
IBM Developer
SKILLS NETWORK
IBM
LVP
Vasile Lucian Popa
February 1st, 2022

Outline

- Executive Summary;
- Introduction;
- Methodology;
- Results;
- Conclusion.

Executive Summary

- *Summary of methodologies:*
  - Data Collection through API;
  - Data Collection with Web Scraping;
  - Data Wrangling;
  - Exploratory Data Analysis with SQL;
  - Exploratory Data Analysis with Data Visualization;
  - Interactive Visual Analytics with Folium;
  - Machine Learning Prediction.
- *Summary of all results:*
  - Exploratory Data Analysis result;
  - Interactive analytics in screenshots;
  - Predictive Analytics result.

Introduction

- **Project background and context**

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

- **Problems you want to find answers:**

  - What key factors determine if the rocket will land successfully or not?
  - The interaction amongst various features that determine the success rate of a successful landing of each rocket;
  - What operating conditions for SpaceX needs to be in place to ensure a successful landing program and get a better success landing rate.

Section 1

METHODOLOGY

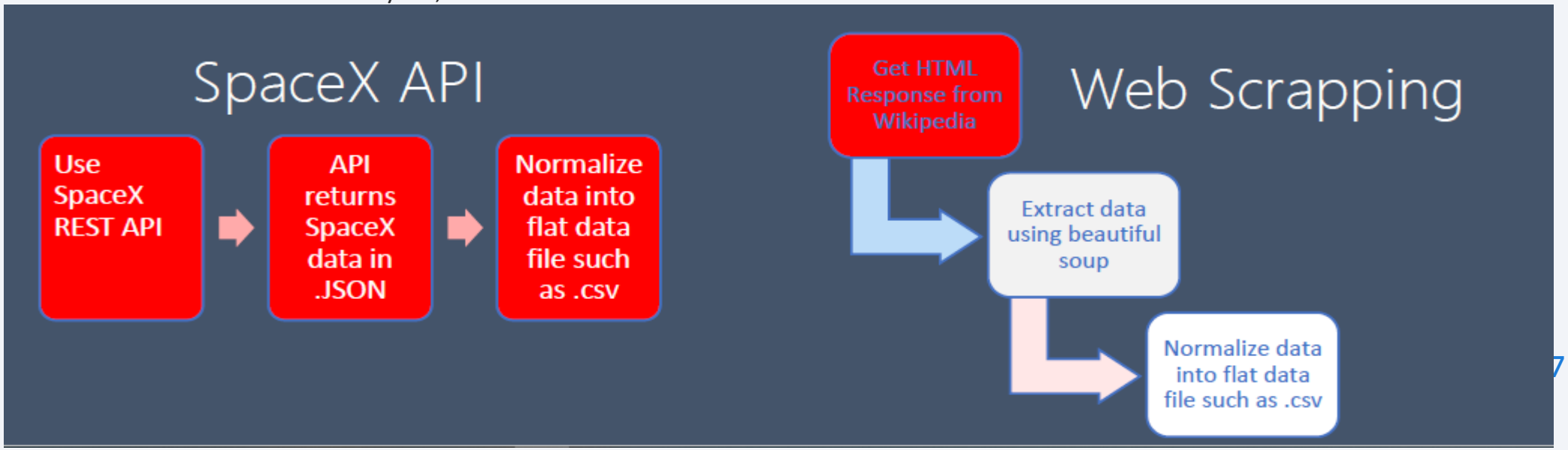
Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using :
  - 1. SpaceX API;
  - 2. 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

- Data collection was gathered using get request to the SpaceX REST 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;



Data Collection – SpaceX API

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

[GitHub URL of the notebook]

Getting Response from the Rest API:

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

Converting response to .json file:

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

Apply functions to get data ready for construction

```
getBoosterVersion(data), getLaunchSite(data),
getPayloadData(data), getCoreData(data)
```

Construct dataset, combine columns and create dataframe

```
launch_dict = {'FlightNumber': list(data['flight_number'])
# Create a data from launch_dict
data_falcon9 = pd.DataFrame(launch_dict)
```

Save new dataset to CSV

```
#Export data to csv
data_falcon9.to_csv('data_falcon9.csv', index=False)
```

Data Collection - Scraping

- I applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup and parsed the table and converted it into a pandas dataframe.

[GitHub URL of the notebook]

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

- [GitHub URL of the notebook]

Process

Perform Exploratory Data Analysis EDA on dataset

Calculate the number of launches at each site

Calculate the number and occurrence of each orbit

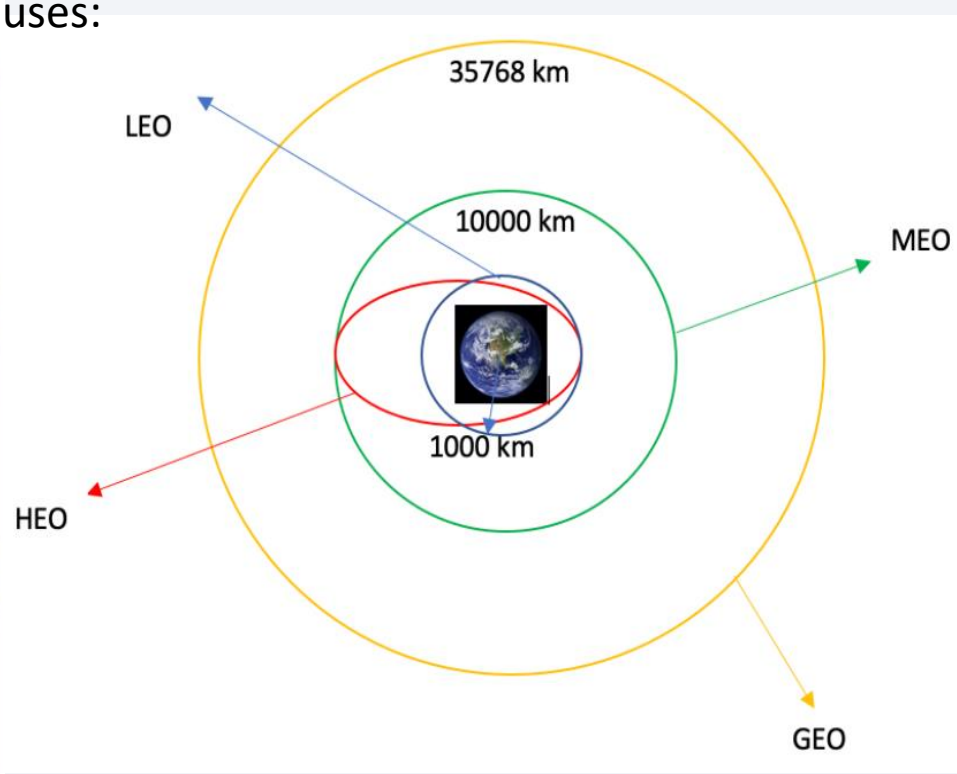
Calculate the number and occurrence of mission outcome per orbit type

Export dataset as .CSV

Create a landing outcome label from Outcome column

Work out success rate for every landing in dataset

Diagram shoeing common orbit types SpaceX uses:



EDA with Data Visualization

Scatter Graphs being drawn:

Flight Number VS. Payload Mass

Flight Number VS. Launch Site

Payload VS. Launch Site

Orbit VS. Flight Number

Payload VS. Orbit Type

Orbit VS. Payload Mass

Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation . Scatter plots usually consist of a large body of data.

Bar Graph being drawn:

Mean VS. Orbit

A bar diagram makes it easy to compare sets of data between different groups at a glance. The graph represents categories on one axis and a discrete value in the other. The goal is to show the relationship between the two axes. Bar charts can also show big changes in data over time.

