# COURSERA IBM DATASCIENCE CAPSTONE

- By khushi

# 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

1. 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. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

2. Problems you want to find answers

   -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.

# Methodology

- Data collection :

  Data was collected using SpaceX API and web scraping from Wikipedia.

- Perform data wrangling

  One-hot encoding was applied to categorical 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

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

# EDA and interactive visual analytics methodology

- Exploratory data analysis use to early data analysis on data to see and find the needful parameters and data
- Visual analytics compromises the steps to visual the data in less is more attractive ,less is more effective strategies is applied

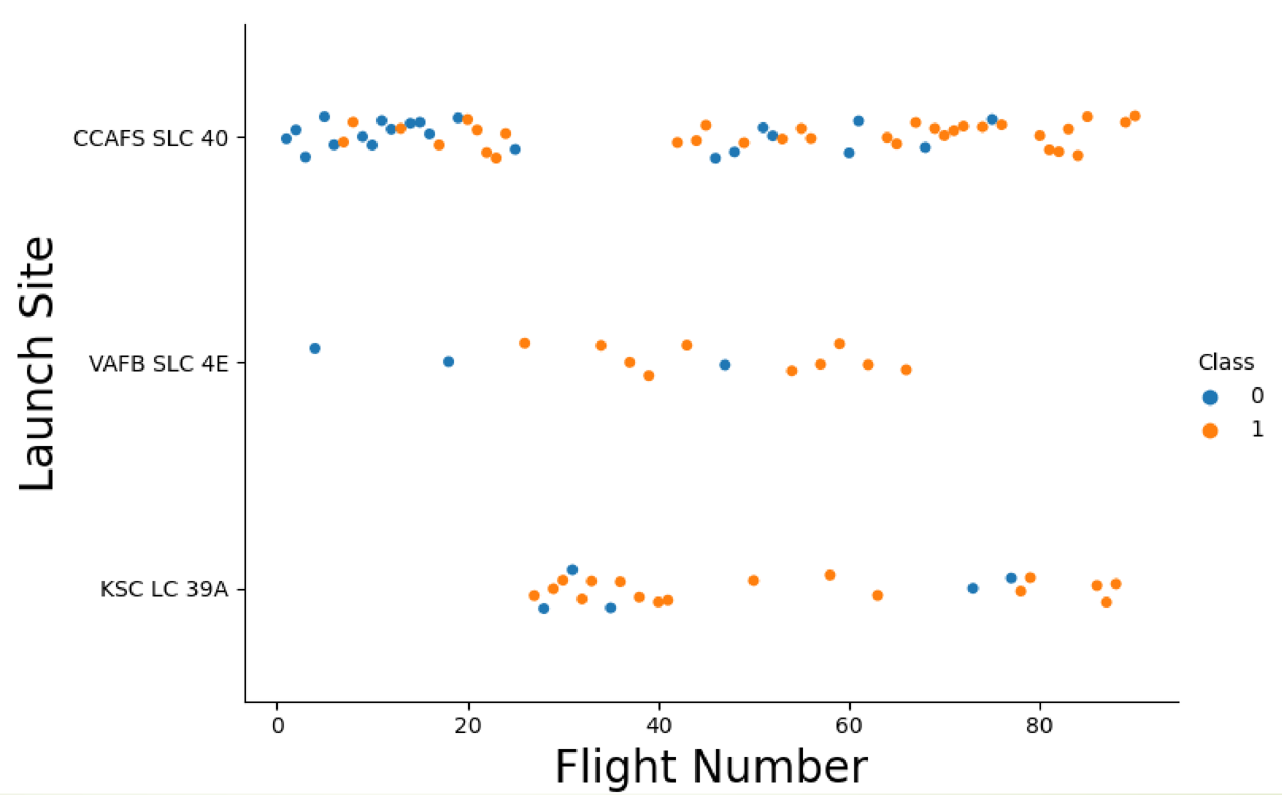
# Predictive analysis methodolgy

- Predicted analysis is done to predicted the  data from test data using the model which is created using training data.

