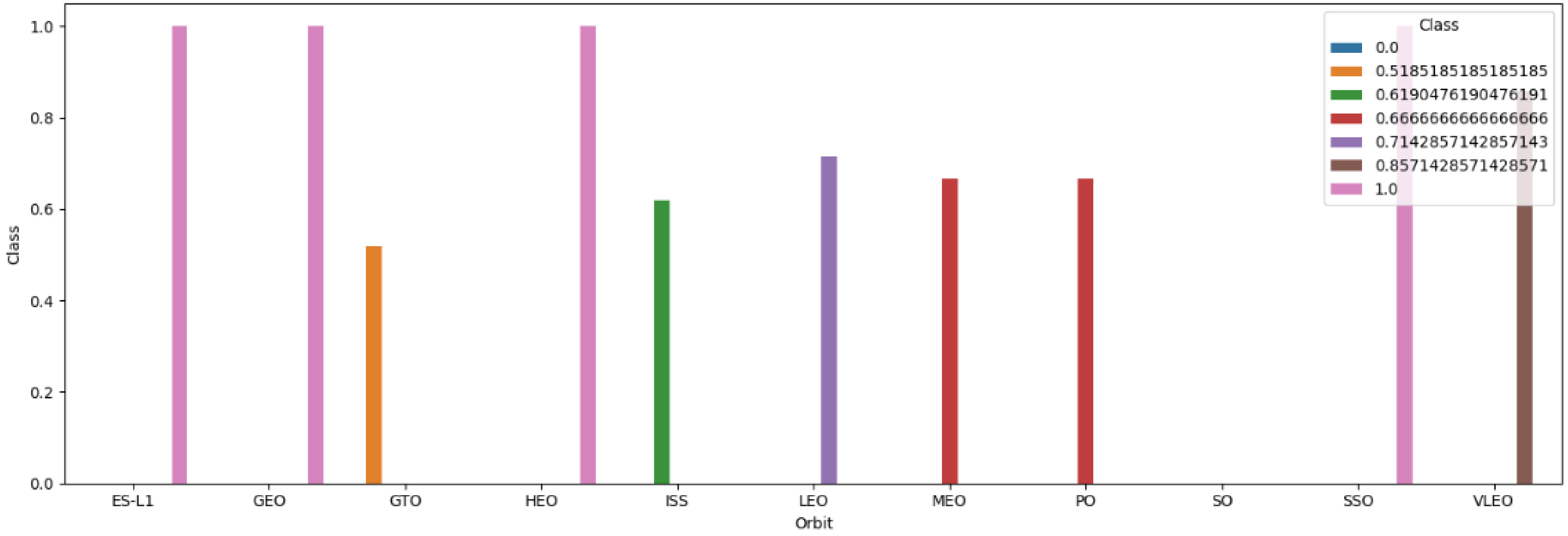


Payload vs. Launch Site

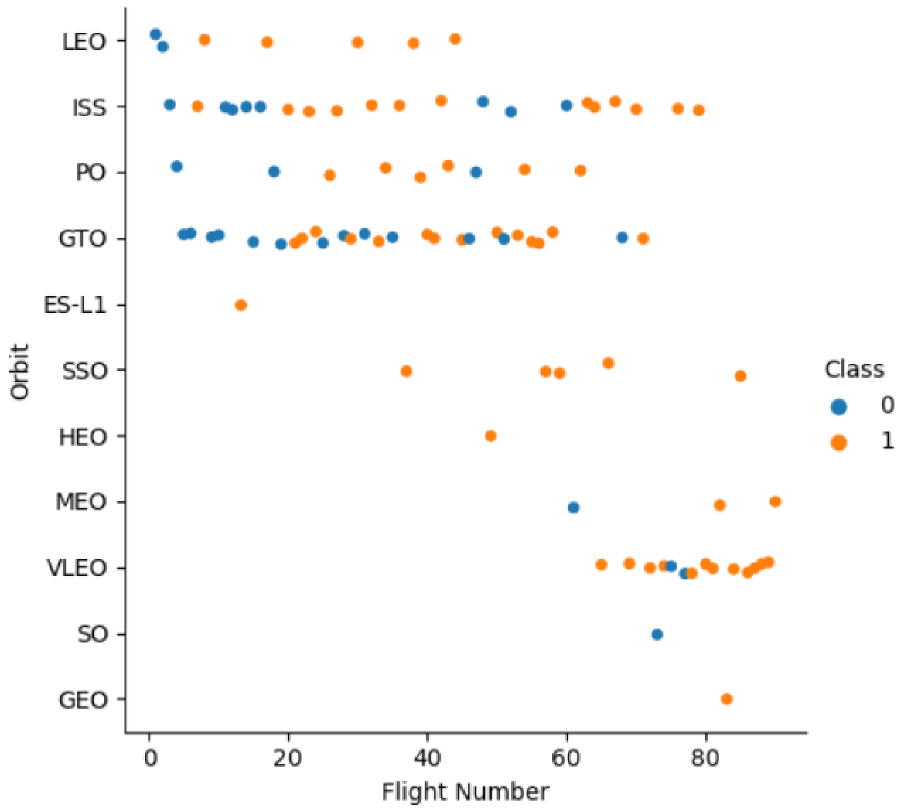
- Launches with lower payloads have majorly been from launch site CCAFS SLC 40, but some of the highest payload launches have also been made from here.
- With an increase in payload mass, launch site success rate seems to be improving for sites that perform higher payload mass launches.

