IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Halyna Zhdan
27.07.2024

Outline

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

Executive Summary

- Summary of methodologies
    - i. Data understanding
    - ii. Data collection with Web Scraping
    - iii. Data Wrangling
    - iv. Using SQL to analyze data
    - v. Exploring data with visuals and building analytics with Folium
    - vi. Building ML model

- Summary of all results
    - i. Data Analysis results
    - ii. Data Visualizations results
    - iii. Machine Learning predictive 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. We need to collect a data and build a ML model that predicts if SpaceX can reuse the first stage and if launch will land successfully.

- Problems you want to find answers

  ✓ What indicators must exist to land successfully

  ✓ How to eliminate known issues

Section 1

Methodology

n this lab we will mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using web scraping and SpaceX REST API
- Perform data wrangling
  - We encoded data to 0 and 1 , 0 landing was unsuccessful and 1 successful
- 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

- We used below techniques to collect and clean data
  - ✓ Data was collected using SpaceX REST API with get request
  - ✓ Then we used json() and .json_normalize() to convert to dataframe
  - ✓ Once in df data was cleaned
  - ✓ Also we used beautifulsoap package for web scraping to get data from wikipedia

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

- Add the GitHub URL of the completed SpaceX API calls notebook [IBM-SpaceX/01-jupyter-labs-spacex-data-collection-api.ipynb at main · halynazhdan/IBM-SpaceX (github.com)]() as an external reference and peer-review purpose

```
In [6]:  spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
In [7]:  response = requests.get(spacex_url)
```

2. Use json_normalize method to convert json result to dataframe

```
In [12]: # Use json_normalize method to convert the json result into a dataframe

         # decode response content as json
         static_json_df = res.json()
```

```
In [13]: # apply json_normalize
         data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

```
In [30]: rows = data_falcon9['PayloadMass'].values.tolist()[0]

         df_rows = pd.DataFrame(rows)
         df_rows = df_rows.replace(np.nan, PayloadMass)

         data_falcon9['PayloadMass'][0] = df_rows.values
         data_falcon9
```

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts

- [IBM-SpaceX/02-jupyter-labs-webscraping.ipynb at main · halynazhdan/IBM-SpaceX (github.com)]

1. Apply HTTP Get method to request the Falcon 9 rocket launch page

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

Out[5]:
```
200
```

2. Create a BeautifulSoup object from the HTML response

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

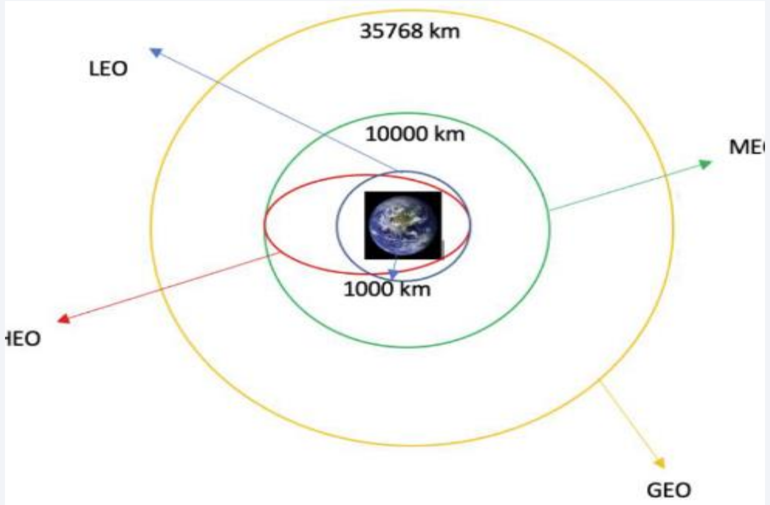
3. Extract all column names from the HTML table header

In [10]:
```
column_names = []

# Apply find_all() function with 'th' element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name ('if name is not None and len(name) > 0') into a list called column_names