# Data visualization section

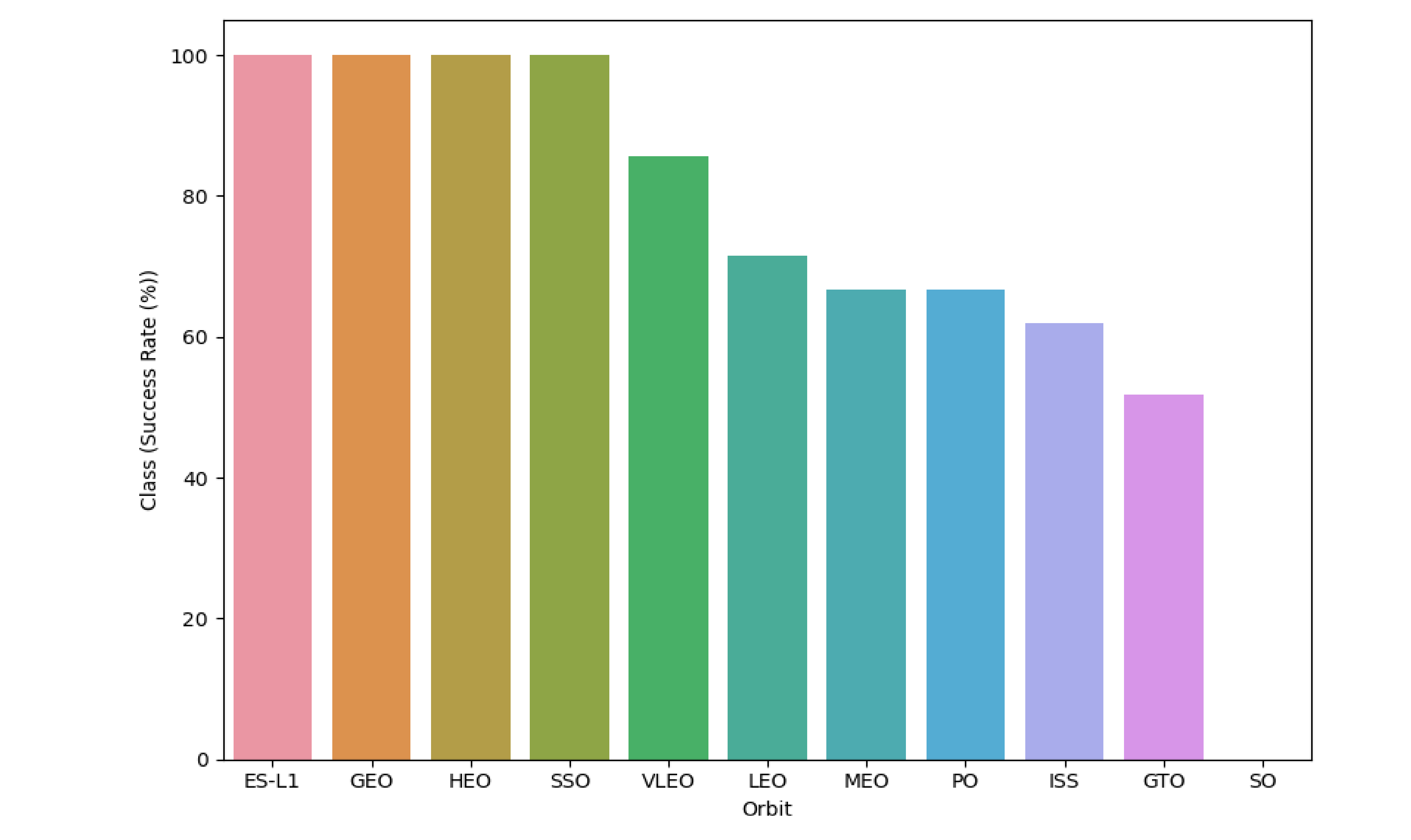
# Visualization

From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.

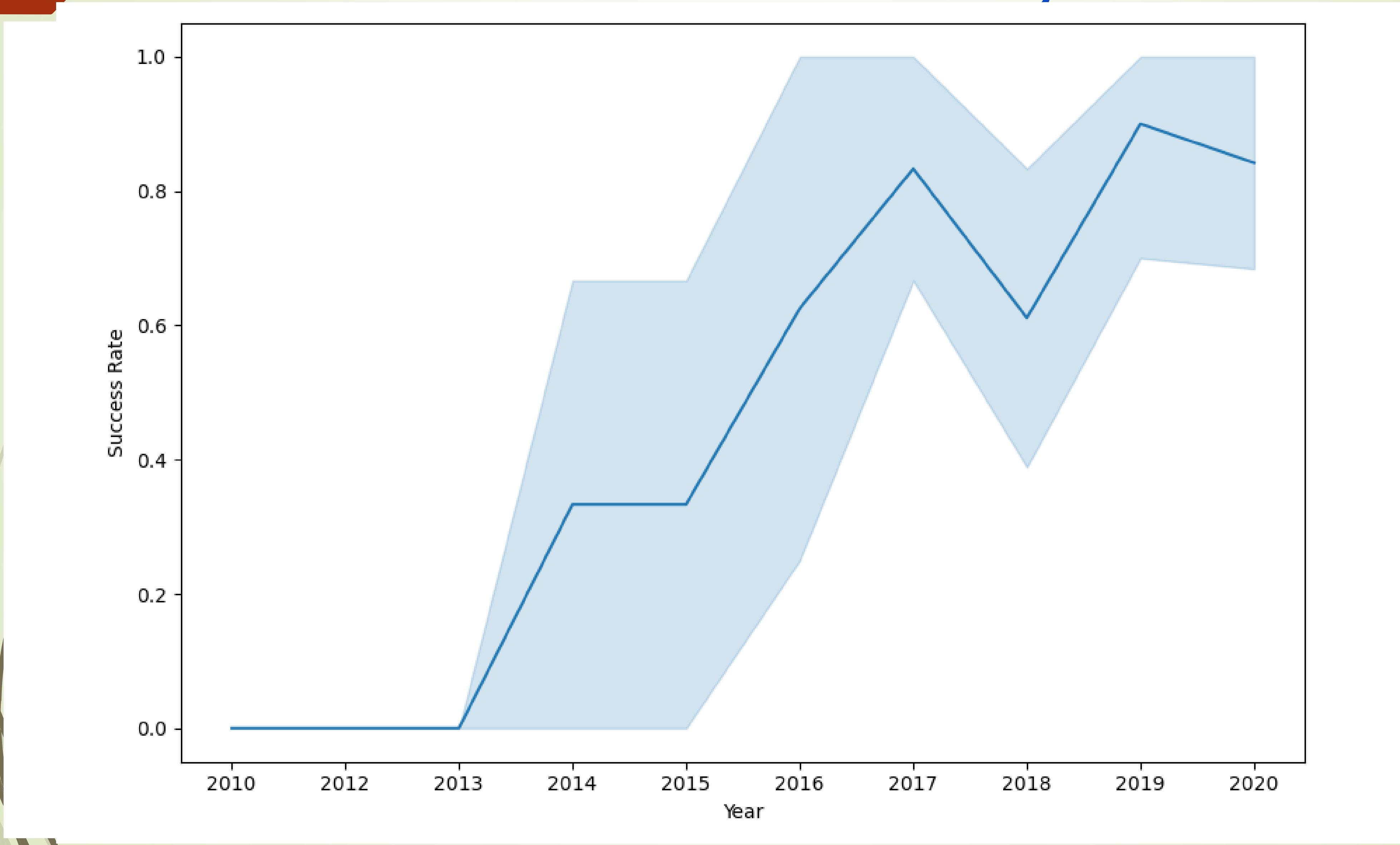


# Visualization Payload vs. Launch Site

# Visualization Success Rate vs. Orbit Type



# Visualization Launch Success Yearly Trend



# SQL ANALYSIS section

# SQL data analysis

## Task 1

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

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

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

| Launch_Site |
|---|
| CCAFS LC-40 |
| VAFB SLC-4E |
| KSC LC-39A |
| CCAFS SLC-40 |
| None |

# SQL data analysis

Task 2

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

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

[10]

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

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 06/04/2010 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0.0 | LEO | SpaceX | Success | Failure (parachute) |
| 12/08/2010 | 15:43:00 | F9 v1.0 B0004 | CCAFS LC-40 | Dragon demo flight C1, two CubeSats, barrel of Brouere cheese | 0.0 | LEO (ISS) | NASA (COTS) NRO | Success | Failure (parachute) |
| 22/05/2012 | 7:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525.0 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 10/08/2012 | 0:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500.0 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 03/01/2013 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677.0 | LEO (ISS) | NASA (CRS) | Success | No attempt |

# SQL data analysis

## Task 3

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

[11]
```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where CUSTOMER = 'NASA (CRS)'
```

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

| sum(PAYLOAD_MASS__KG_) |
| --- |
| 45596.0 |

# SQL data analysis

## Task 4

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

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

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

| avg(PAYLOAD_MASS__KG_) |
|---|
| 2928.4 |

# SQL data analysis

## Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

*Hint:Use min function*

[14]