Success Rate vs. Orbit Type

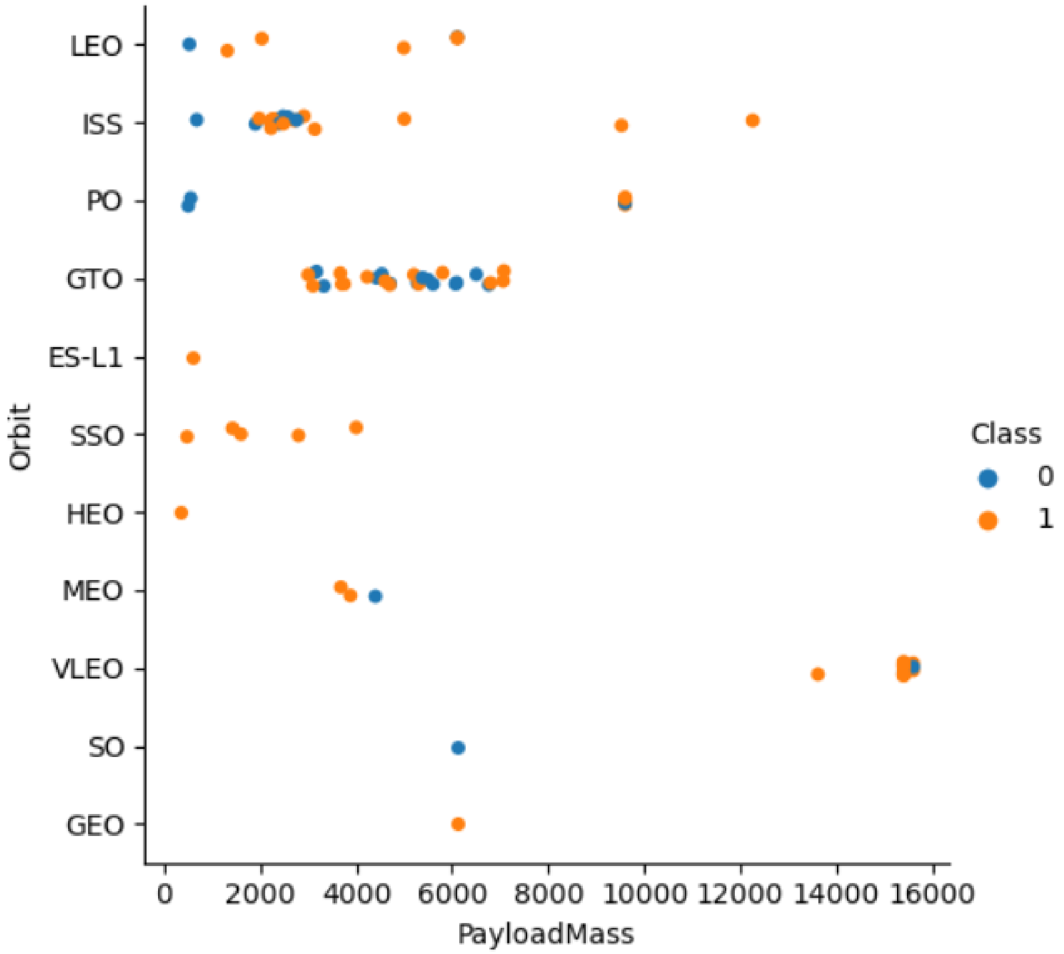
- The orbits ES-L1, GEO, HEO, SSO have 100% success rate, followed by VLEO with a high success rate of around 86%.
- GTO has the lowest success rate, around 52%.



Flight Number vs. Orbit Type



- In recent years, orbit VLEO is being preferred for more launches.
- VLEO also has a high success rate.
- SSO has had fewer launches, but has 100% success rate.
- ES-L1, HEO, GEO all have 100% success rates, but have handled only one launch each.
- There seems to be no apparent relationship between flight number and orbit type, however.

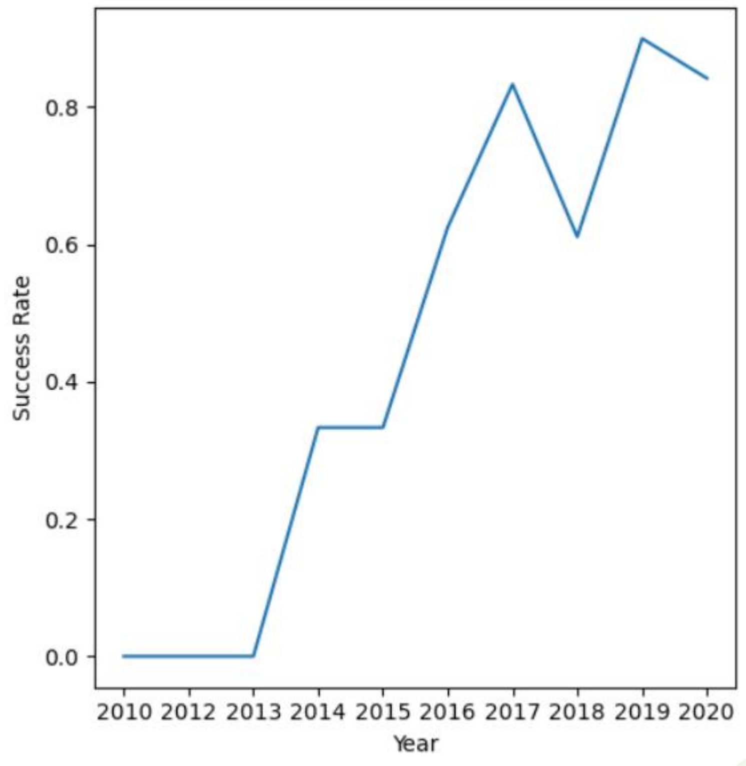


Payload vs. Orbit Type

- The highest payloads are handled by VLEO, ISS, and PO.
- LEO and ISS perform better when payload> 3000.
- PO does not have a good success rate.
- SSO has 100% success rate and has been used with a maximum payload of around 4000.
- ES-L1, HEO, SSO have gained success at the lower payloads where LEO, ISS, PO have had failed launches.
- GTO has the lowest success rate as per success rate vs. Orbit type chart, and the same is represented here.
- SO has failed at the payload for which GEO succeeded.