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

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

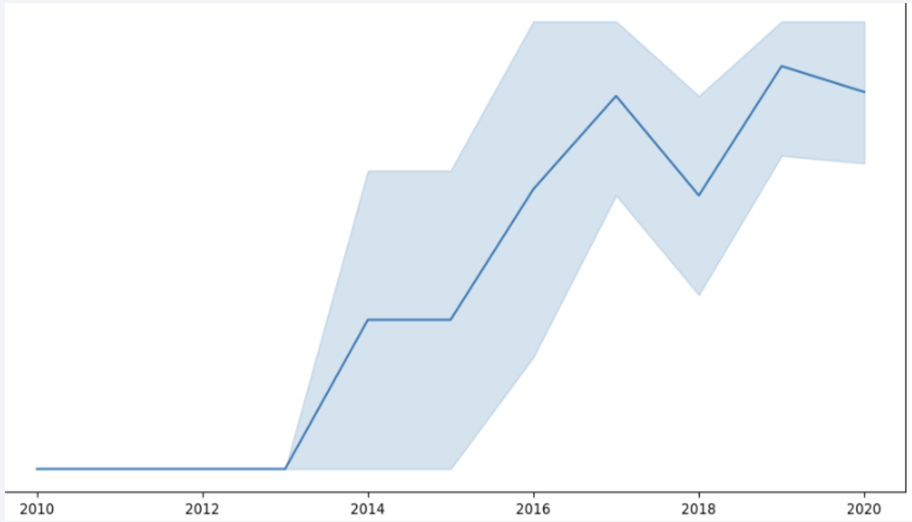
Data Wrangling

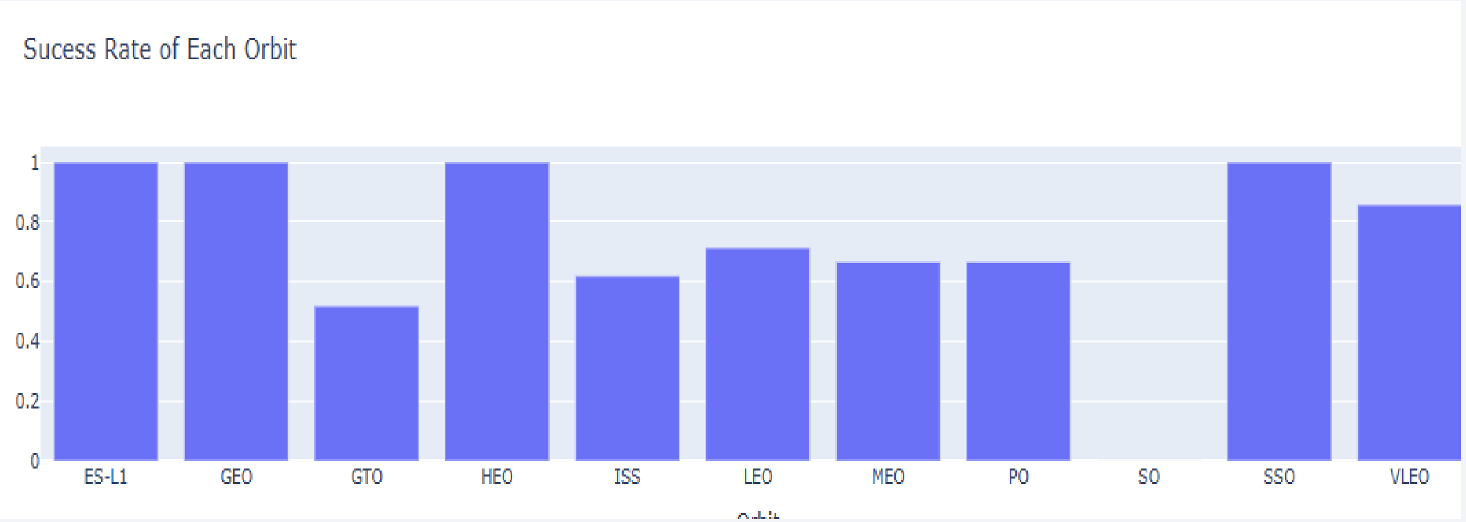
- Describe how data were processed
  - We did exploratory data analysis and defined training labels, encoded data to 0 and 1 , 0 landing was unsuccessful and 1 successful, after exported to csv
- [IBM-SpaceX/03-labs-jupyter-spacex-Data wrangling (1).ipynb at main · halynazhdan/IBM-SpaceX (github.com)]



EDA with Data Visualization

- Summarize what charts were plotted and why you used those charts
- We built dashboards between flight # and launch site, success rate, flight number and orbit and success trend by year
- [IBM-SpaceX/04-edadataviz.ipynb at main · halynazhdan/IBM-SpaceX (github.com)]()





EDA with SQL

- Below SQL queries have been built and executed:
  - Unique Sites names in the space mission
  - We calculated the total payload mass carries by boosters
  - We calculated the average payload mass carried by booster
  - The total of success and failure outcomes
  - We calculated the failed landing outcome , booster version and site name
- [IBM-SpaceX/05-jupyter-labs-eda-sql-coursera_sqllite (1).ipynb at main · halynazhdan/IBM-SpaceX (github.com)]

Build an Interactive Map with Folium

- We added objects :markers, circles, lines to mark launches of success or failure on the folium map
- 0 was assigned for failure and 1 for success, use color marker to check which site was successful and which failed
- IBM-SpaceX/06-lab_jupyter_launch_site_location.ipynb at main · halynazhdan/IBM-SpaceX (github.com)

Build a Dashboard with Plotly Dash

- Interactive dashboard was created using Plotly
- We built pie charts total values by sites
- Plot scatter with Outcome, Payload for booster's version
- IBM-SpaceX/07-spacex_dash_app_copy (1).py at main · halynazhdan/IBM-SpaceX (github.com)

Predictive Analysis (Classification)

- Data was loaded with numpy and pandas, transformed
- After transformed it was split into test and train
- Used GridSearchCV to build ML
- Applied accuracy for the model, improved the model and developed the classification ML with the best performance
- IBM-SpaceX/08-ML-Predict.ipynb at main · halynazhdan/IBM-SpaceX (github.com)