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

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

| min(DATE) |
|---|
| 01/08/2018 |

# SQL data analysis

## Task 6

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

[18]
```
%sql select BOOSTER_VERSION from SPACEXTBL where Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4000 and PAY
```

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

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

# SQL data analysis

## Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome"
```

[28]

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

| Mission_Outcome | Total |
|---:|---:|
| None | 0 |
| Failure (in flight) | 1 |
| Success | 98 |
| Success | 1 |
| Success (payload status unclear) | 1 |

# SQL data analysis

## Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

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

[29]

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

| Booster_Version |
|---|
| F9 B5 B1048.4 |
| F9 B5 B1049.4 |
| F9 B5 B1051.3 |
| F9 B5 B1056.4 |
| F9 B5 B1048.5 |
| F9 B5 B1051.4 |
| F9 B5 B1049.5 |
| F9 B5 B1060.2 |
| F9 B5 B1058.3 |
| F9 B5 B1051.6 |
| F9 B5 B1060.3 |
| F9 B5 B1049.7 |

PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

# SQL data analysis

## Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

**Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.**

```
%sql SELECT substr(Date,7,4), substr(Date, 4, 2),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_", "Mission_Outcome", "Landing _Outcome" FROM SPACEXTBL WHERE substr(Date,7,4)='2015' AND "Land
```

Python

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

| substr(Date,7,4) | substr(Date, 4, 2) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Mission_Outcome | "Landing _Outcome" |
|---|---|---|---|---|---|---|---|

# SQL data analysis

Task 10

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.