Launch Success Yearly Trend

- Successful launches started occurring in 2013. Till 2015, around 40% launches would be successful. The success rate remained constant between 2014 and 2015, however the steep rise thereafter until 80% success rate is achieved in 2016 indicates that technological upgrades were definitely under process.
- Success rate dropped to around 60% in 2018, rose to around 90% in 2019, and arrived around 80% again in 2020.
- The success rate seems to have stabilized at a higher value in recent years thanks to R&D.



Insights from EDA with SQL

All Launch Site Names

- There are 4 unique launch sites in the data. Their names have been displayed with the help of the SQL query on the right.

Display the names of the unique launch sites in the space mission

```
sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1;
```

```
 * sqlite:///my_data1.db
Done.
```

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

Launch Site Names Begin with 'CCA'

Display 5 records where launch sites begin with the string 'CCA'

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

```
 * sqlite:///my_data1.db
Done.
```

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

- All of the 5 records correspond to CCAFS LC 40.

Total Payload Mass

- The total payload mass carried by boosters launched by NASA is 45596 kg, and it is generated with the query on the right.

Display the total payload mass carried by boosters launched by NASA (CRS)

```
sql SELECT SUM (PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER='NASA (CRS)'
```

```
 * sqlite:///my_data1.db
Done.
```

SUM (PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- The average payload mass carried by the booster version F9 v1.1 is 2534.67 kg as generated with the help of the query on the right.

Display average payload mass carried by booster version F9 v1.1

```
sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE booster_version LIKE 'F9 v1.1%'
```

```
 * sqlite:///my_data1.db
Done.
```

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

First Successful Ground Landing Date

- The first successful ground landing date was 01-03-2013.

```
sql select min(DATE) from SPACEXTBL;
```

```
 * sqlite:///my_data1.db
Done.
```

min(DATE)
01-03-2013

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
df_suc_drone_Mass4000_6000=df[(df['Landing _Outcome'] == 'Success (drone ship)') & (df['PAYLOAD_MASS__KG_'] < 6000) & (df['PAYLOAD_MASS__KG_'] > 4000)
booster=df_suc_drone_Mass4000_6000["Booster_Version"].to_list()
print(f"Booster versions with the description {booster} have masses between 4000 and 6000 and succeded in landing by drone ship ")
```

```
Booster versions with the description ['F9 FT B1022', 'F9 FT B1026', 'F9 FT  B1021.2', 'F9 FT  B1031.2'] have masses between 4000 and 6000 and succeded in landing by drone ship
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- The booster versions that have success with drone ship landing and have a payload mass >4000 and <6000 have been displayed above.

Total Number of Successful and Failure Mission Outcomes

- There have been 100 successful missions and 1 failed.

List the total number of successful and failure mission outcomes

```
%sql select count(MISSION_OUTCOME) as "Successful mission" from SPACEXTBL where MISSION_OUTCOME like 'Success%';
```

```
 * sqlite:///my_data1.db
Done.
```

Successful mission
100

```
%sql select count(MISSION_OUTCOME) as "Failed mission" from SPACEXTBL where MISSION_OUTCOME like 'Failure%';
```

```
 * sqlite:///my_data1.db
Done.
```

Failed mission
1

Boosters that Carried Maximum Payload

```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

boosterversion
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

- The above query generates the output column on the right. The booster versions that carried the maximum payloads have been displayed.

2015 Failed Launch Records

- Both of the failure landings on drone ship are attributed to booster version F9 v1.1% launched from CCAFS LC 40 in Florida.

MONTH	booster_version	launch_site
1	F9 v1.1 B1012	CCAFS LC-40
4	F9 v1.1 B1015	CCAFS LC-40

```
%%sql
select substr(Date, 4, 2) as 'month names', 'landing_outcome', booster_version, launch_site from SPACEXTBL
where substr(Date, 7, 4)='2015' and 'Landing_Outcome'='Failure(drone ship)'
```