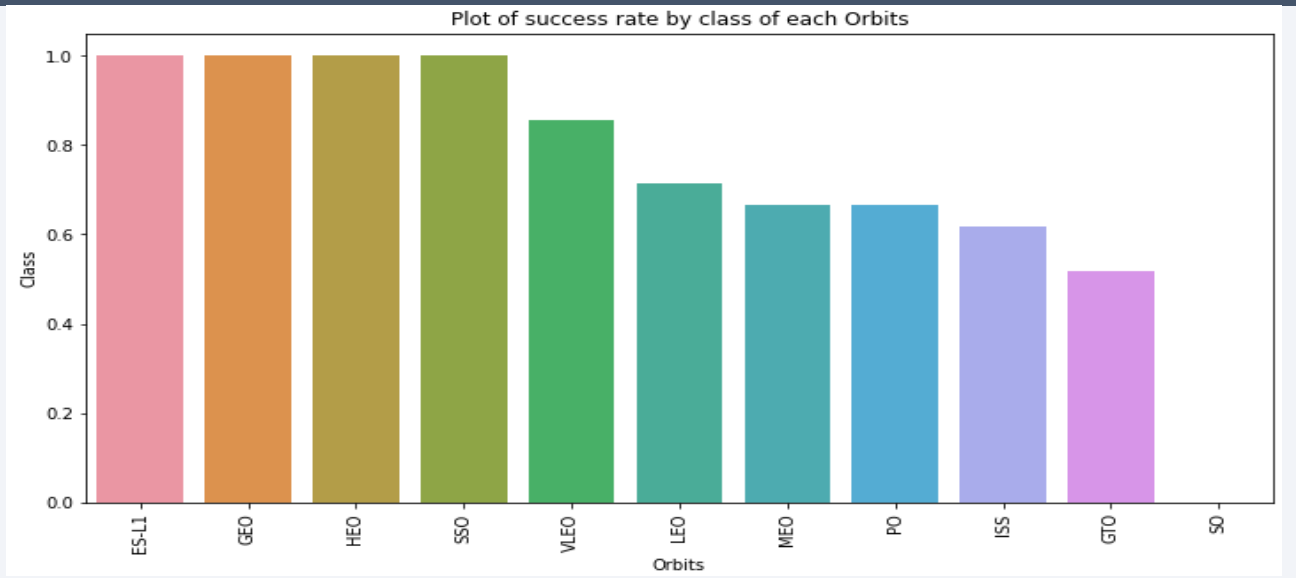
Line Graph being drawn:

Success Rate VS. Year

Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded

LVP

[GitHub link of the notebook]



EDA with SQL

- We loaded the SpaceX dataset into a IBM Db2 database and worked in Watson Studio where added the notebook and worked with the data;
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.

GitHub link of the notebook

Build an Interactive Map with Folium

- Used Python Anywhere to hot the website live so I can consult the data anytime (Flask and Dash web framework were used);
- I marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map;
- I assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success;
- Using the color-labeled marker clusters, I identified which launch sites have relatively high success rate.;
- I calculated the distances between a launch site to its proximities. We answered some question for instance;
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

Build a Dashboard with Plotly Dash

- I have built an interactive dashboard with Plotly dash
- I plotted pie charts showing the total launches by a certain sites (to display the data the following graphs were used: pie chart (to show the total lunches by a certain site and all sites and displaying proportions of multiple classes of data and scatter plots showing the relationship with Outcome and Payload Mass (kg) for the different Booster versions);
- I plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version, so we can see:
- - the relationship between the two variables;
- - finding the best method to show the non-linear pattern within the data;
- - to determine the range of the data flow (i.e.: max and min value).

Github link to Dash app

Predictive Analysis (Classification)

Building the model:

- Load the dataset into NumPy and Pandas;
- Transform data;
- Split the data into training and test data sets;
- Pick up which type of ML algorithm I want to use (used KNN, LinearRegression, SVM and Decision Tree);
- Set parameters and use GridSearchCV to find the best ones;
- Fit data int the GridSearch CV and train on the dataset.

Evaluating the model:

- Check accuracy of the models;
- Tune hyperparameters of each model;
- Plot Confusion Matrix.

Improving the model:

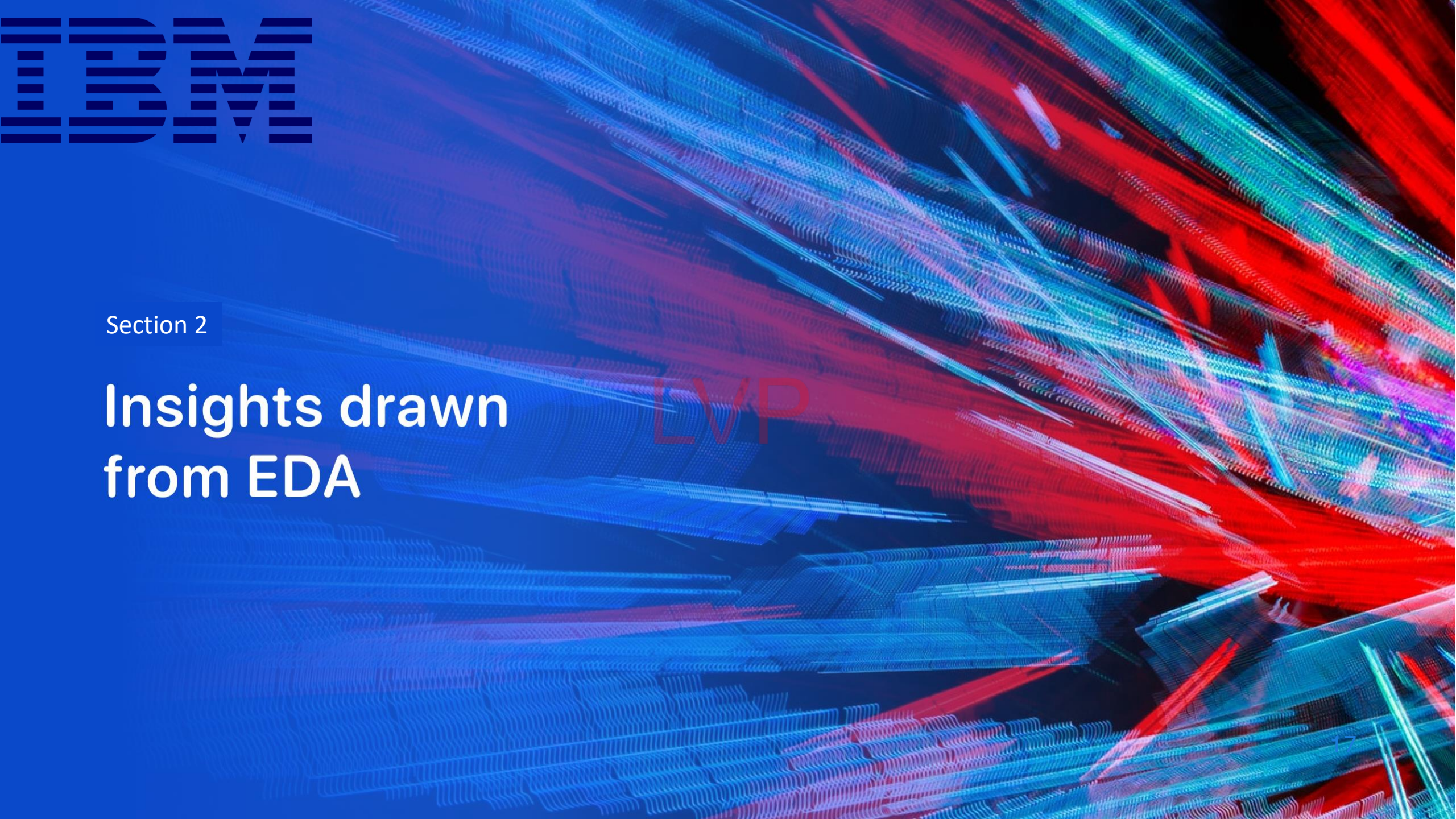
- Feature Engineering;
- Algorithm tuning;

Finding the best model:

- Evaluate the best performing model, on the accuracy.
- Find the scores here: GutHub link of the notebook