```
%sql select * from SPACEXTBL where Landing_Outcome like 'Success%' and (DATE between '2010-06-04' and '2017-03-20') order by date desc
```

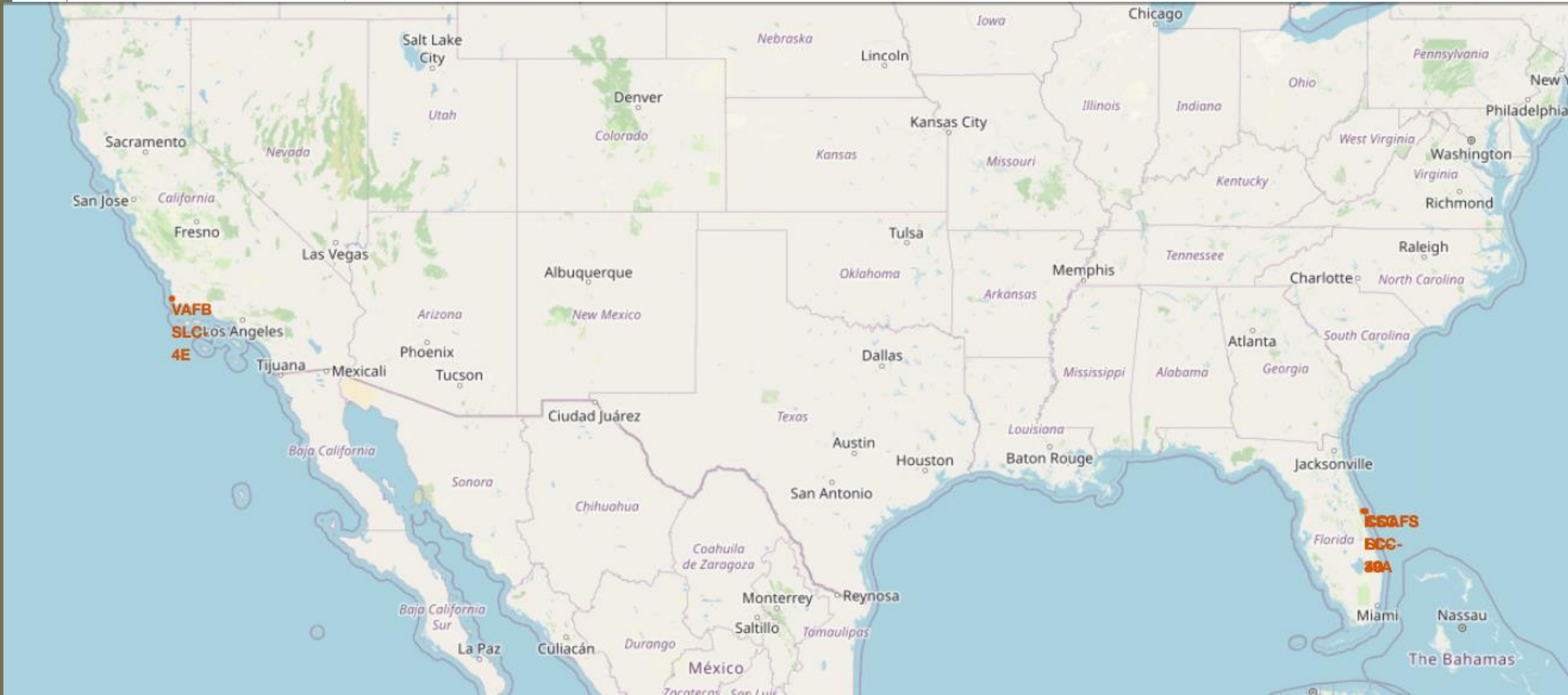
[25]

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

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|---|---|

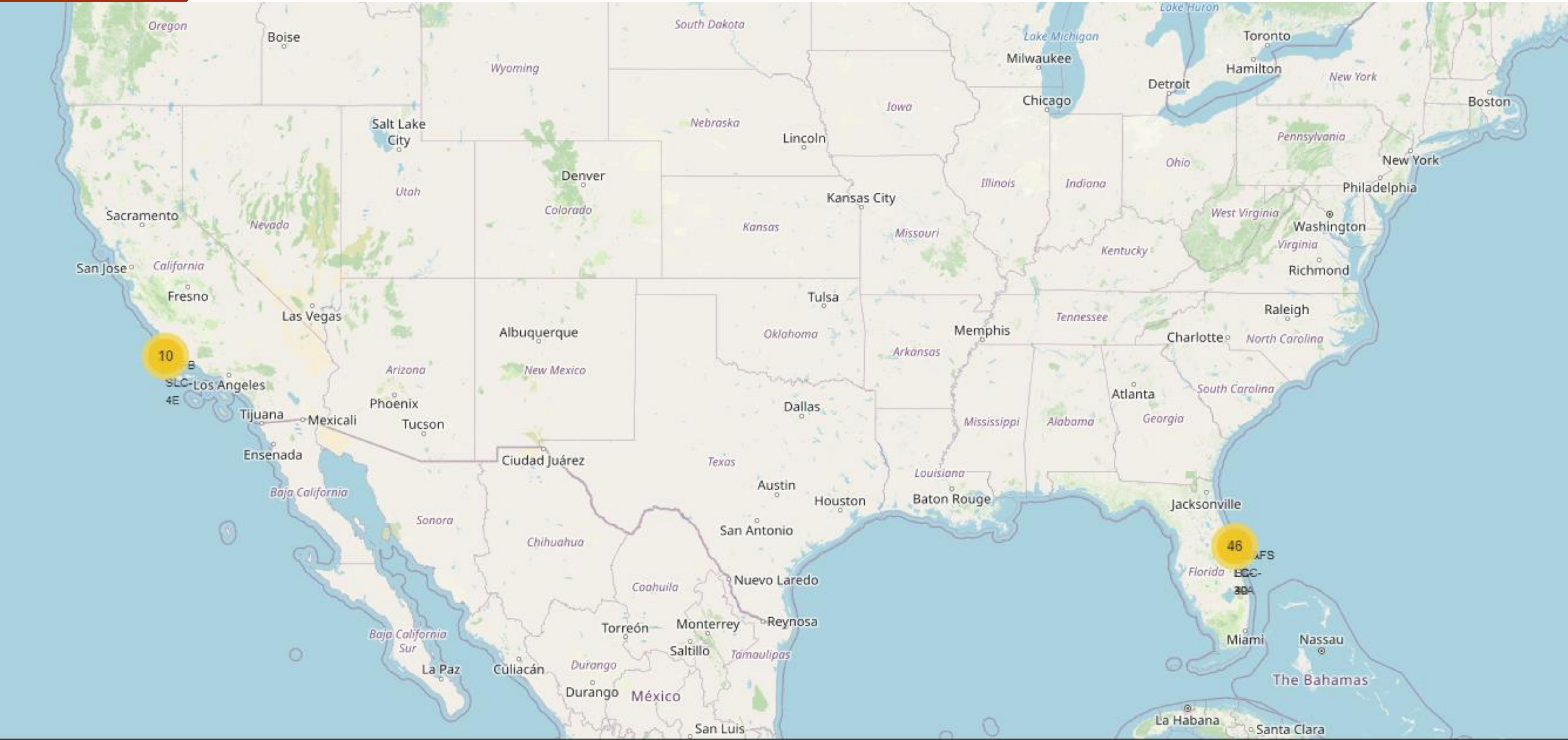
# FOLIUM section

# TASK 1

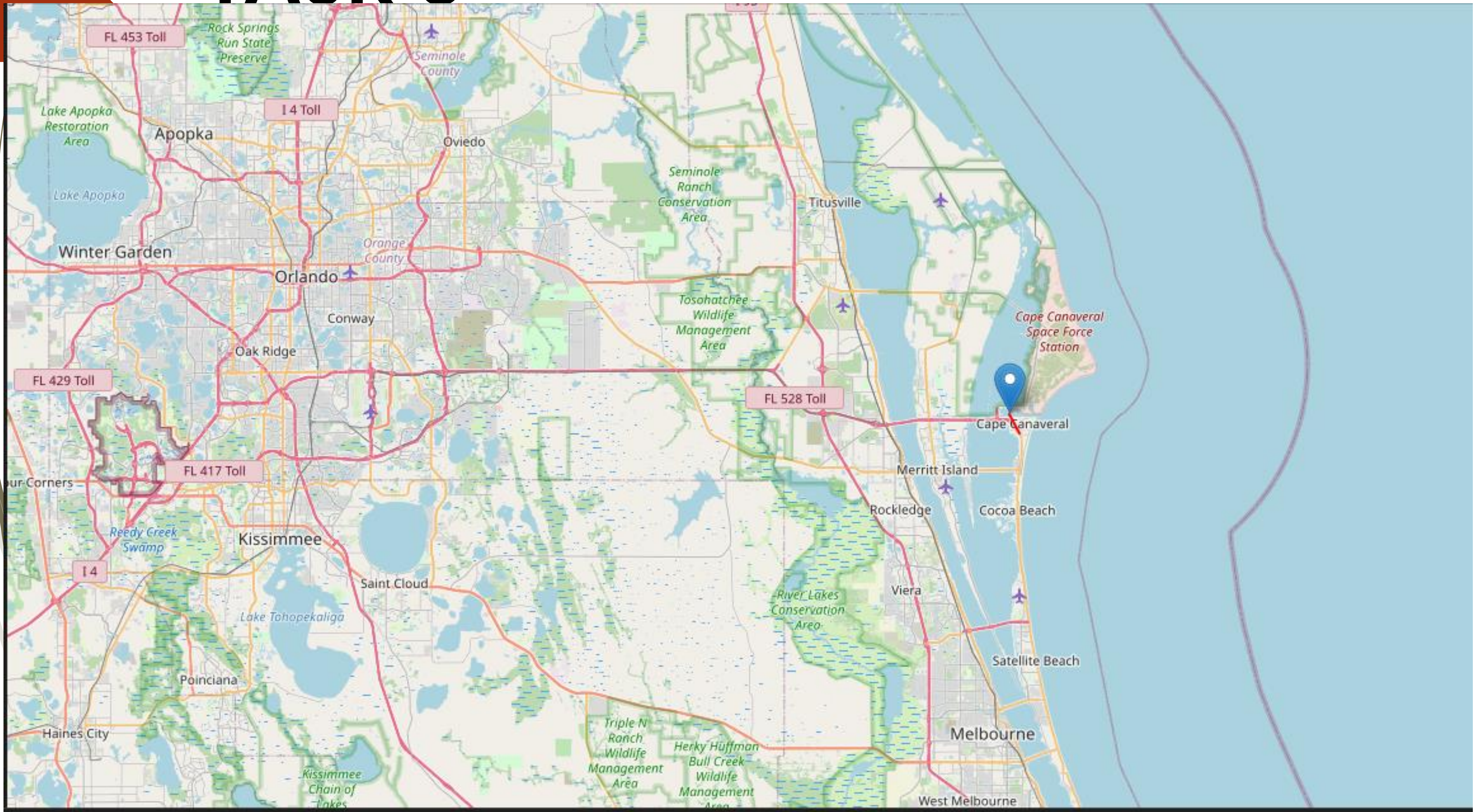


Salt Lake City
Nebraska
Lincoln
Iowa
Chicago
Pennsylvania
Ohio
New Y
Philadelphia
Denver
Utah
Colorado
Kansas City
Illinois
Indiana
West Virginia
Washington
Sacramento
Nevada
Kansas
Missouri
Kentucky
Virginia
San Jose
California
Richmond
Fresno
Las Vegas
Tulsa
Raleigh
Albuquerque
Memphis
Tennessee
Charlotte
North Carolina
Oklahoma
Arkansas
VAFB
Arizona
New Mexico
SLC
Los Angeles
South Carolina
Atlanta
4E
Phoenix
Dallas
Tijuana
Mexicali
Tucson
Mississippi
Alabama
Georgia
Ciudad Juárez
Texas
Louisiana
Austin
Baja California
Houston
Baton Rouge
Jacksonville
San Antonio
Sonora
Chihuahua
Florida
Coahuila de Zaragoza
Monterrey
Reynosa
Baja California Sur
Saltillo
Tamaulipas
Miami
Nassau
La Paz
Culiacán
Durango
México
The Bahamas

# TASK 2


Oregon
Boise
Wyoming
South Dakota
Lake Huron
Lake Michigan
Milwaukee
Toronto
Hamilton
New York
Boston
Detroit
Chicago
Iowa
Nebraska
Lincoln
Salt Lake City
Pennsylvania
Ohio
New York
Philadelphia
Denver
Utah
Colorado
Kansas City
Illinois
Indiana
Sacramento
Nevada
Kansas
Missouri
West Virginia
Washington
Virginia
Kentucky
Richmond
San Jose
California
Fresno
Las Vegas
Tulsa
Raleigh
Albuquerque
Oklahoma
Memphis