```
lower_date=df[df["Date"]=="04-06-2010"]
higher_date=df[df["Date"]=="20-03-2017"]
df_timed=df.iloc[0:31,]
df_timed['Landing _Outcome'].value_counts(ascending=False)
```

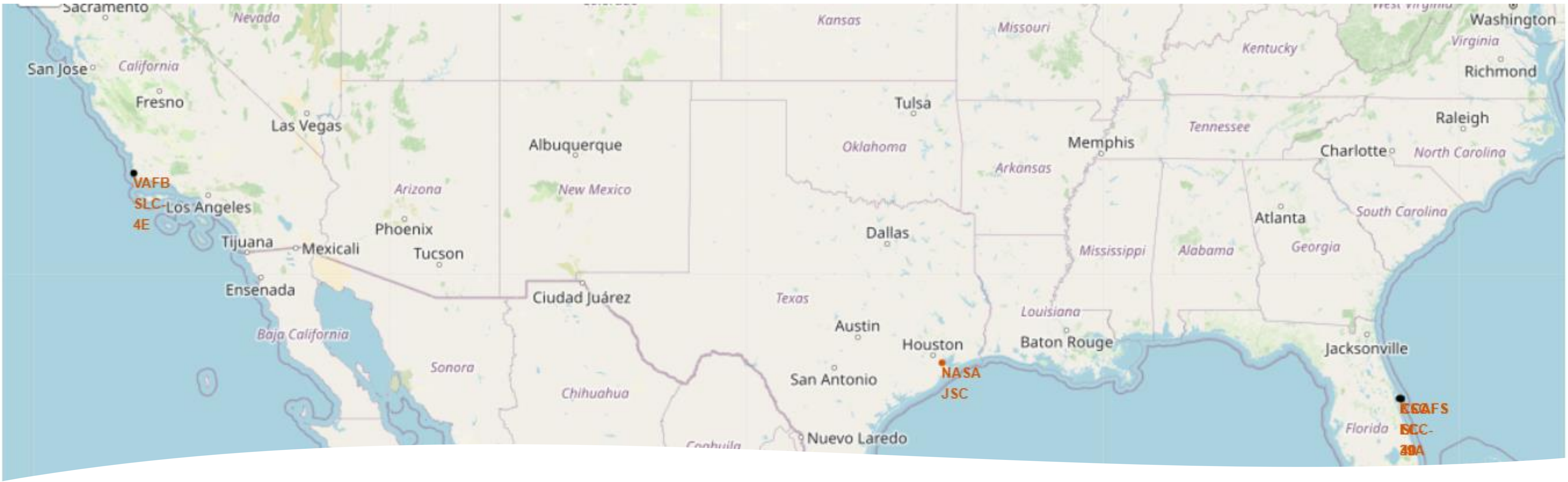
```
No attempt                10
Failure (drone ship)       5
Success (drone ship)       5
Controlled (ocean)         3
Success (ground pad)       3
Failure (parachute)        2
Uncontrolled (ocean)       2
Precluded (drone ship)     1
Name: Landing _Outcome, dtype: int64
```

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

- The ranking has been achieved with the SQL query as outlined on the left.

Section 3

Launch Sites Proximities Analysis

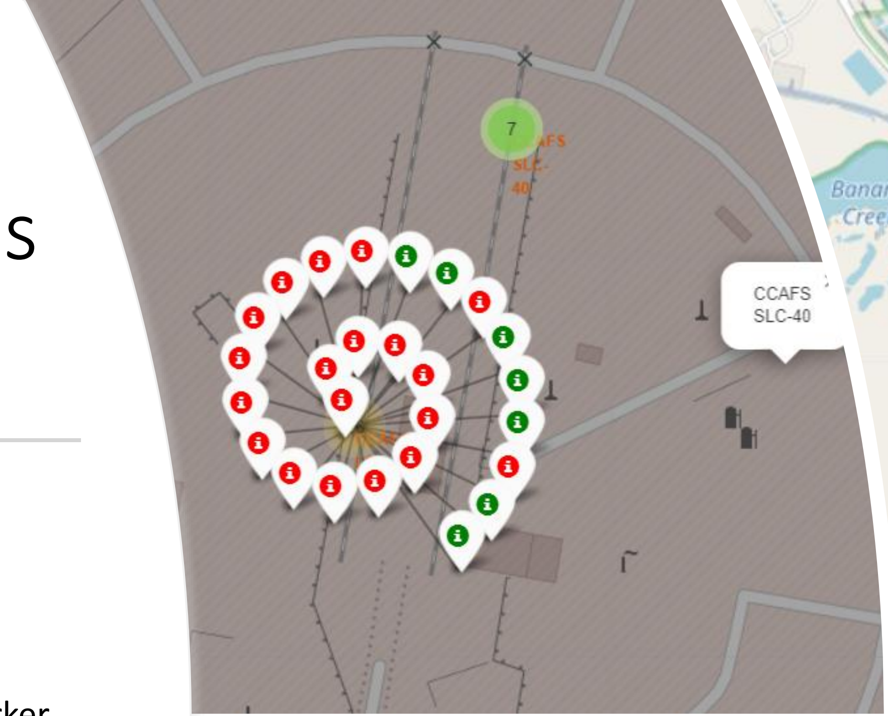


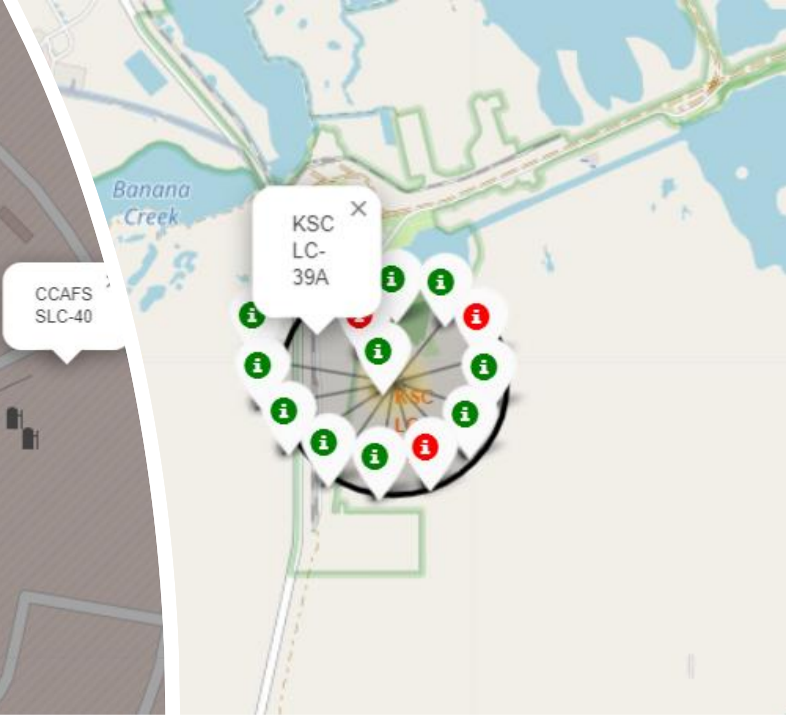
All Launch Sites

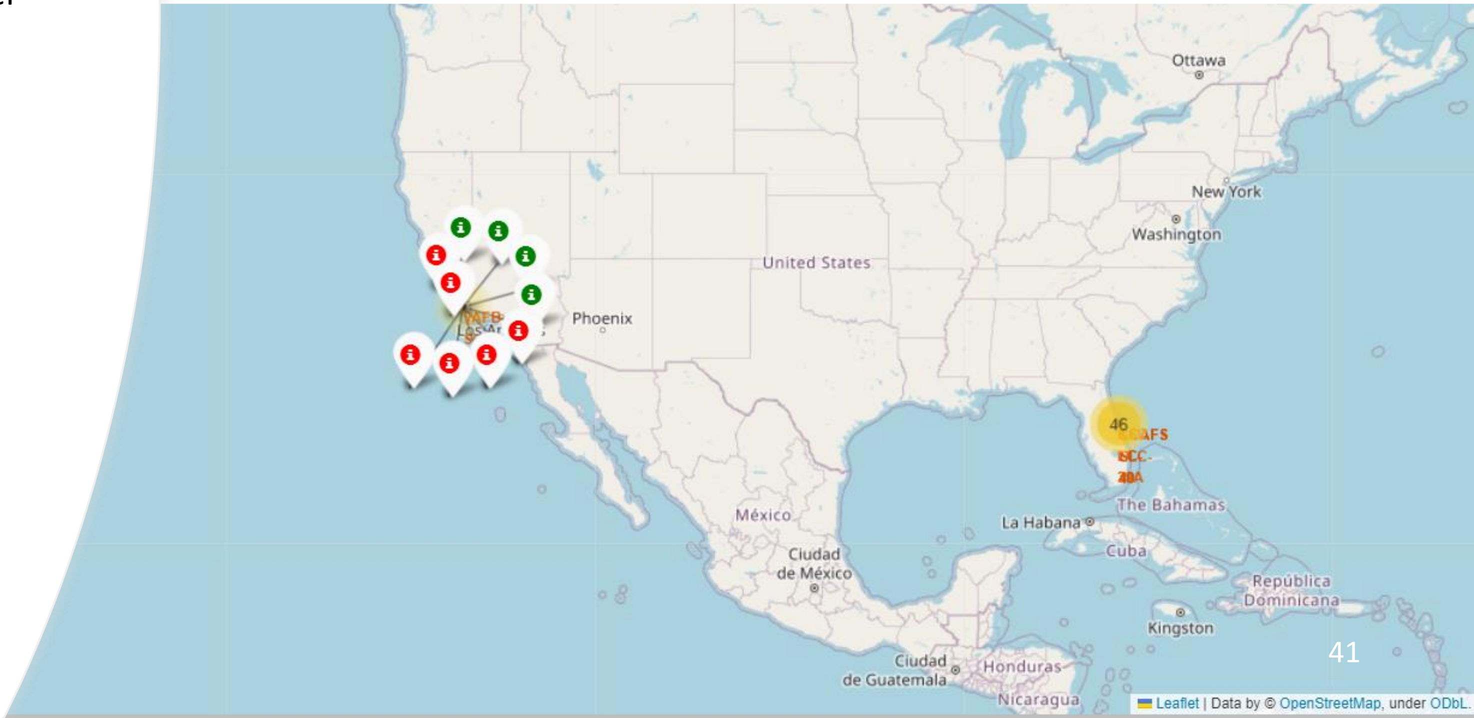