Results

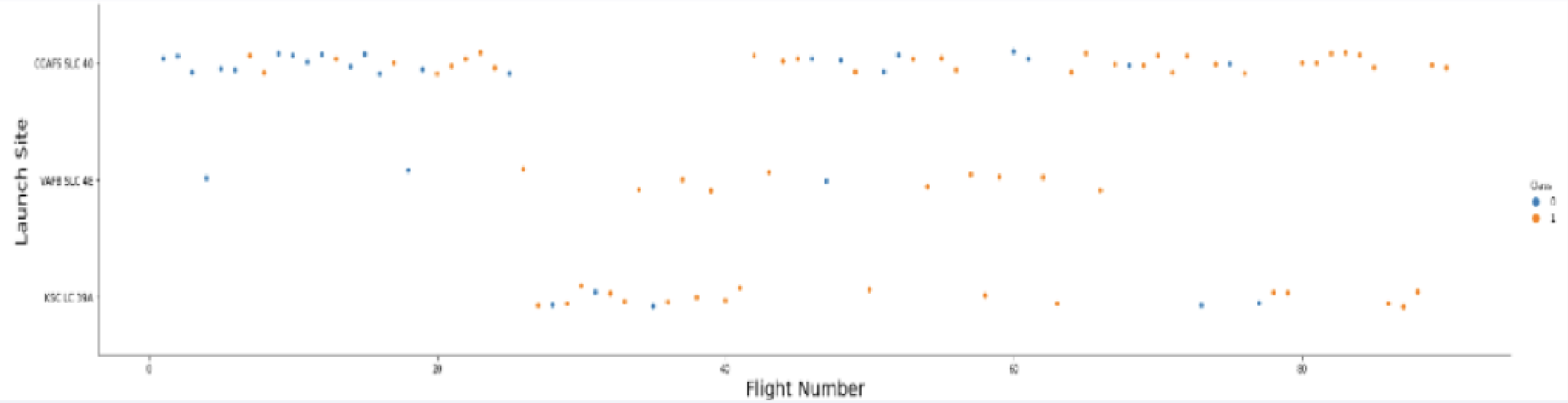
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

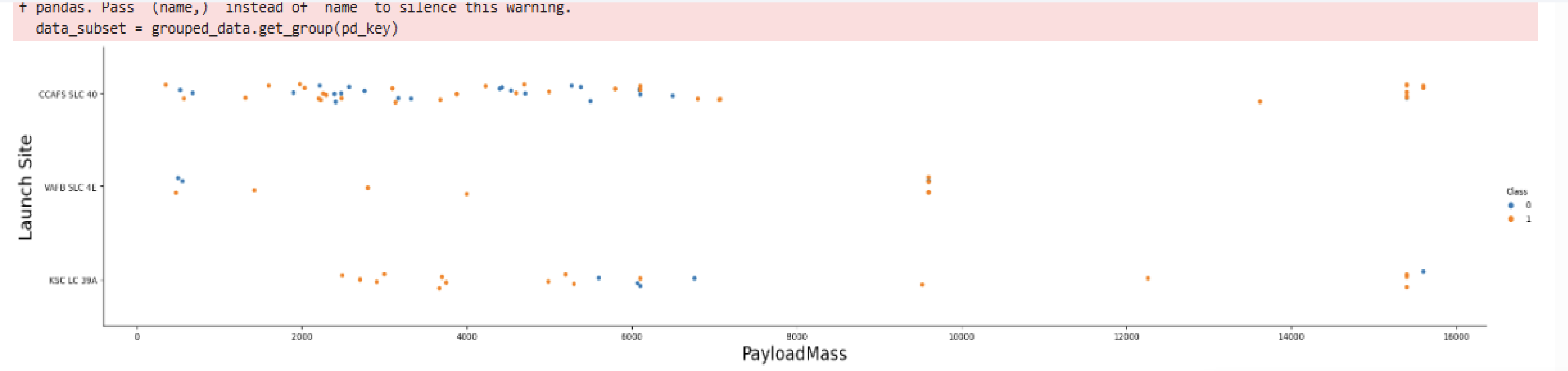
Flight Number vs. Launch Site

- As per below plot flight amount is correlated to success rate, larger flight-> more success rate