Tennessee
Charlotte
North Carolina
10
Los Angeles
Arizona
New Mexico
Arkansas
Atlanta
South Carolina
Phoenix
Tijuana
Mexicali
Tucson
Dallas
Mississippi
Alabama
Georgia
Ensenada
Ciudad Juárez
Texas
Louisiana
Austin
Houston
Baton Rouge
Jacksonville
Baja California
Sonora
San Antonio
46
Chihuahua
Florida
Coahuila
Nuevo Laredo
Torreón
Monterrey
Reynosa
Miami
Nassau
Baja California Sur
Saltillo
Tamaulipas
The Bahamas
La Paz
Culiacán
Durango
Durango
México
La Habana
San Luis
Santa Clara

# TASK 3



FL 453 Toll
Rock Springs Run State Preserve
Seminole County
I 4 Toll
Lake Apopka Restoration Area
Apopka
Oviedo
Lake Apopka
Seminole Ranch Conservation Area
Titusville
Winter Garden
Orange County
Orlando
Tosohatchee Wildlife Management Area
Conway
Cape Canaveral Space Force Station
Oak Ridge
FL 429 Toll
FL 528 Toll
Cape Canaveral
FL 417 Toll
Merritt Island
Four Corners
Rockledge
Cocoa Beach
Reedy Creek Swamp
Kissimmee
I 4
Saint Cloud
Viera
River Lakes Conservation Area
Lake Tohopekaliga
Satellite Beach
Poinciana
Haines City
Triple N Ranch Wildlife Management Area
Herky Huffman Bull Creek Wildlife Management Area
Melbourne
Kissimmee Chain of Lakes
West Melbourne

# Predictive analysis section

# Predictive analysis

| | Type | Distance (km) | Duration | Average Pace | Average Speed (km/h) | Climb (m) | Average Heart Rate (bpm) |
|---|---|---|---|---|---|---|---|
| **Date** | | | | | | | |
| **2018-11-11 14:05:12** | Running | 10.44 | 58:40 | 5:37 | 10.68 | 130 | 159.0 |
| **2018-11-09 15:02:35** | Running | 12.84 | 1:14:12 | 5:47 | 10.39 | 168 | 159.0 |