- The launch sites are located on the coastline in Florida, Texas, and California.
- All the launch sites are in the southern states, and hence, closer to the equator.
- Being located on the coastline has public safety benefits in case of a crash as the debris can fall directly into the ocean.
- Being close to the equator helps harness the thrust of the earth's spinning motion which is higher near the equator. This aids in building the right escape velocity for a rocket to escape the earth's gravity.

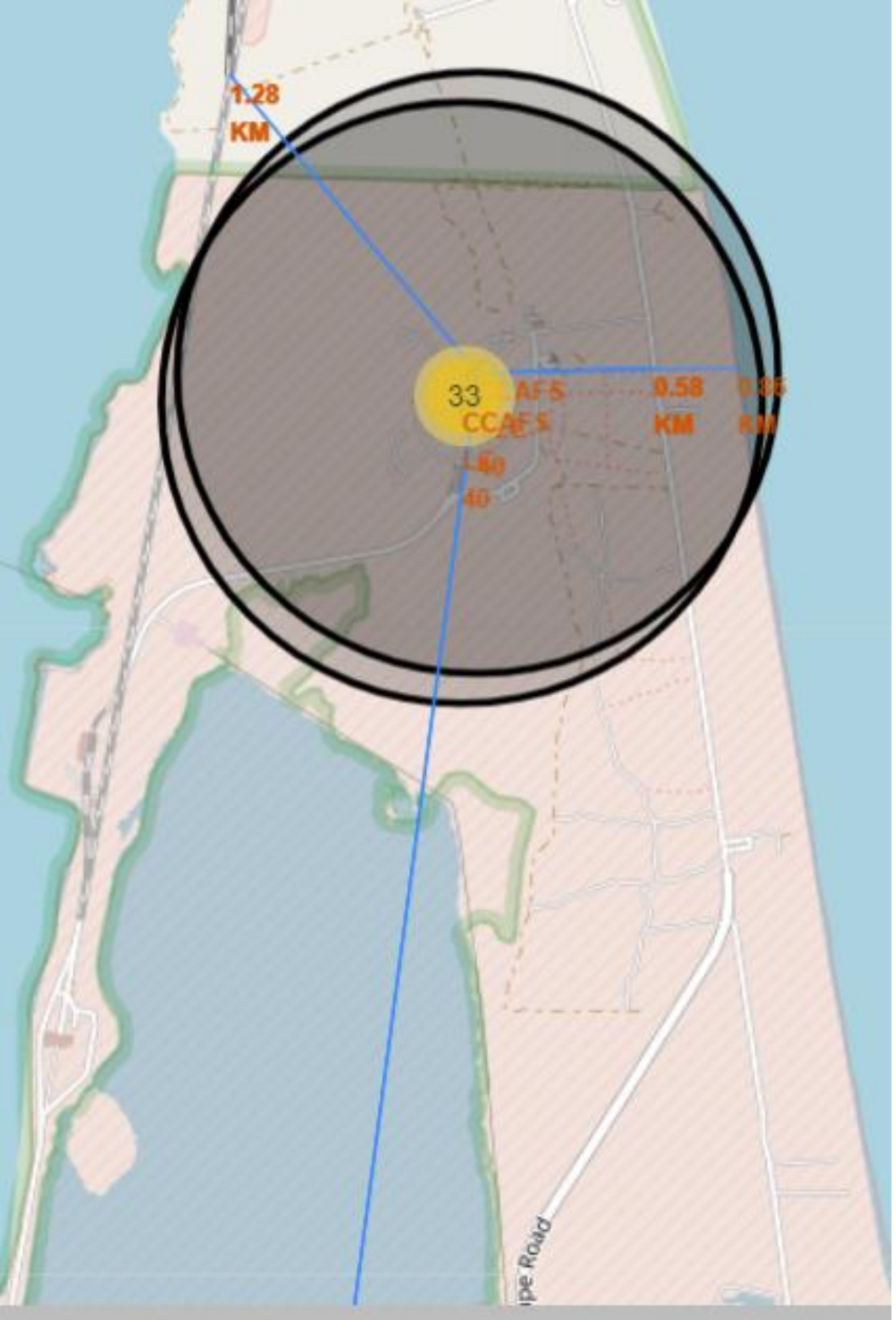
Successful/Failed Launches for each Site

- In order to analyze the launch outcomes for each of the sites, individual markers have been added for every launch from a particular site.
- A green marker indicates success, a red marker failure. Marker clusters have been used to organize them for easy viewing.