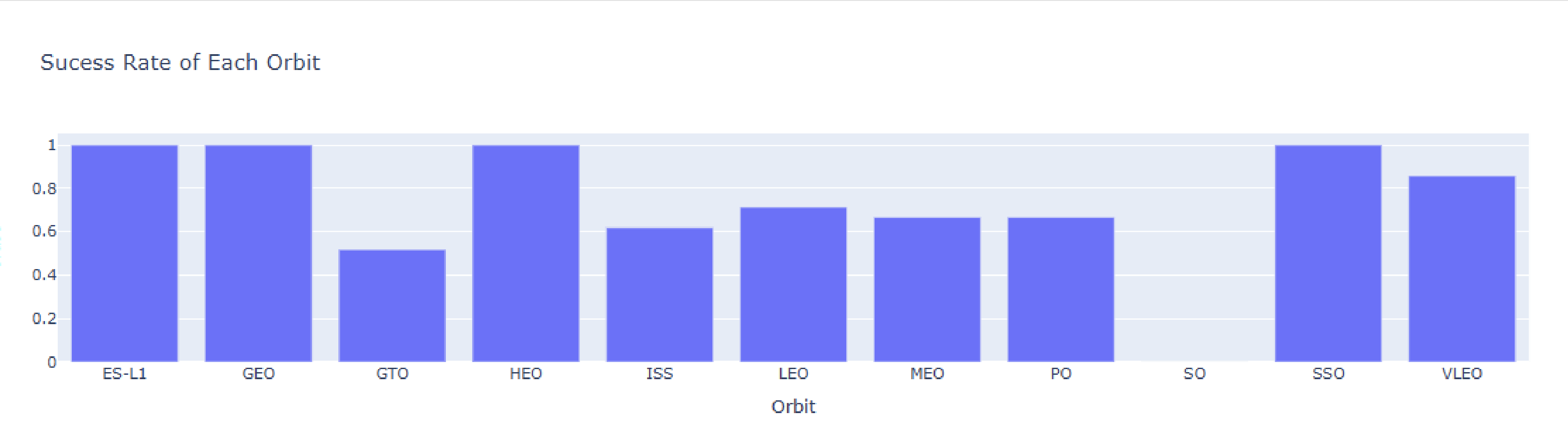
Payload vs. Launch Site

- Greater payload the higher success rate



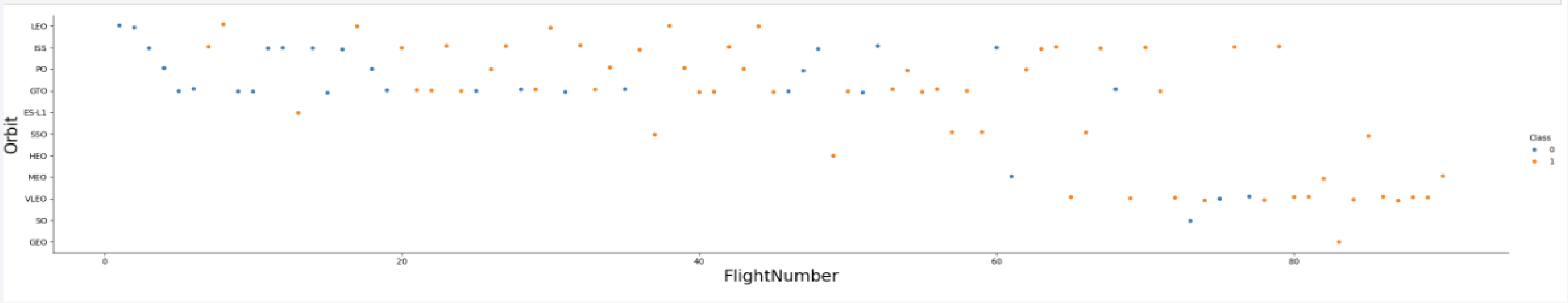
Success Rate vs. Orbit Type

- ES-L1,HEO,GEO and SSO having the greatest success rate



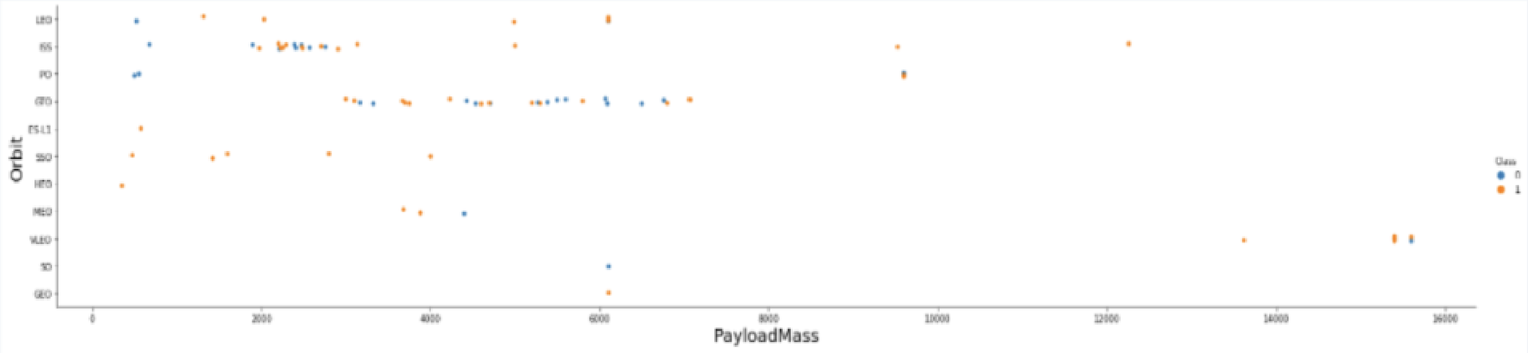
Flight Number vs. Orbit Type

- If Orbit=LEO, success is related to # of flights.
- If Orbit=GTO, no correlation