| **2018-11-04 16:05:00** | Running | 13.01 | 1:15:16 | 5:47 | 10.37 | 171 | 155.0 |
| **2018-11-01 14:03:58** | Running | 12.98 | 1:14:25 | 5:44 | 10.47 | 169 | 158.0 |
| **2018-10-27 17:01:36** | Running | 13.02 | 1:12:50 | 5:36 | 10.73 | 170 | 154.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **2012-09-08 08:35:02** | Running | 3.27 | 15:55 | 4:52 | 12.32 | 15 | 144.0 |
| **2012-09-04 19:12:17** | Running | 6.26 | 32:35 | 5:12 | 11.53 | 34 | 144.0 |
| **2012-09-02 08:41:31** | Running | 3.14 | 16:16 | 5:11 | 11.56 | 18 | 144.0 |
| **2012-08-24 08:13:12** | Running | 3.15 | 16:00 | 5:05 | 11.82 | 17 | 144.0 |
| **2012-08-22 18:53:54** | Running | 5.69 | 31:08 | 5:29 | 10.95 | 32 | 144.0 |

459 rows × 7 columns

| | Activity Id | Type | Route Name |
|---|---|---|---|
| count | 508 | 508 | 1 |
| unique | 508 | 4 | 1 |
| top | c9627fed-14ac-47a2-bed3-2a2630c63c15 | Running | Two roads - Irpin |
| freq | 1 | 459 | 1 |
| mean | NaN | NaN | NaN |
| std | NaN | NaN | NaN |
| min | NaN | NaN | NaN |
| 25% | NaN | NaN | NaN |
| 50% | NaN | NaN | NaN |
| 75% | NaN | NaN | NaN |
| max | NaN | NaN | NaN |

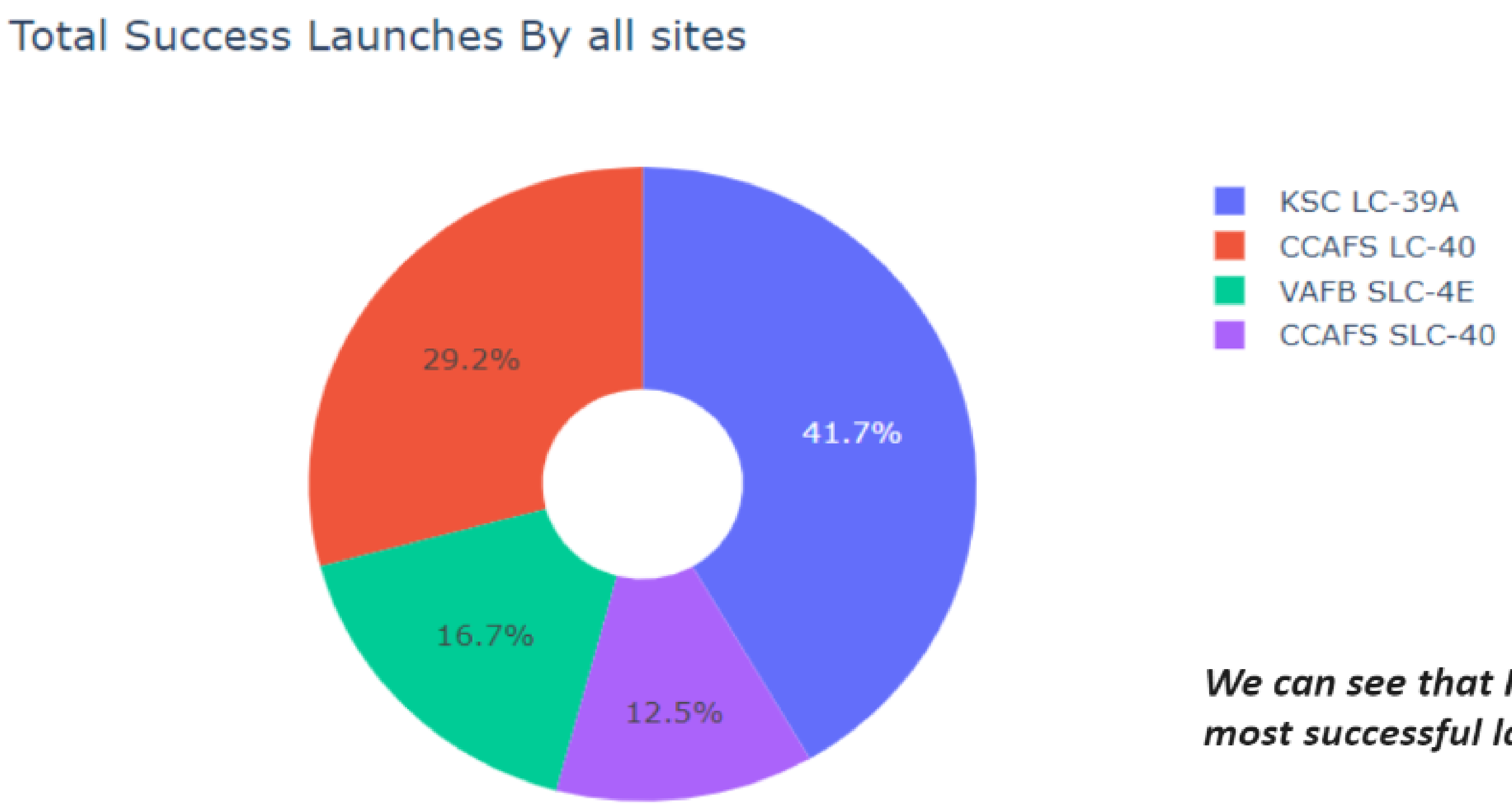
| | Distance (km) | Duration | Average Pace | Average Speed (km/h) |
|---|---|---|---|---|
| count | 508.000000 | 508 | 508 | 508.000000 |
| unique | NaN | 458 | 146 | NaN |
| top | NaN | 32:00 | 5:24 | NaN |
| freq | NaN | 6 | 16 | NaN |
| mean | 11.757835 | NaN | NaN | 11.341654 |
| std | 6.209219 | NaN | NaN | 2.510516 |
| min | 0.760000 | NaN | NaN | 1.040000 |
| 25% | 7.015000 | NaN | NaN | 10.470000 |
| 50% | 11.460000 | NaN | NaN | 11.030000 |