- The VSFB site has had 6 failures and 4 successful launches.
- There are 3 launch sites in Florida.
    - KSC LC 39A has had 3 failures and 10 successful launches.
    - CCAFS SLC 40 and CCAFS LC 40 together have had the highest number of launches among all the sites, but only 10 successful launches against 33 attempts.









Distances from Launch Site to its Proximities

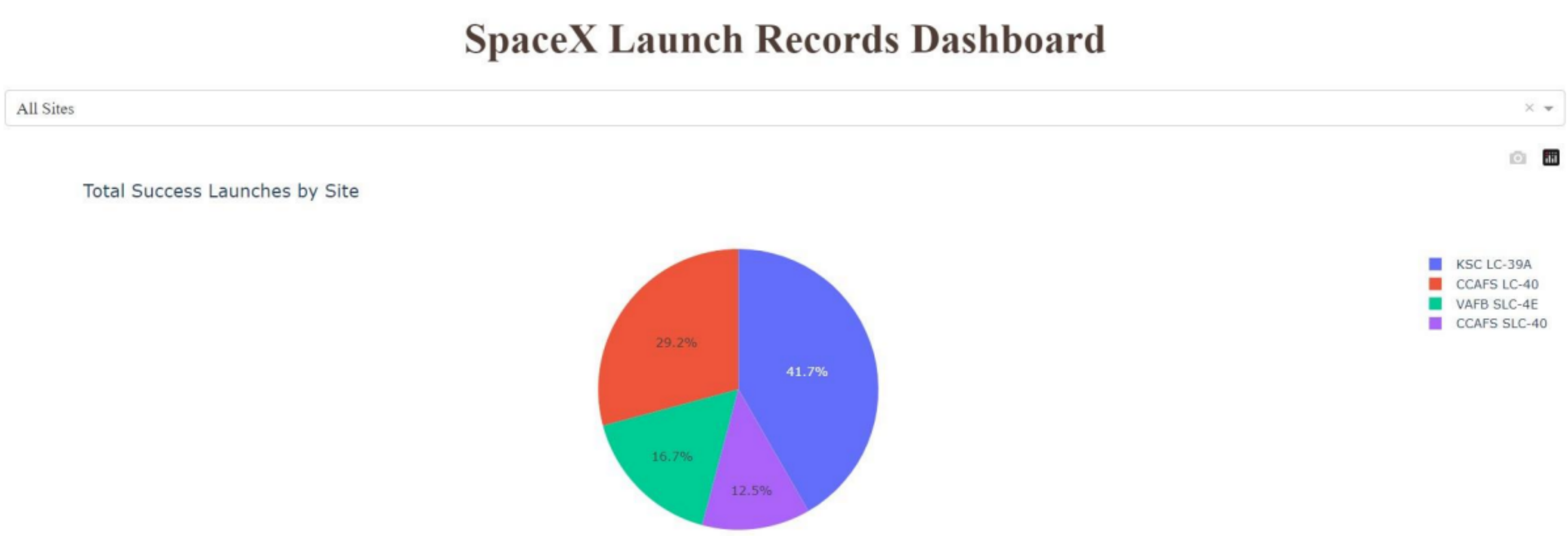
- With the help of MousePosition and Polylines, we can observe the distance of the launch sites from important locations like the nearest highway, railroad, city, and the coastline.
- The launch sites are very close to the nearest highways, railroads, and coastline, but away from cities. This makes sense as having highways and railroads in close proximity allows for cheaper transportation of people, material, and cargo. Being situated away from cities is to ensure public safety.