Results

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



Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

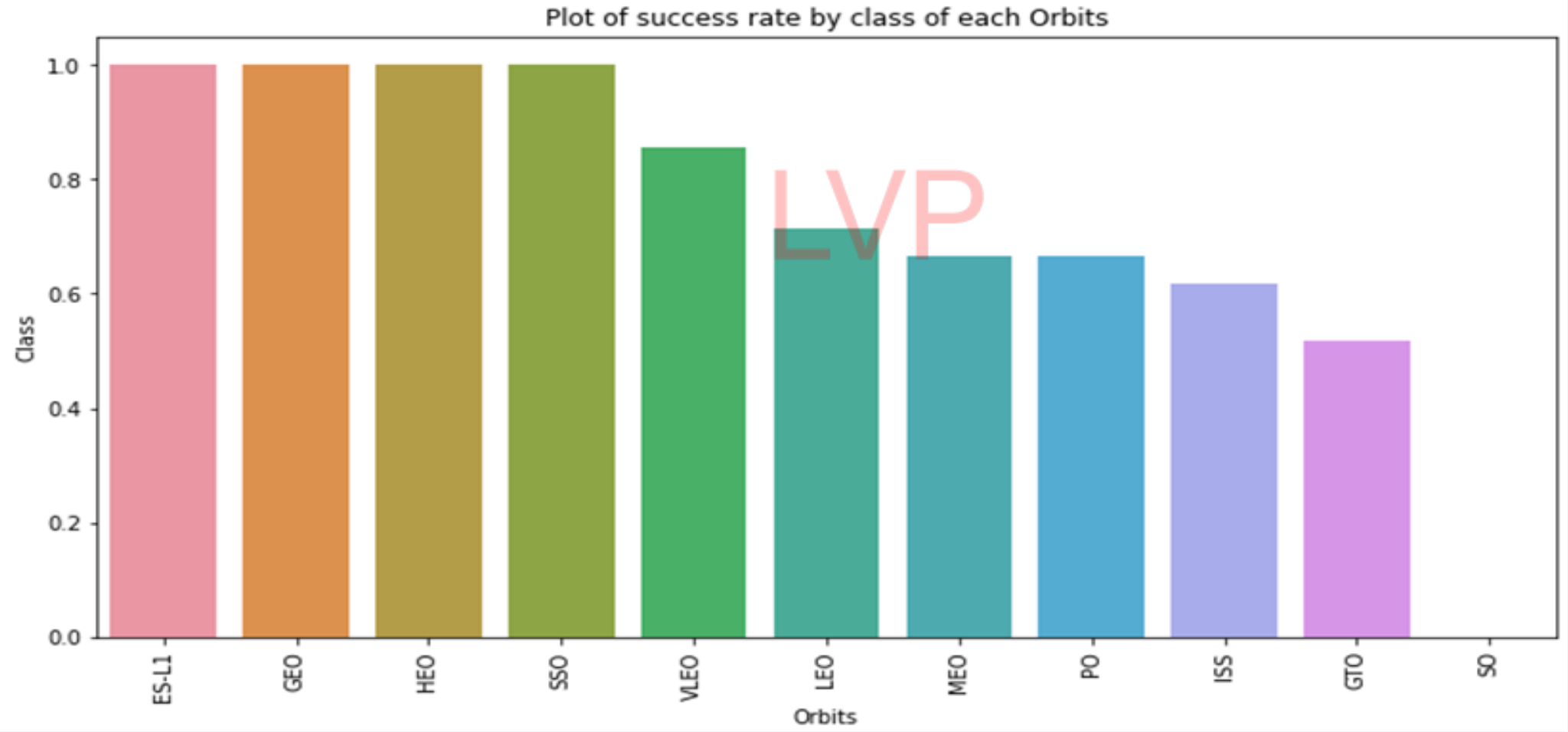
We can see that the success rate is increasing with the number of flights at a specific lunch site.

Payload vs. Launch Site

Even if is not quite clear if Lunch Site is dependent on Pay Load Mass variable for a successful lunch, we cand see from this plot that the greater the payload on specific rocket the higher the success rate (for example, look at Lunch Site CCAFS SLC 40)

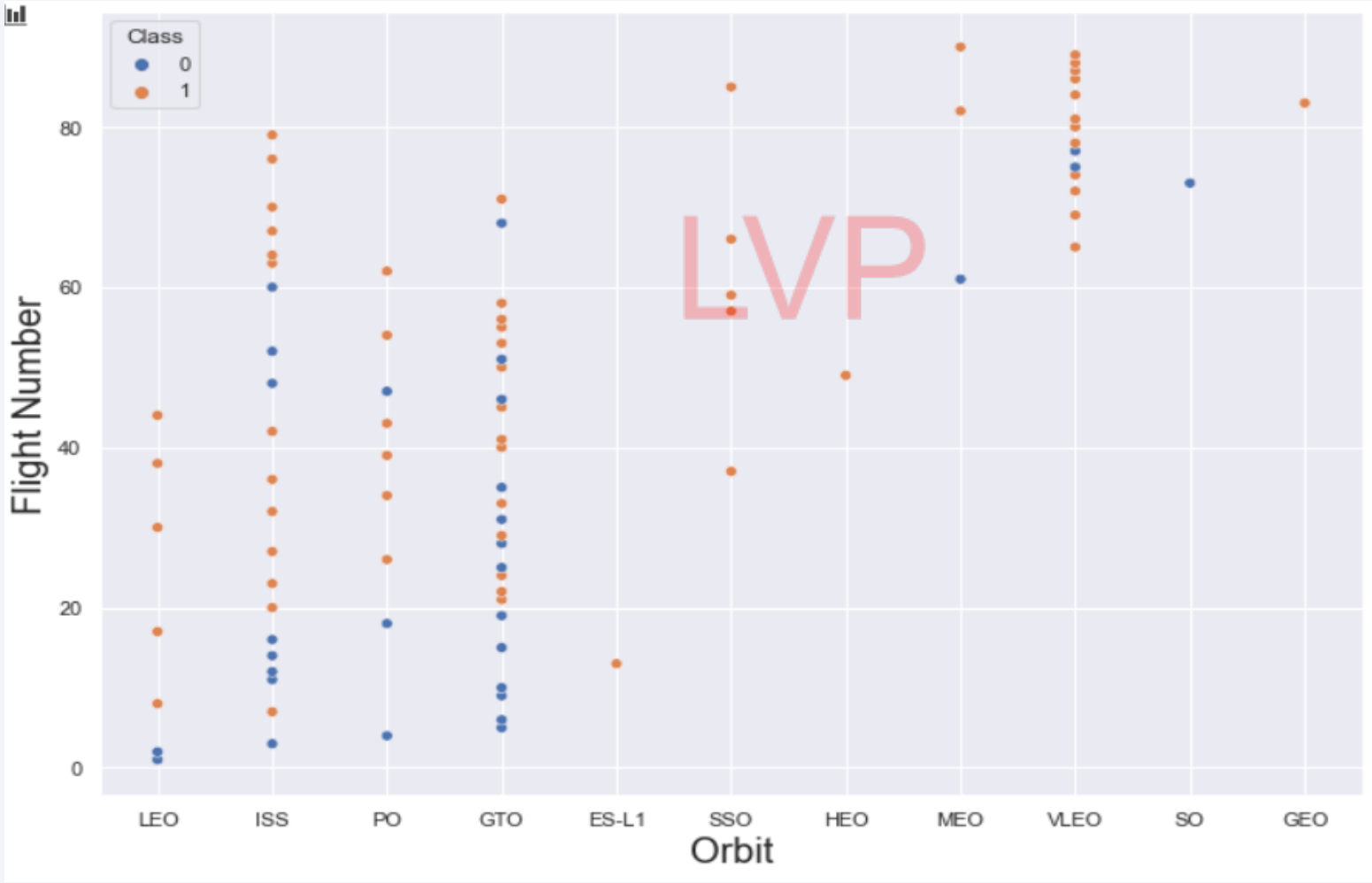
Success Rate vs. Orbit Type

From this plot, we can see that ES-L1, GEO, HEO, SSO and VLEO have the higher success rate of all:



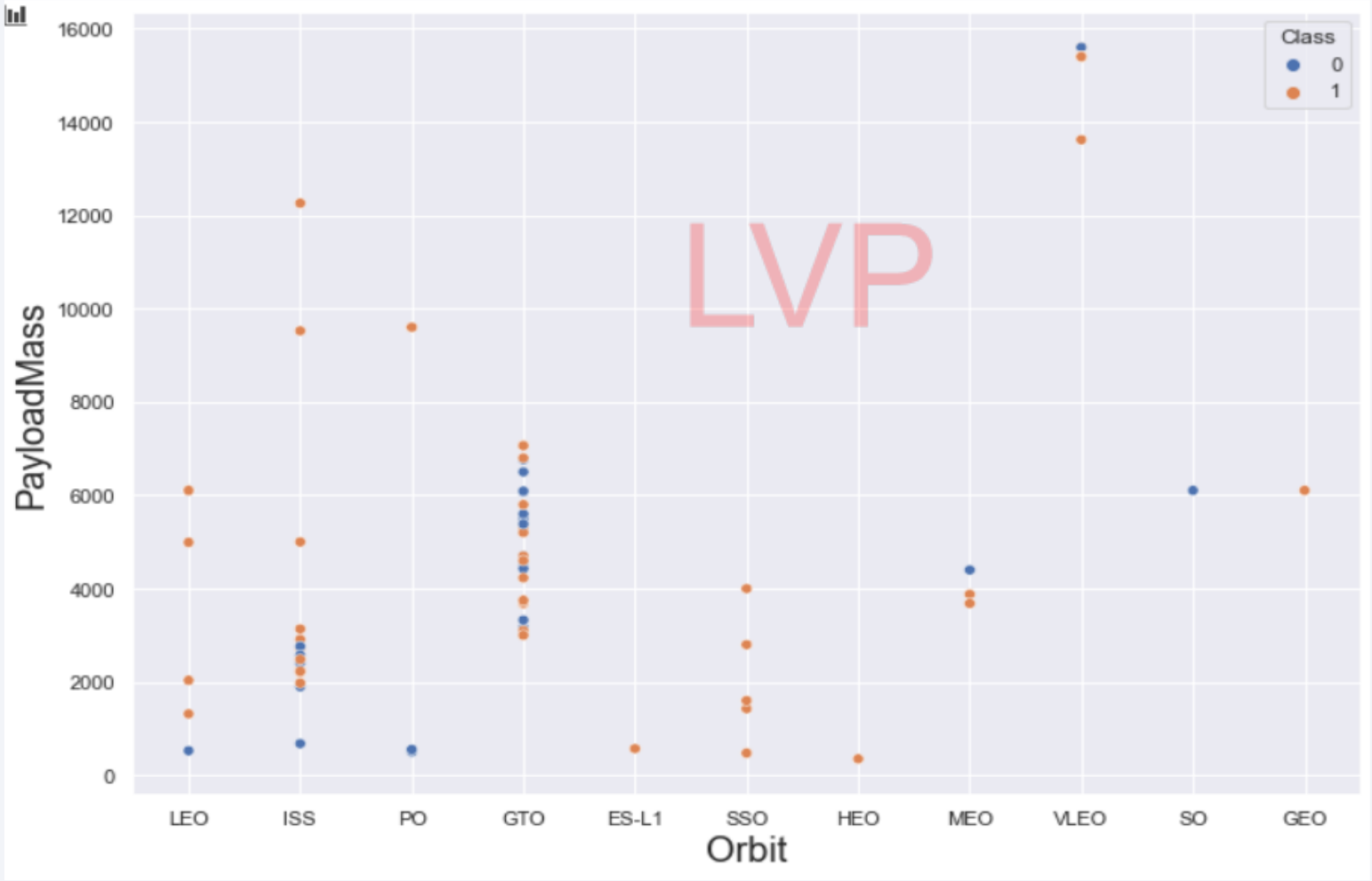
Flight Number vs. Orbit Type

You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit:



Payload vs. Orbit Type

You should observe in this plot that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits:



Launch Success Yearly Trend

We can easily see that the success rate kept increasing starting with 2013:

All Launch Site Names

I used the key word **DISTINCT** to show only unique launch sites from the SpaceX data:

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

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```


Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'CCA'

I used the query below to display 5 records where launch sites begin with `CCA`:

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

In [11]:
```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```


Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	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
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Calculated the total payload carried by boosters from NASA as 45596 using the query below:

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

In [12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

Calculated the average payload mass carried by booster version F9 v1.1 as 2928.4:

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

In [13]:
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

- Found out that the first successful landing took place in 2015 in December on 22nd and we used the below line to pull that date out of the data:

In [14]:

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```


Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000:

In [15]:

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

Used the wildcard like '%' to filter for **WHERE** MissionOutcome was a success or a failure:

List the total number of successful and failure mission outcomes

In [16]:
```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

Out[16]:

The total number of successful mission outcome is:

	successoutcome
0	100

The total number of failed mission outcome is:

	failureoutcome
0	1

Boosters Carried Maximum Payload

I determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function:

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

In [17]:

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```


Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

2015 Launch Records

I used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015:

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [18]:

```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- I selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- I applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

In [19]:
```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```


Out[19]:

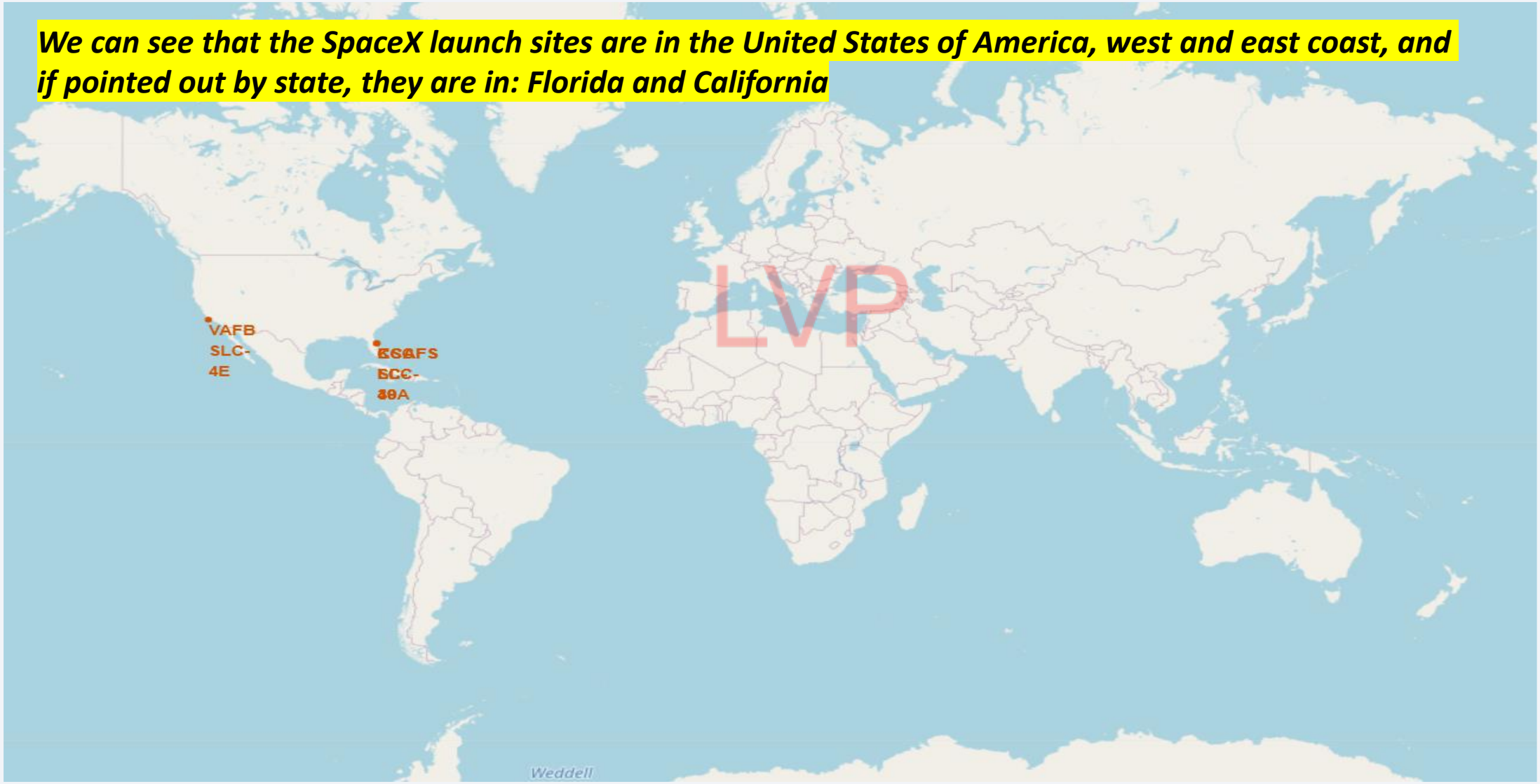
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