| 75% | 13.642500 | NaN | NaN | 11.642500 |
| max | 49.180000 | NaN | NaN | 24.330000 |

| | Calories Burned | Climb (m) | Average Heart Rate (bpm) | Friend's Tagged |
|---|---|---|---|---|
| count | 5.080000e+02 | 508.00000 | 294.000000 | 0.0 |
| unique | NaN | NaN | NaN | NaN |
| top | NaN | NaN | NaN | NaN |
| freq | NaN | NaN | NaN | NaN |
| mean | 1.878197e+04 | 128.00000 | 143.530612 | NaN |
| std | 2.186930e+05 | 108.52604 | 10.583848 | NaN |
| min | 4.000000e+01 | 0.00000 | 77.000000 | NaN |
| 25% | 4.917500e+02 | 53.00000 | 140.000000 | NaN |
| 50% | 7.280884e+02 | 92.00000 | 144.000000 | NaN |
| 75% | 9.212500e+02 | 172.25000 | 149.000000 | NaN |
| max | 4.072685e+06 | 982.00000 | 172.000000 | NaN |

| | Notes | GPX File |
|---|---|---|
| count | 231 | 504 |
| unique | 7 | 504 |
| top | TomTom MySports Watch | 2018-11-11-140512.gpx |
| freq | 225 | 1 |
| mean | NaN | NaN |
| std | NaN | NaN |
| min | NaN | NaN |
| 25% | NaN | NaN |
| 50% | NaN | NaN |
| 75% | NaN | NaN |
| max | NaN | NaN |

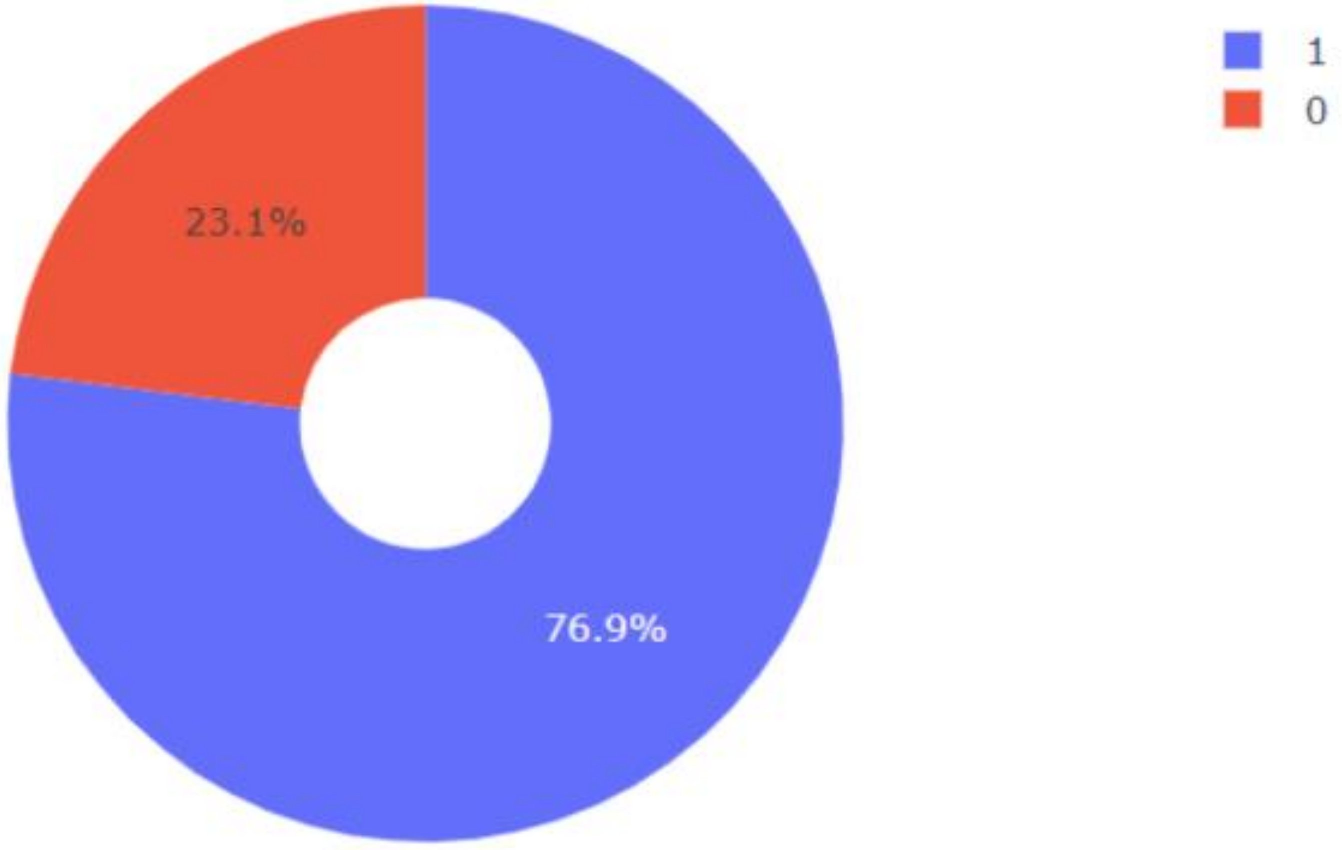
# PLOTLY DASHBOARD section

# DASHBOARD PIECHART



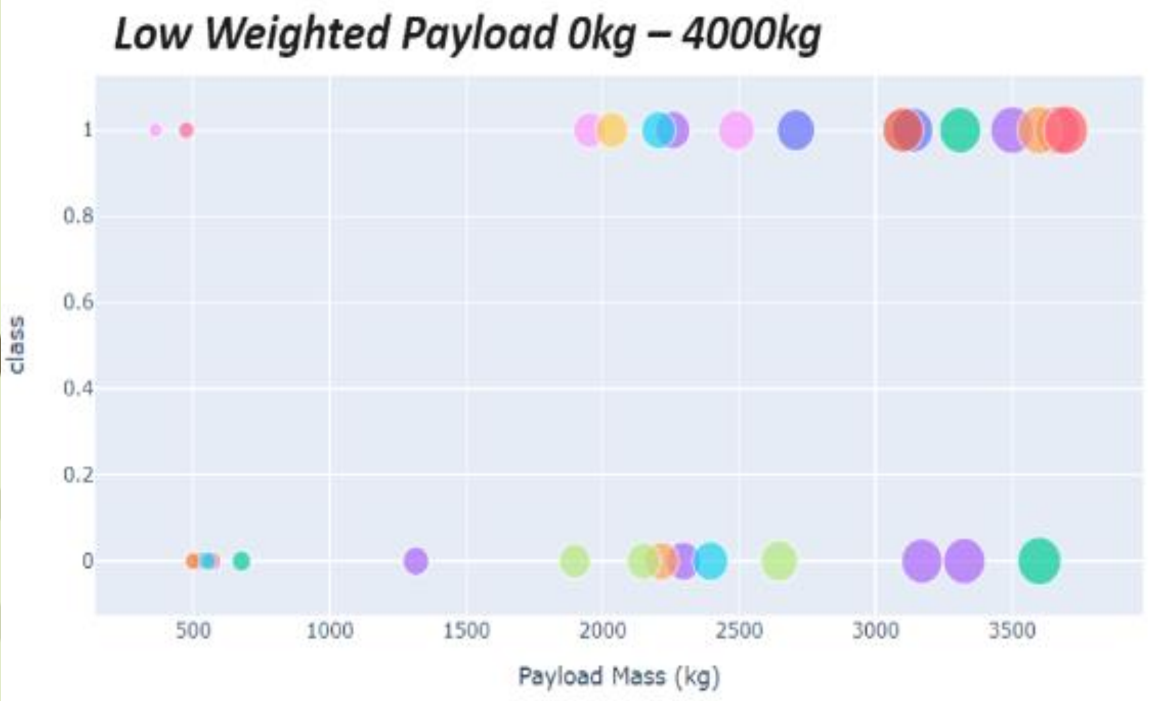

*We can see that KSC LC-39A had the most successful launches from all the sites*

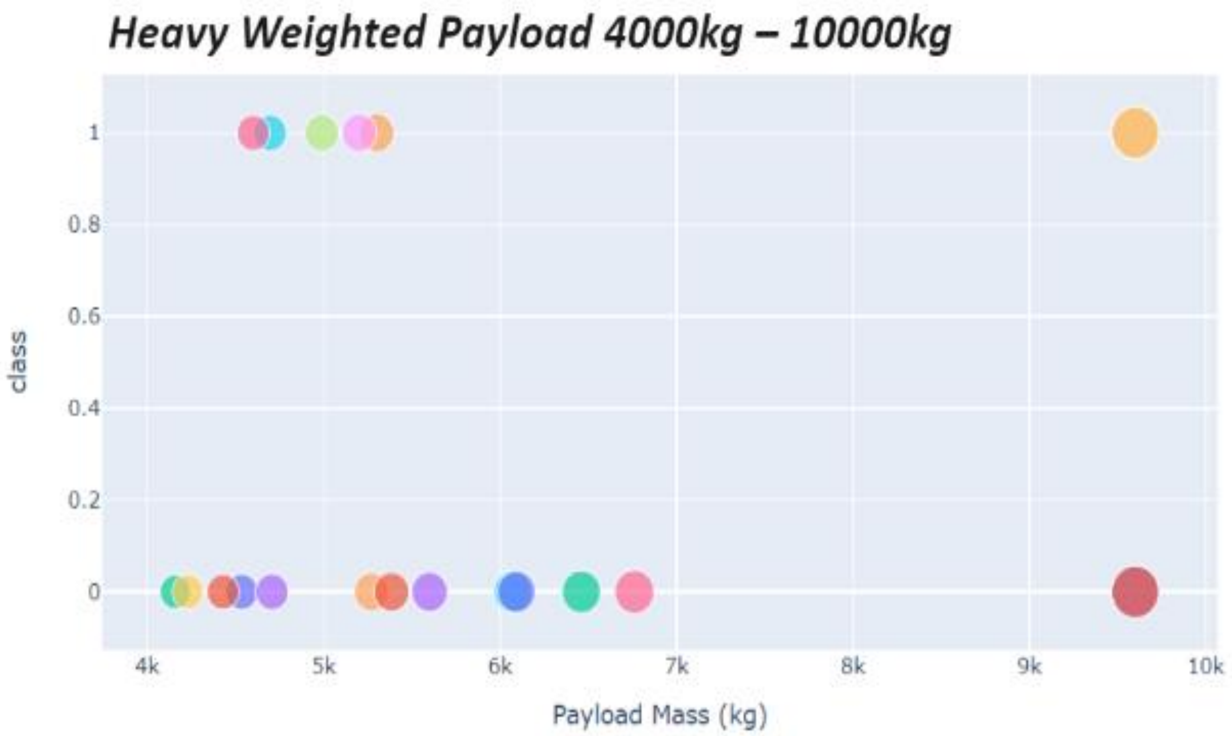
# DASHBOARD PIECHART




***KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate***

# DASHBOARD PIECHART




*We can see the success rates for low weighted payloads is higher than the heavy weighted payloads*

# **MACHINE LEARNING MODEL section**

# DESIGN TREE MODEL CALCULATION