Section 4

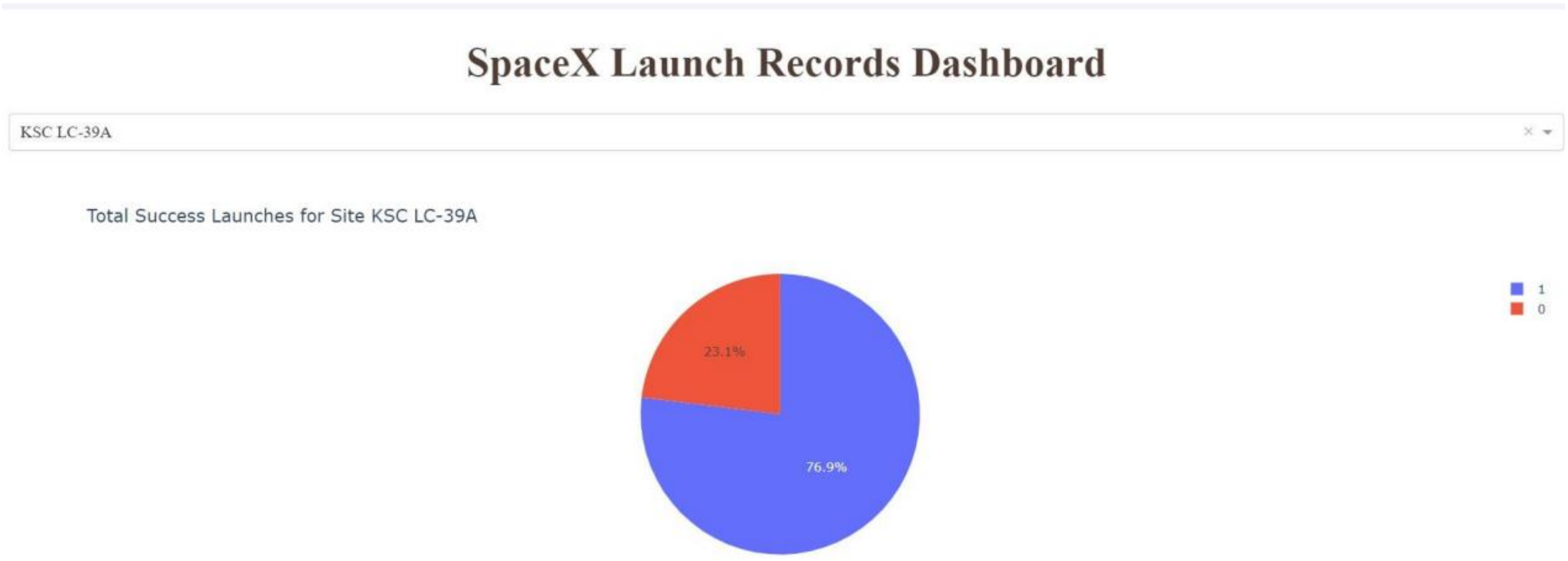
Build a Dashboard with Plotly Dash

Successful Launches by Launch Sites

- KSC LC 39A has the highest number of successful launches, and CCAFS SLC 40 has the lowest.



Launch Success Pie Chart for KSC LC 39A



- KSC LC 39A has a launch success rate of 76.9%.

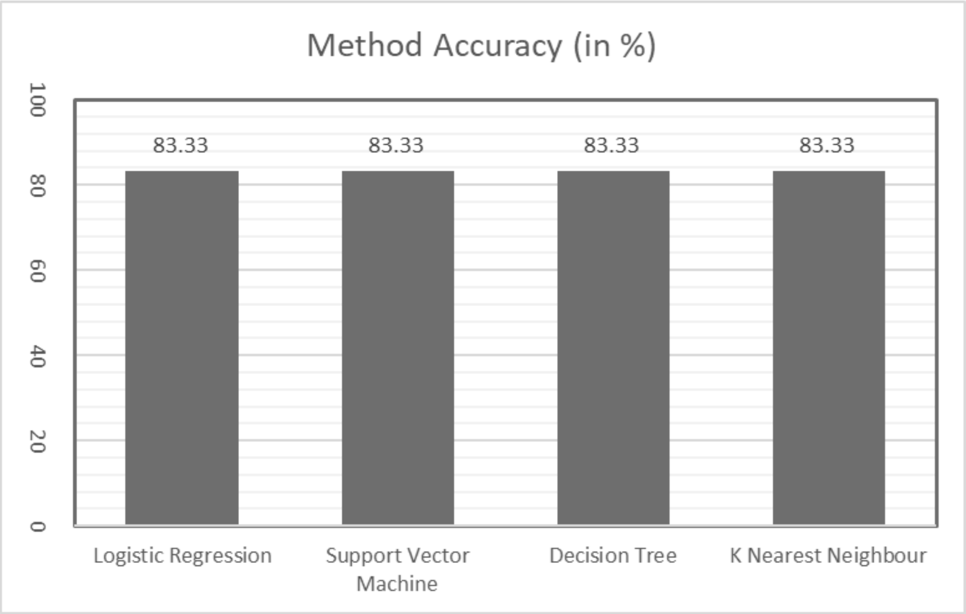
Payload vs. Launch Outcome

- The screenshot on the top is for payloads upto 4000 kg, the one on the bottom is for heavier payloads (4000 kg to 10000 kg).
- Higher payload launches have a higher chance of launch success than lower payload ones.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



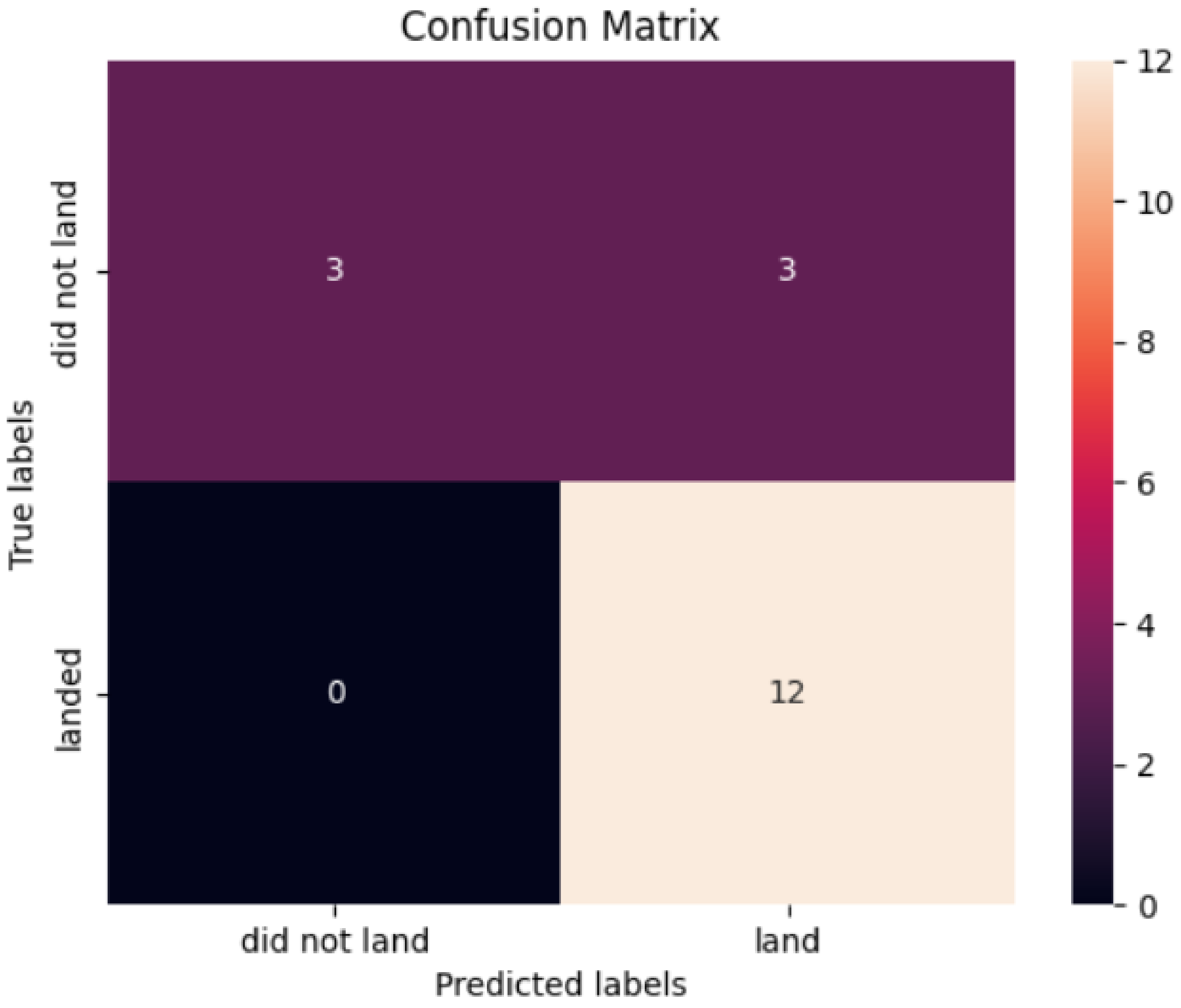