Section 3

Launch Sites Proximities Analysis

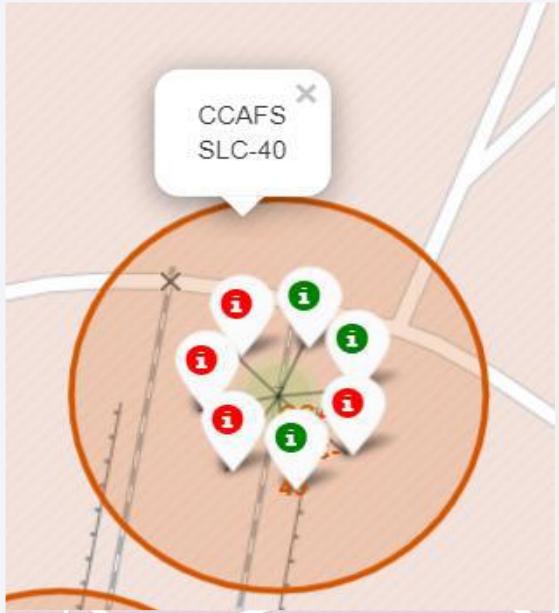
All launch sites global map markers

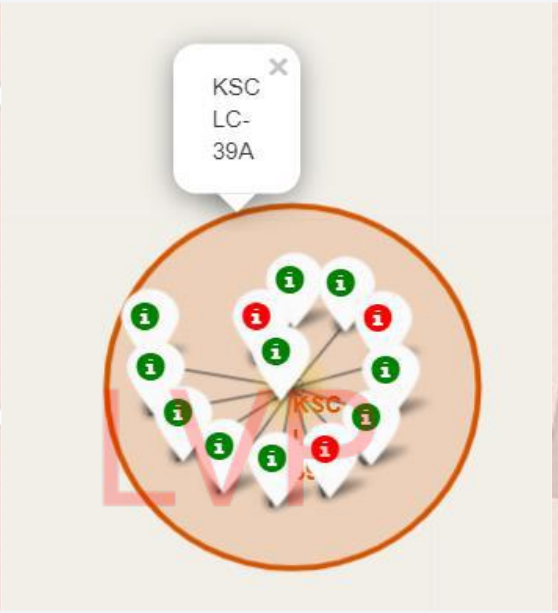
We can see that the SpaceX launch sites are in the United States of America, west and east coast, and if pointed out by state, they are in: Florida and California

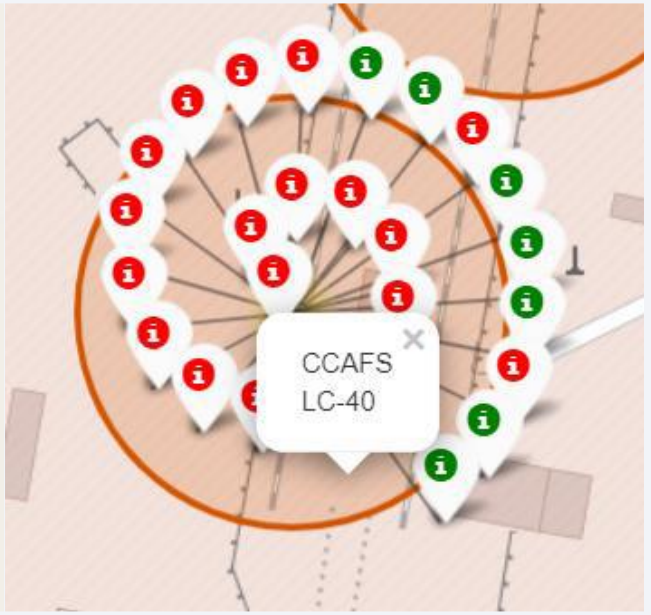


Markers showing launch sites with color labels

Florida lunch sites:







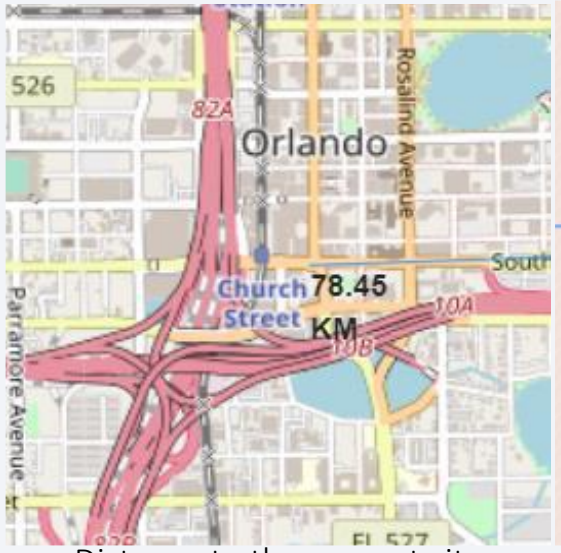
California lunch site:

* Where the green markers are for successful lunches, and the red one for failures;