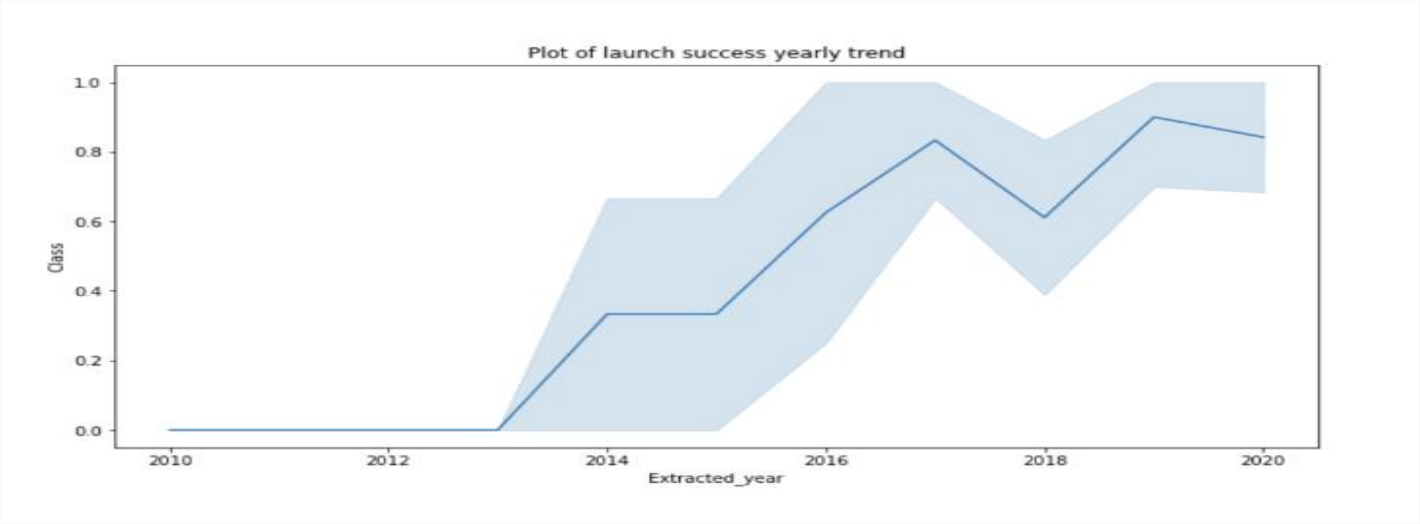
Payload vs. Orbit Type

- Orbit=PO,LEO,IIS having more success



Launch Success Yearly Trend

- Starting from 2013 success goes up



All Launch Site Names

- Used distinct

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

```
[12]: %sql select distinct(LAUNCH_SITE)from SPACEXTBL;
```

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

[12]:

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

Launch Site Names Begin with 'CCA'

- We used below query

Task 2

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

```
[15]: %sql select * from SPACEXTBL where Launch_Site like'CCA%' limit 5;
```

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

[15]:

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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 used below query

Task 3

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

[16]:
```
%sql select sum (PAYLOAD_MASS__KG_) from SPACEXTBL where customer = 'NASA (CRS)';
```

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

[16]:

sum (PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- We used below query:

Task 4

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

```
[17]: %sql select avg (PAYLOAD_MASS__KG_) from SPACEXTBL where BOOSTER_VERSION='F9 v1.1';
```

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

[17]:

avg (PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- We used below query:

Task 5

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

Hint:Use min function

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

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

min(DATE)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used below query

Task 6

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

```
%sql select *  from SPACEXTBL where Landing_Outcome='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2016-05-06	5:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-08-14	5:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-10-11	22:53:00	F9 FT B1031.2	KSC LC-39A	SES-11 / EchoStar 105	5200	GTO	SES EchoStar	Success	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

- We used query below

Task 7

List the total number of successful and failure mission outcomes

```
%sql select MISSION_OUTCOME, count(MISSION_OUTCOME)from SPACEXTBL Group by MISSION_OUTCOME;
```

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

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

Boosters Carried Maximum Payload

- We used query below

Task 8

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

```
%sql select BOOSTER_VERSION, PAYLOAD_MASS__KG_ from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL order by PAYLOAD_MASS__KG_ desc limit 1);
```

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

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

2015 Launch Records

- We used query below

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, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%sql select * from SPACEXTBL where Landing_Outcome like '%drone%'and substr(Date,0,5) like '2015%' ;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2015-01-10	9:47:00	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)
2015-04-14	20:10:00	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)
2015-06-28	14:21:00	F9 v1.1 B1018	CCAFS LC-40	SpaceX CRS-7	1952	LEO (ISS)	NASA (CRS)	Failure (in flight)	Precluded (drone ship)

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

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 Landing_Outcome,count(*) FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome order by 2 DESC;
```

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

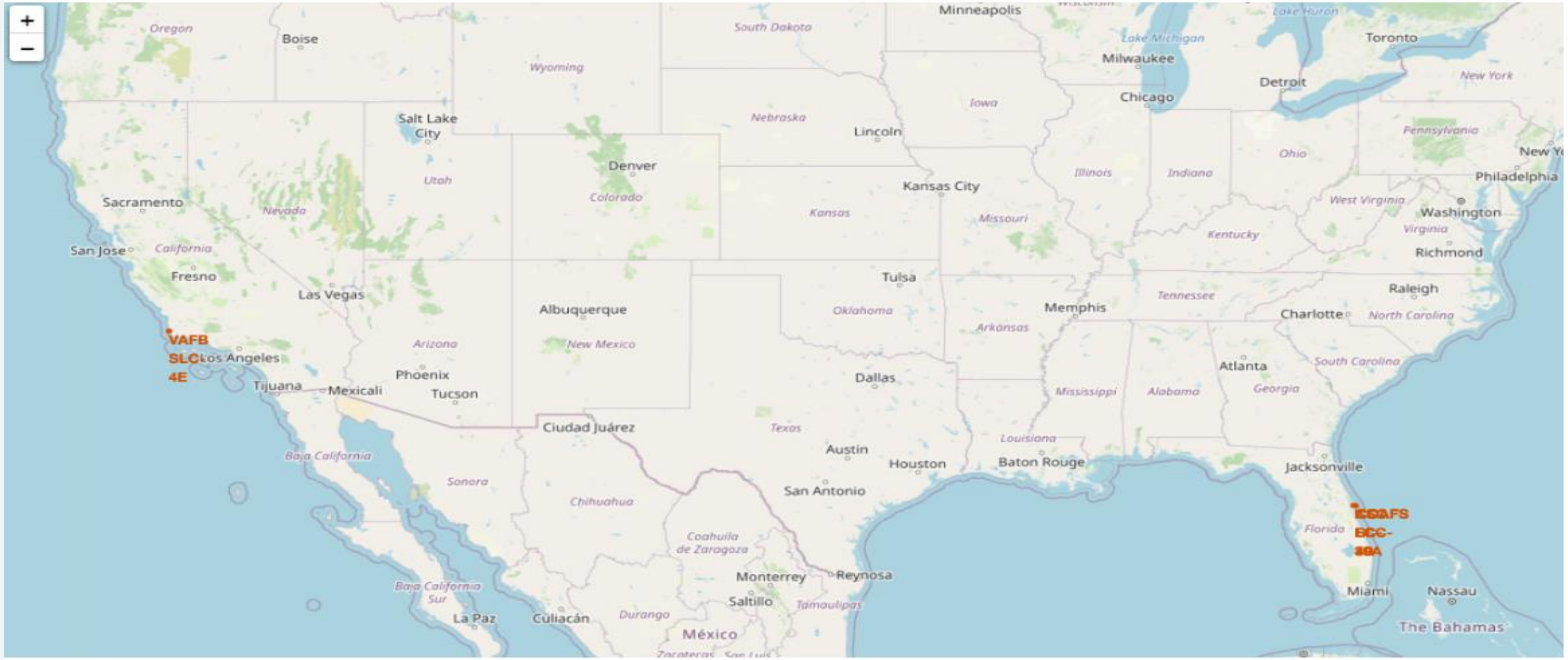
Landing_Outcome	count(*)
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