```
parameters = {'criterion': ['gini', 'entropy'],
      'splitter': ['best', 'random'],
      'max_depth': [2*n for n in range(1,10)],
      'max_features': ['auto', 'sqrt'],
      'min_samples_leaf': [1, 2, 4],
      'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

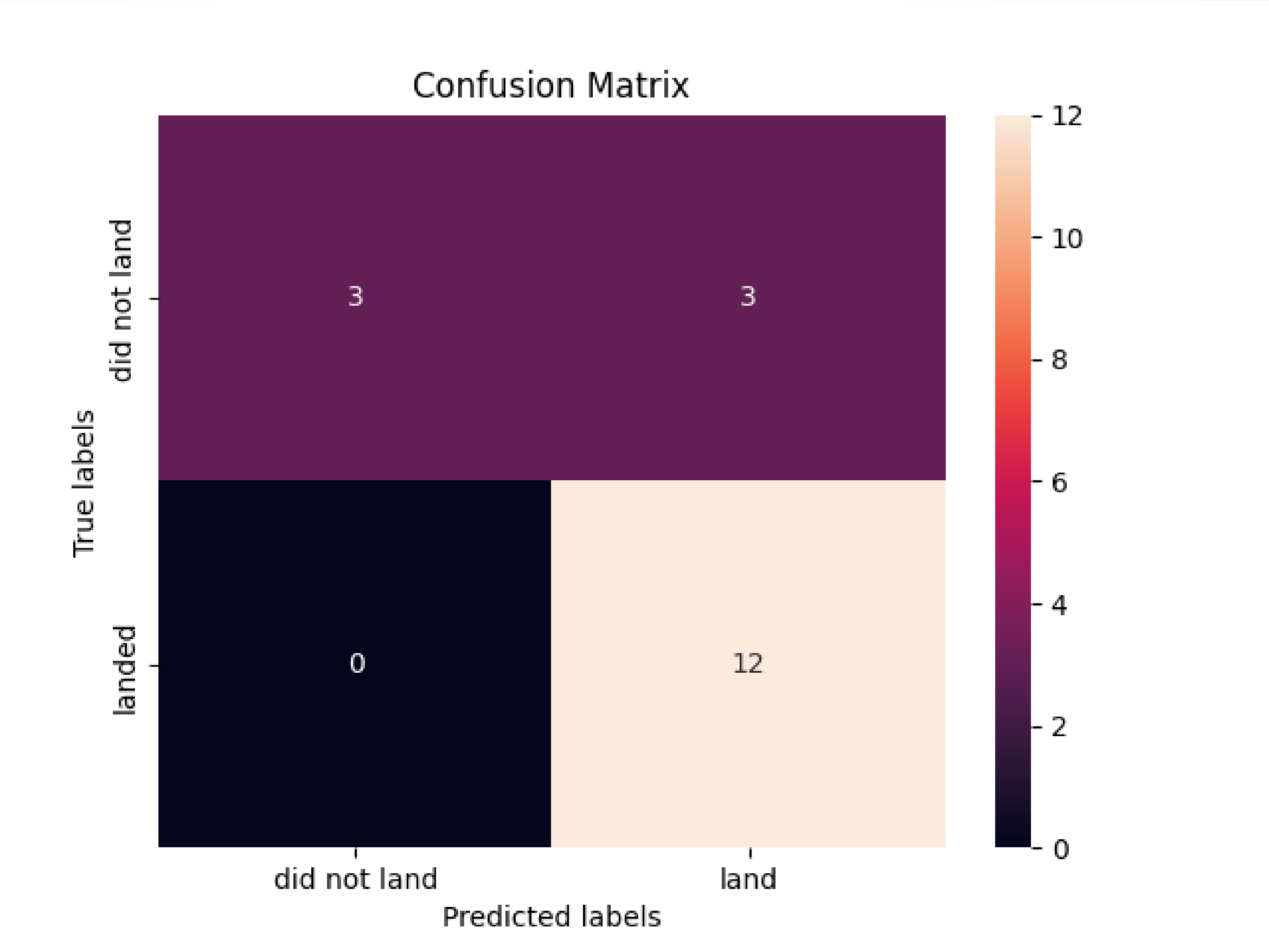
```
tree_cv = GridSearchCV(tree, parameters, cv=10)
tree_cv.fit(X_train, Y_train)
```

```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
accuracy : 0.8892857142857145
```

# CONFUSION MATRIX



# RESULT

```
accuracy_lr = logreg_cv.score(X_test, Y_test)
accuracy_svm = svm_cv.score(X_test, Y_test)
accuracy_tree = tree_cv.score(X_test, Y_test)
accuracy_knn = knn_cv.score(X_test, Y_test)

# Find the method with the highest accuracy
best_accuracy = max(accuracy_lr, accuracy_svm, accuracy_tree, accuracy_knn)

# Determine the best performing method
if best_accuracy == accuracy_lr:
    best_method = "Logistic Regression"
elif best_accuracy == accuracy_svm:
    best_method = "Support Vector Machine (SVM)"
elif best_accuracy == accuracy_tree:
    best_method = "Decision Tree Classifier"
else:
    best_method = "K Nearest Neighbors (KNN)"

# Print the best performing method
print("The best performing method is:", best_method)
```

```
The best performing method is: Logistic Regression
```