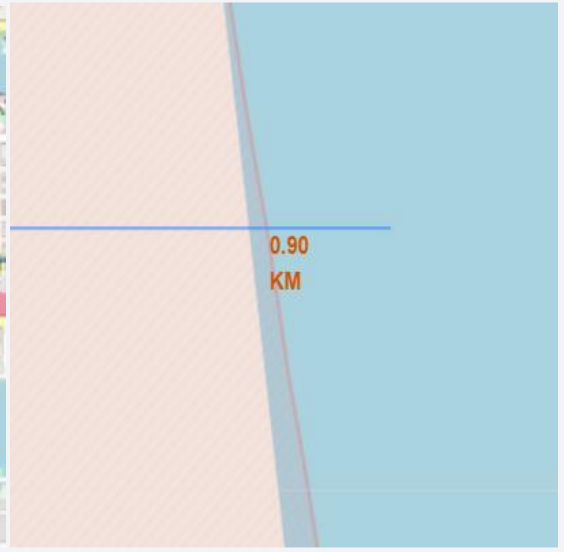
Launch Site distance to landmarks

Distance to nearest highway

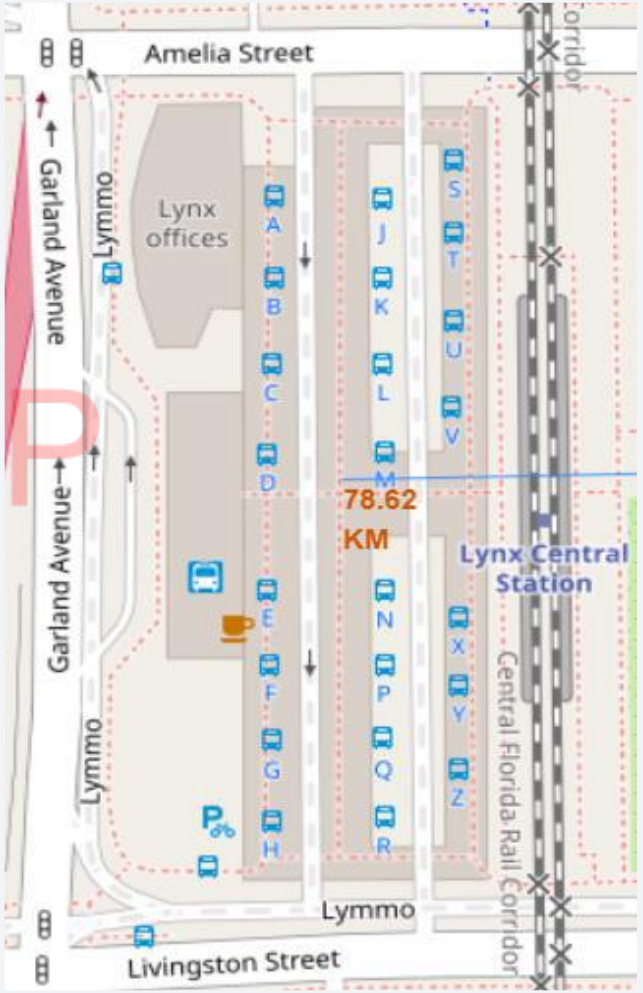
Distance to the coast



Distance to the nearest city



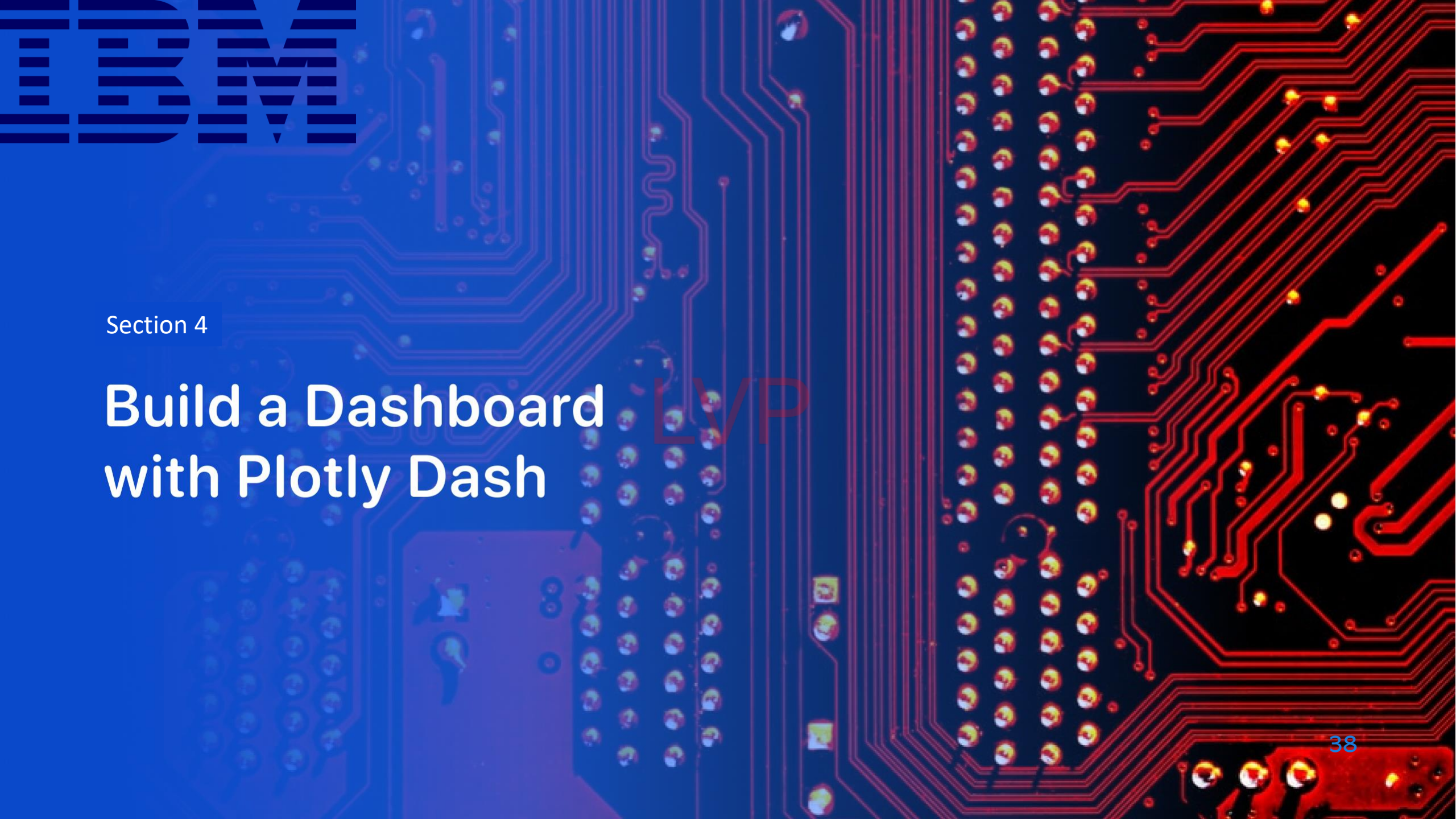
Distance to the coastline



Distance to the Railway station

Questions to be asked:

1. Are lunches sites away from cities? A: ***YES!***
2. Are there any highways close to lunch sites? A: ***NO!***
3. Any railways close to lunch sites? A: ***NO!***
4. Are lunch sites positioned close to the coastline? A:***YES!***