```
print('All the methods perform the same as is proved by the accuracy scores.')
```

```
All the methods perform the same as is proved by the accuracy scores.
```

- The models used were:
  - Logistic Regression
  - Support Vector Machine
  - Decision Tree
  - K Nearest Neighbour
- All the models have the same accuracy of 83.33%.

Confusion Matrix

- All the models have the same accuracy rate and their confusion matrices look the same. The false positives appear to be an issue, however. This means that a launch may falsely be predicted as 'landed successfully' when it actually does not.



Conclusion

Pertaining to Falcon 9 rocket launches,

- The highest number of successful launches were made from KSC LC 39A in Florida with a success rate of 76.9%, followed by CCAFS LC 40, also in Florida. Florida's 3rd launch site, CCAFS SLC 40 however, has the lowest success rate among all the launch sites while having made the highest number of launches.
- More launches are being made from KSC LC 39A and VAFB SLC 4E in recent years and they have better success rates.
- 4 of the orbits being used have 100% success rate. The lowest success rate of all the orbits was 52% (GTO).
- The launch sites are optimally located on the coastlines, are away from cities and are very close to highways and railways. This enables easier transport of materials, cargo, and people, along with public safety in case of a crash.

Conclusion

- The launch sites are producing more successful launches over the years as the variable 'FlightNumber'
  increases. This reflects the continuous efforts at technology perfection by SpaceX and eventually, they will be able to
  minimize launch failures thereby ensuring lower costs.
- The yearly launch success rate of all sites taken together has reached 80% in 2020 from 40% in 2015 and 0% before the first successful
  launch in 2013.
- For predicting launch outcome, all the 4 models trained in the project (LR, SVM, Decision Tree, KNN) provide equally accurate results, with a high accuracy rate of 83.33%. However, we have to watch out for the False Positives.

Appendix

- Please find all resources uploaded on my Github.

Thank you!