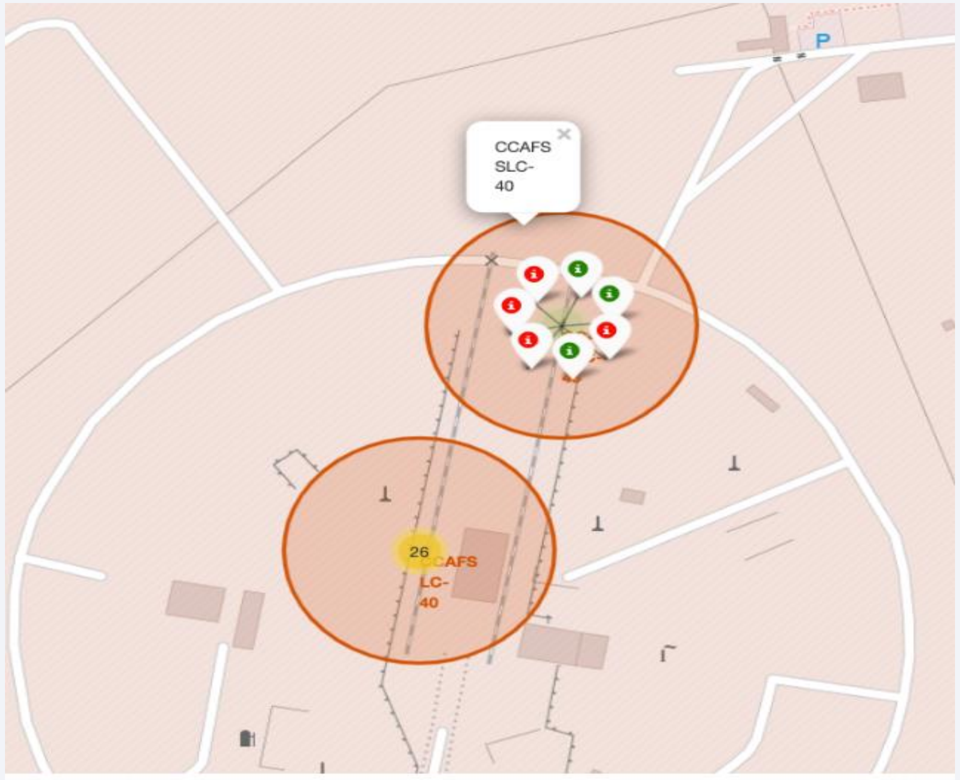
Launching sites map

SpaceX launch sites are located in US: Florida and California



Markers on map

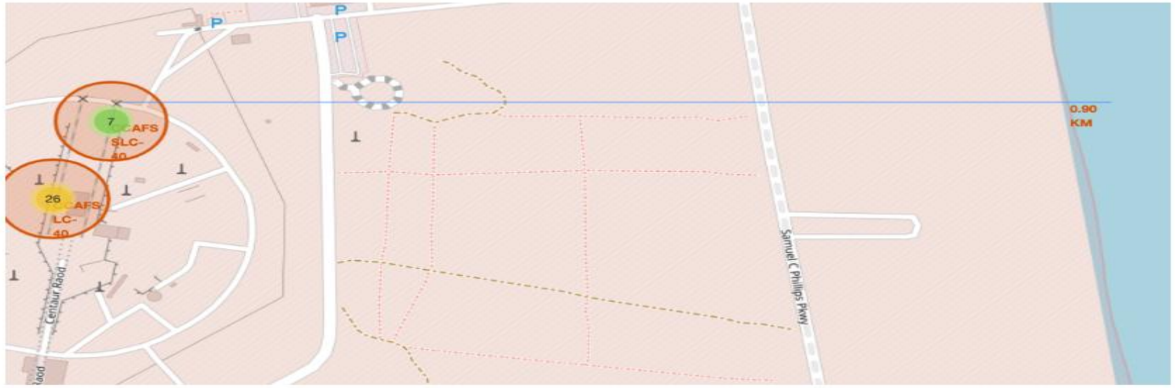
- Green marker is successful launch and red is failure



Distance from Launch site to specific landmarks

Markdown

Pytho



O: Similarly, you can draw a line betwee a launch site to its closest city, railway, highway, etc. You need to use `MousePosition` to find the their coordinates on the map first

lway map symbol may look like this:

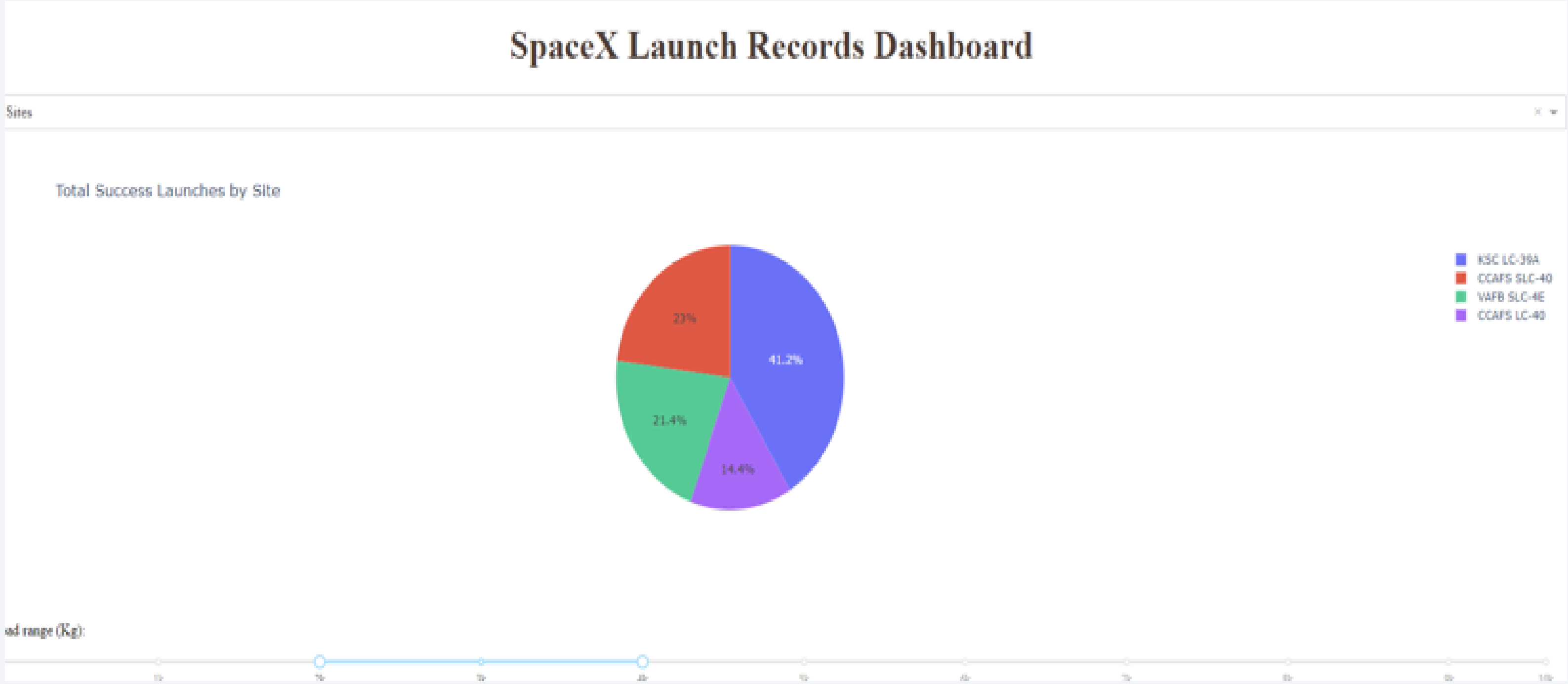
ghway map symbol may look like this:

y map symbol may look like this:

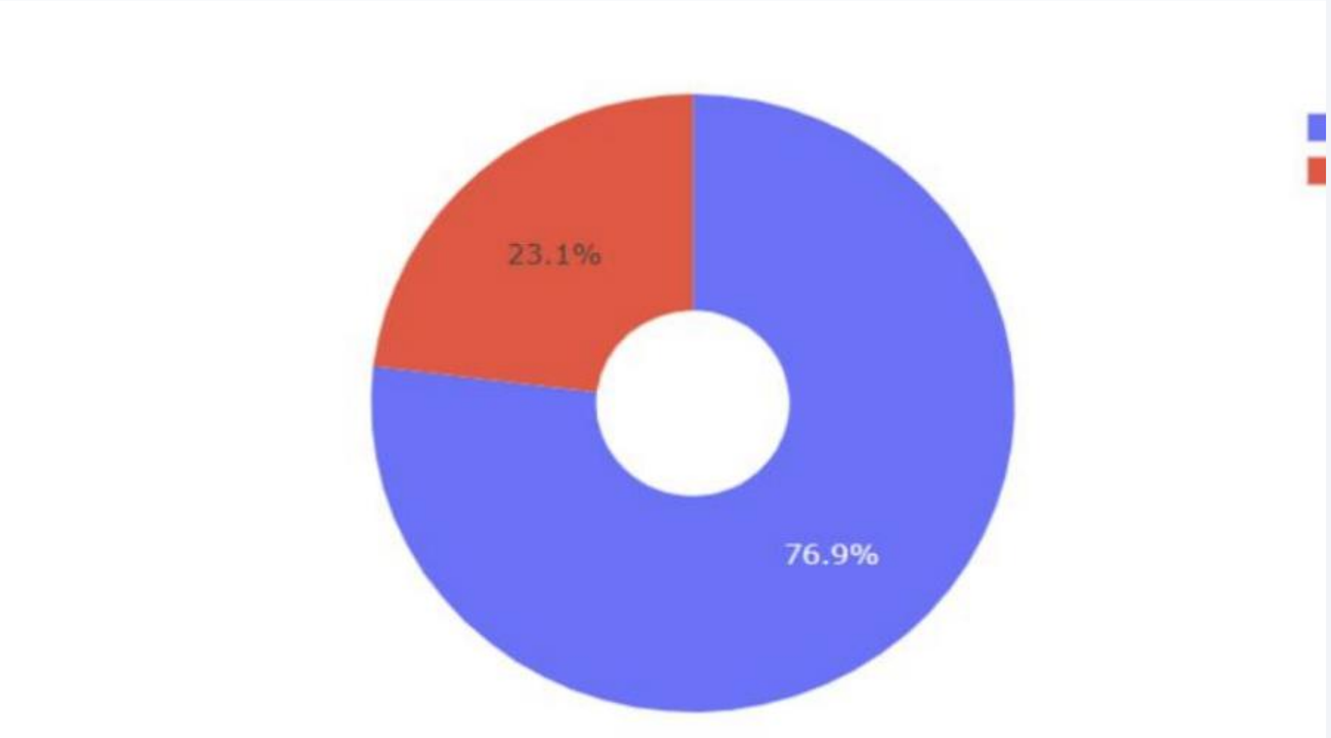
Section 4

Build a Dashboard with Plotly Dash

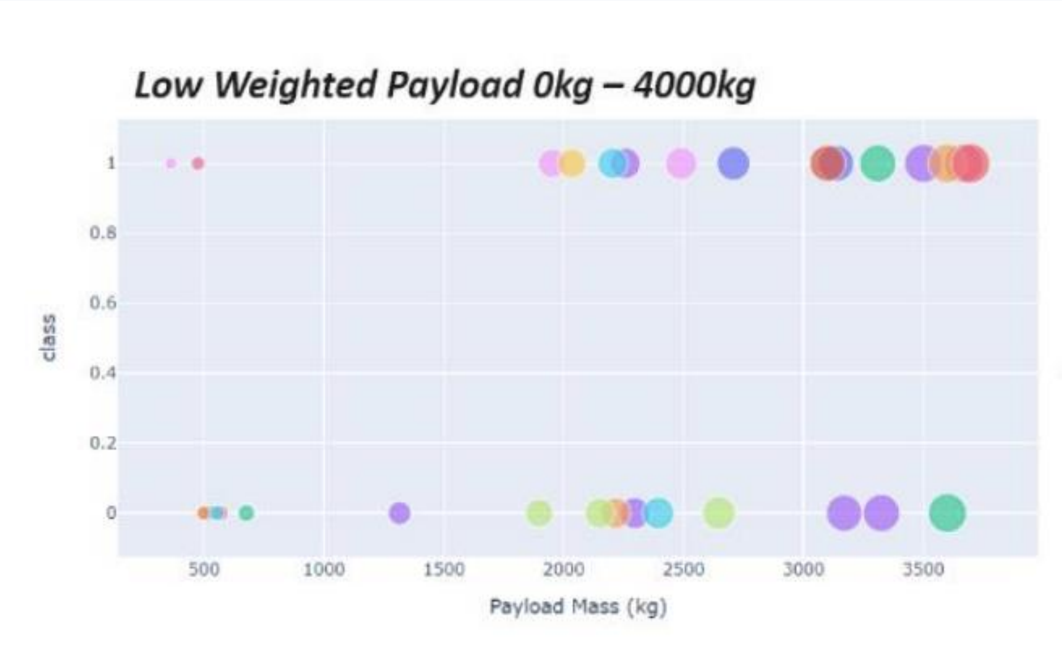
Total success Launches By All sites



Site with highest rate success



Payload vs Launch Outcome plot



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- As per below decision tree has the highest score and decision tree is the best model

TASK 12

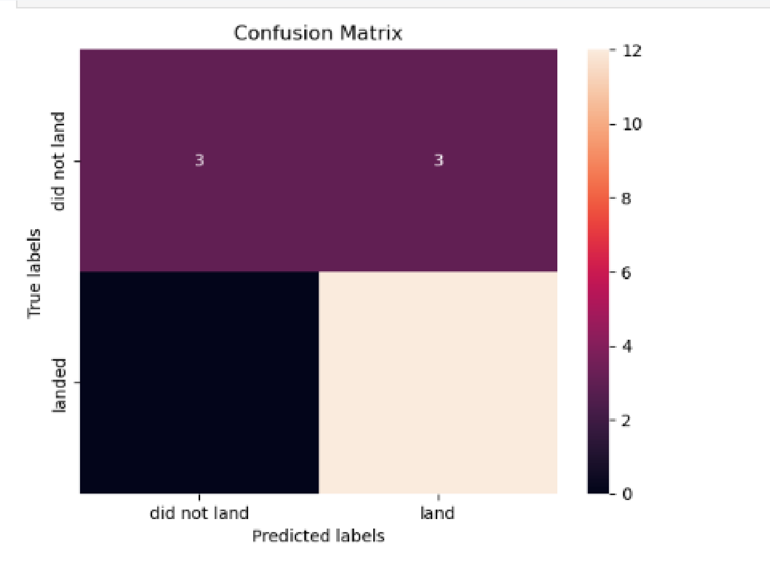
Find the method performs best:

```
[85]: 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 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- It shows performance of classification model.As you can see below it shows the count of true(1) or false(0) predictions.



Conclusions

- As per the analysis we perform we can state that:
- Success rate increase from 2013 and is steady
- The success rate depends on flight amount
- KSC LC-39A is the most successful
- Decision tree is the best ML model for this scenario

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!