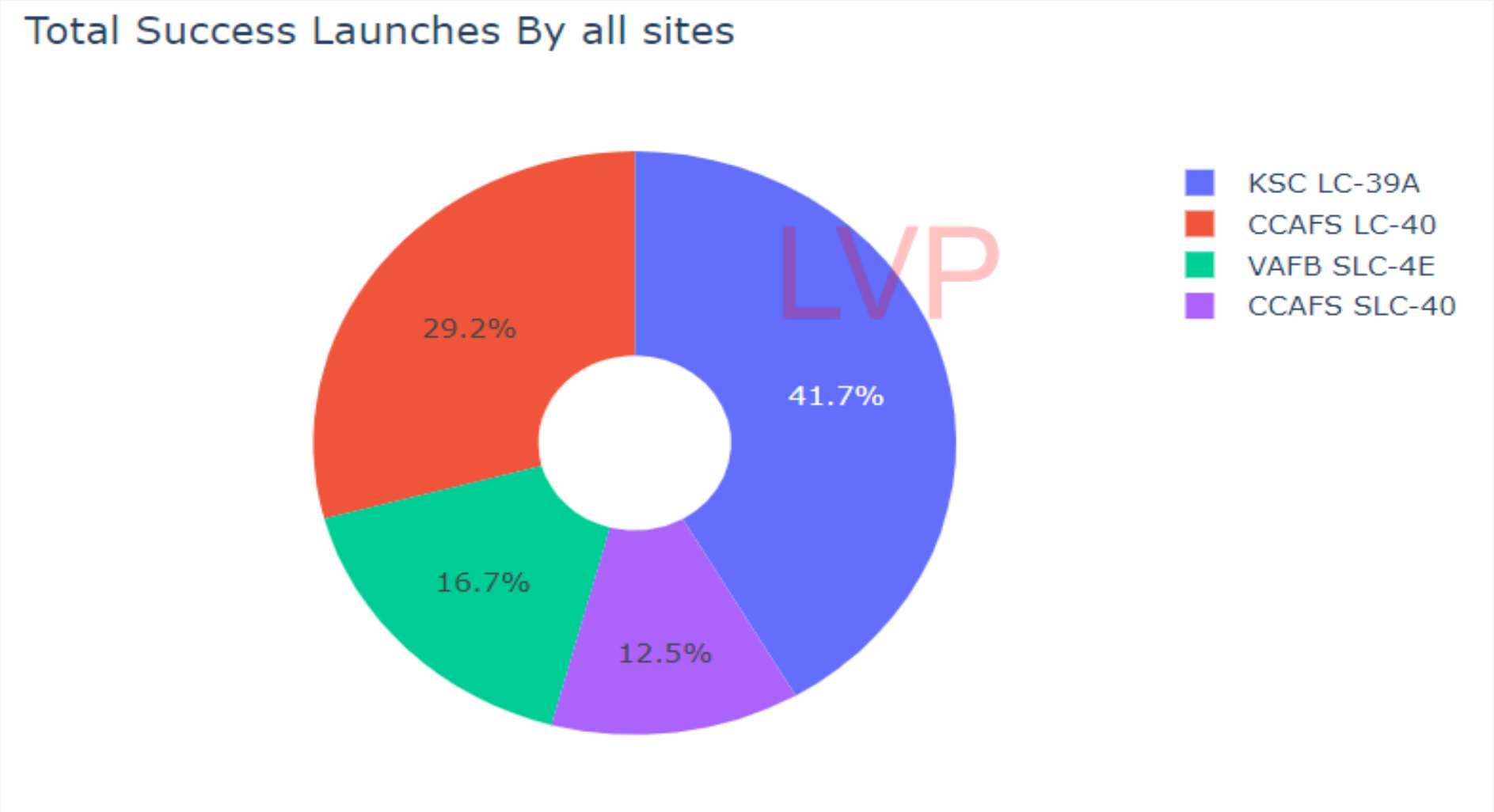


Section 4

Build a Dashboard with Plotly Dash

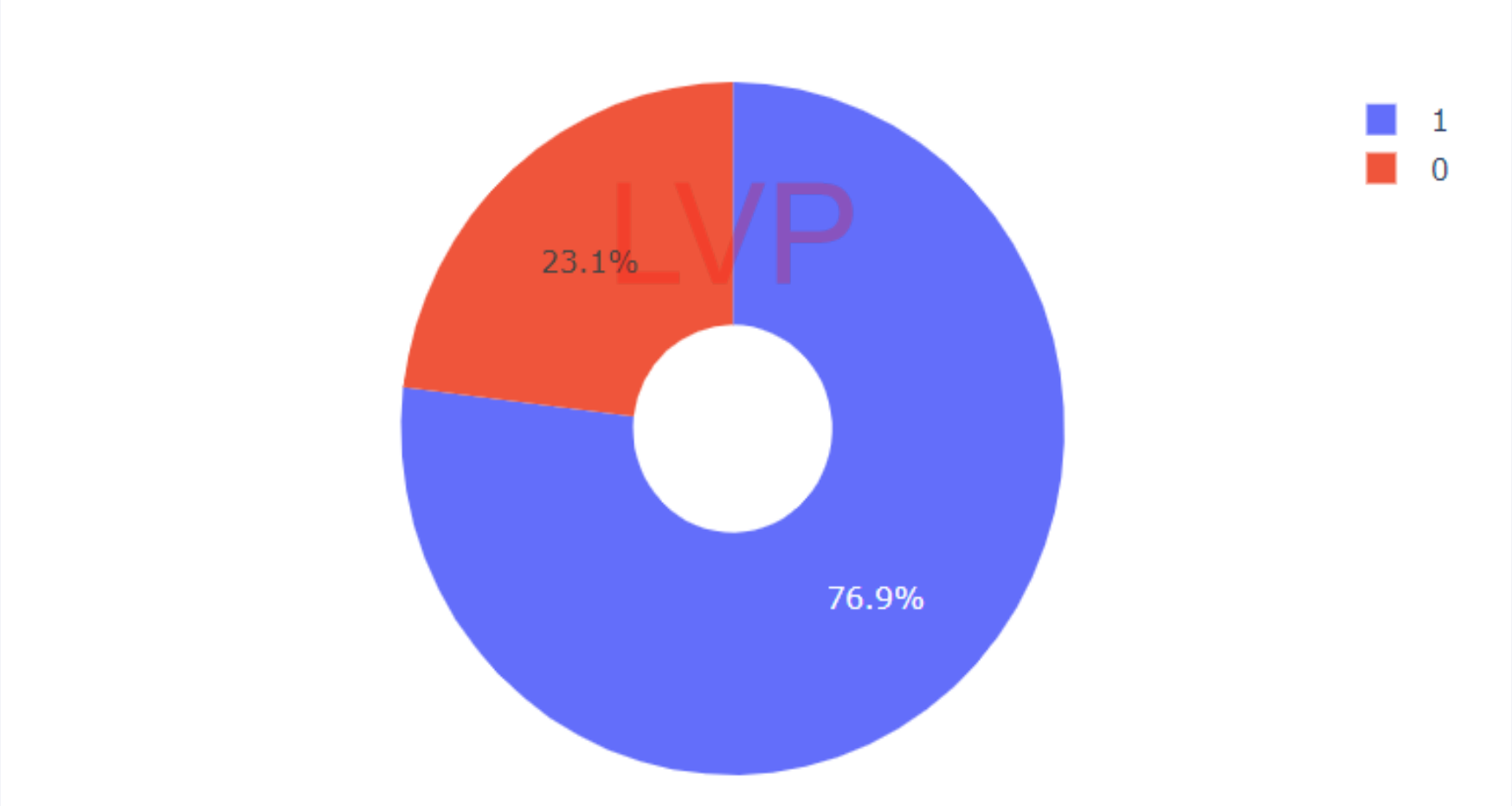
Dashboard: Pie chart - percentage of successful lunches achieved by each launch site

Which lunch site has the most successful lunches of all? A: KSC LC-39A.

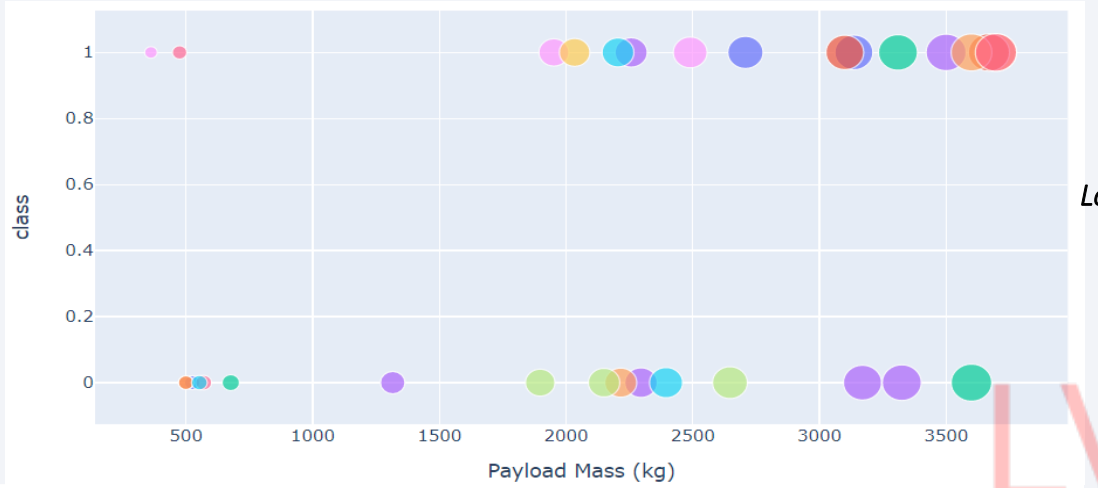


Dashboard: Pie chart on the lunch site with the highest success ratio

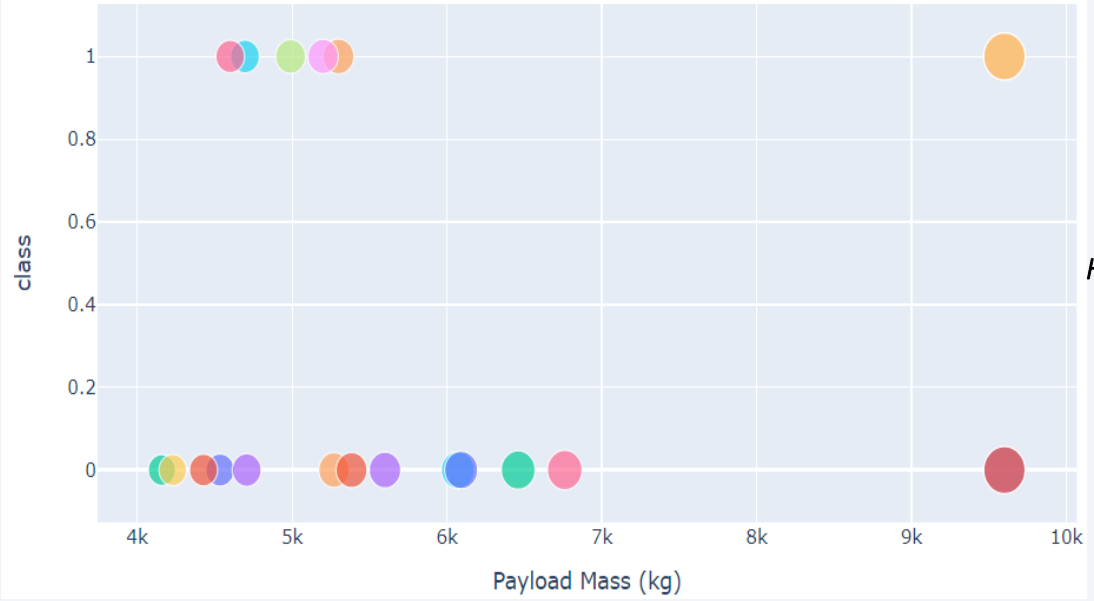
On the KSC LC39A they managed to successfully lunch 76.9% of the rockets, while keeping the failure rate just under 25% (more specific 23.1%)



Dashboard: Payload vs Launch Outcome of all sites with different payload selected, plotted using scatter plot.



Low Weighted Payload 0kg –4000kg



Heavy Weighted Payload 4000kg –10000kg

It is clear that in order to get a higher successful lunch ratio, they had to come up it solutions to reduce the Payload. In this scatter plots we can see that lower payloads have a better lunch outcome than the heavy loads.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The model with the highest classification accuracy after the testing was Decision Tree.
- After selecting the best hyperparameters for the decision tree classifier using the validation data, we achieved ***88.92%*** accuracy on the test data.

```
models = {'KNN':knn_cv.best_score_,'Decision Tree':tree_cv.best_score_,'Logistic Regression':logreg_cv.best_score_,'SVM':svm_cv.best_score_}
best_model = max(models, key= lambda x: models[x])

print('Highest performing model is \"',best_model,'\" with a score of',models[best_model])
```

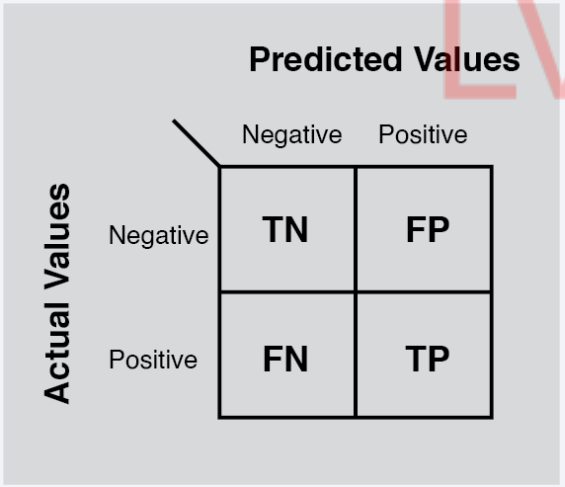
```
Highest performing model is " Decision Tree " with a score of 0.8892857142857142
```

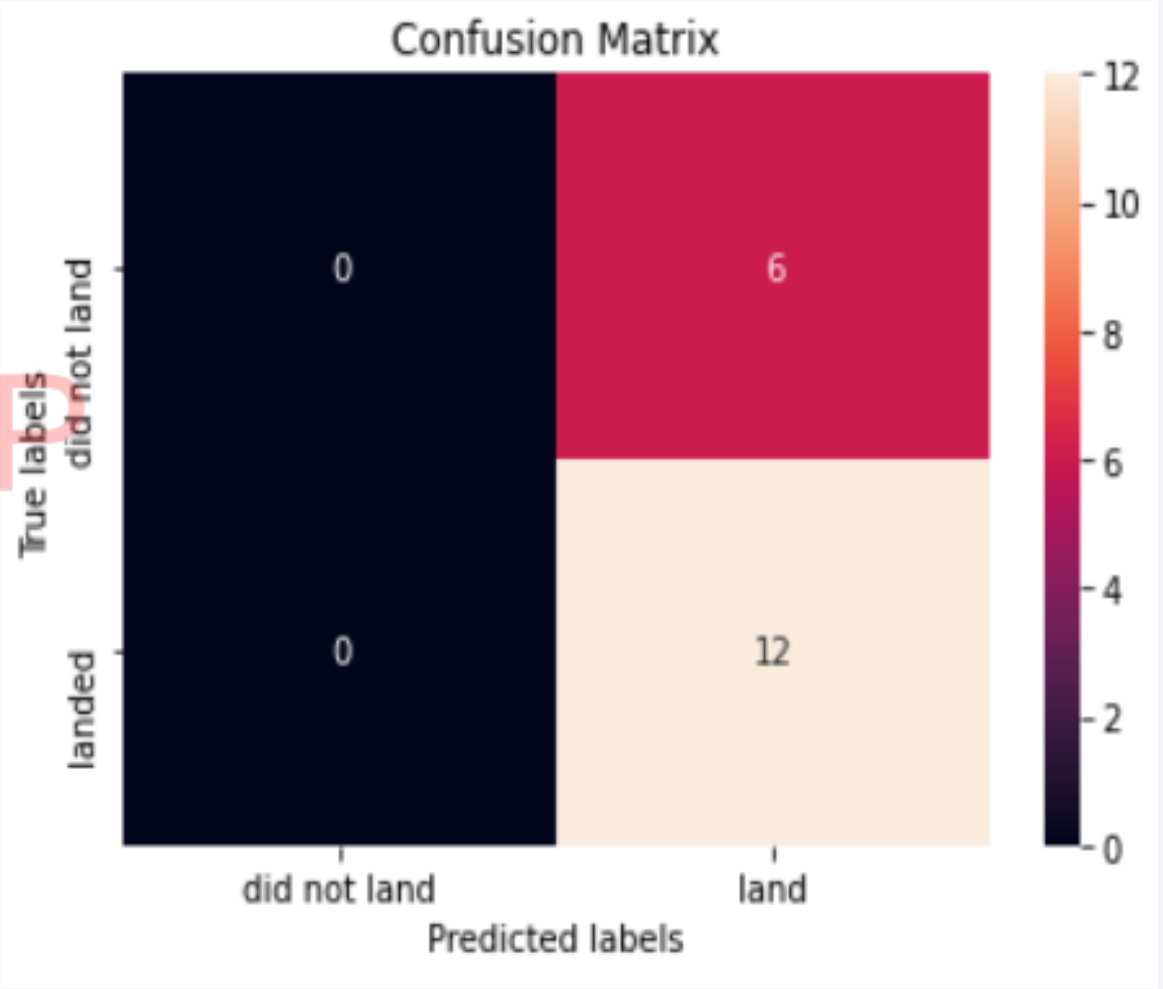

- Accuracy scores on other models on test data:
- - Logistic Regression: 83.33 %;
- - SVM: 83.33 %;
- - KNN: 83.33 %.

Confusion Matrix for Decision Tree Model

- Looking at the confusion matrix of the Decision Tree predicted label we can see that the algorithm can properly identify the correct label between the different labels, having a bit of a hard time on the False Positive labels.
- Confusion Matrix Explained:

[Source Code](#) Here





Conclusions

- In order to get a higher success rate, they need to drop the weight;
- Best results were obtained between 2013 till 2020, with constant improvement starting in 2013;
- KSC LC-39A has the highest score of successful lunches at any of the lunch sites;
- Out of all the Machine Leaning models we tested, best results were obtained with the Decision Tree Classifier with *88.92%* accuracy on the test data.

THANK YOU!

@Vasile Lucian Popa