IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Yan Xia
April 9, 2022

Outline

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

Executive Summary

- The purpose of this project is to train machine learning model using data collected from SpaceX REST API and use the model to predict if SpaceX will result the first stage.
- After Exploratory Data Analysis, features correlated with successful rate including Flight Number, Payload Mass, Orbit, Flights, GridFins, Reused, Legs, Landing Pad, Block, ReusedCount, launch site and Serial.
- The models are trained and performed Grid Search to find the best hyperparameters. Using these hyperparameter values, we determine the model with the best accuracy using the training data.

Introduction

- SpaceX is the most successful company in providing affordable space travel thanks to its Falcon 9 Rocket that can recover the first stage. In order to compete with SpaceX, we need to understand competitor's average cost for each launch.
- the purpose of this project is to use public SpaceX launching information to predict if SpaceX will reuse the first stage.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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 is collected from a SpaceX REST API. This API will give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.

Data Collection – SpaceX API

GitHub link

REST API – end point

api.spacexdata.com/v4/launches/past

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

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

Decode the response content as Json and turn into DataFrame

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

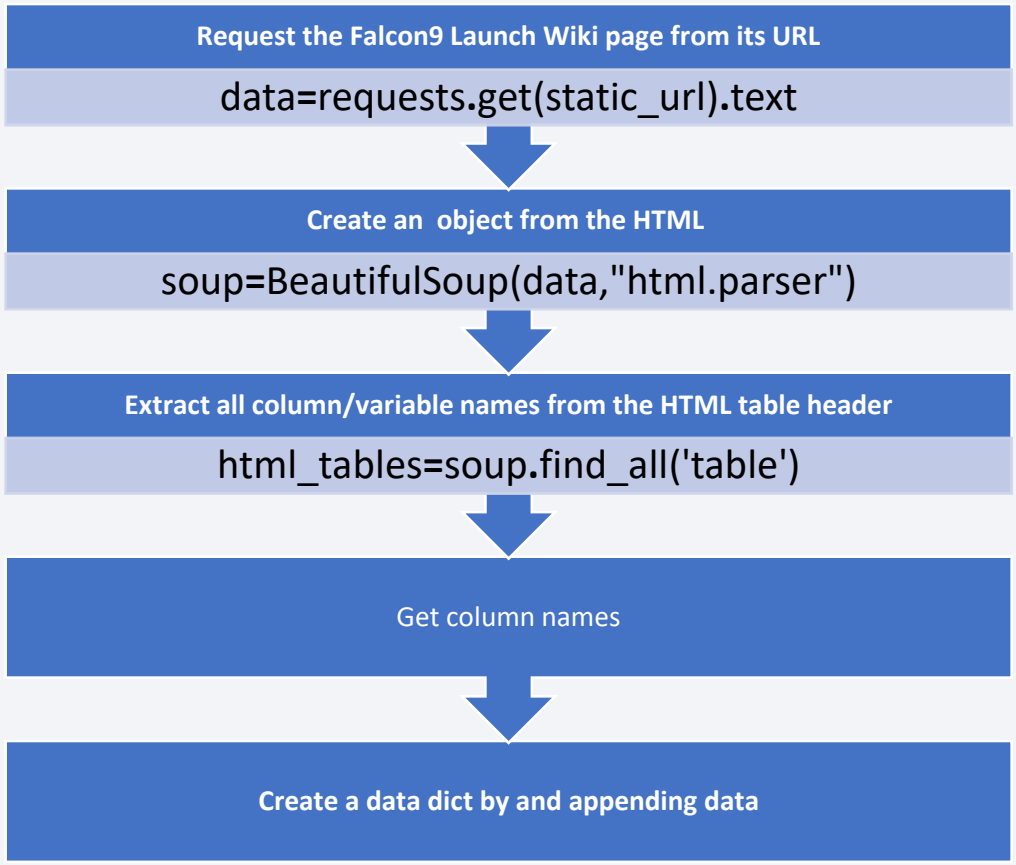
Apply functions to create list and append to dict

```
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}
```

Filter out data and Save results as csv

Data Collection - Scraping

GitHub link



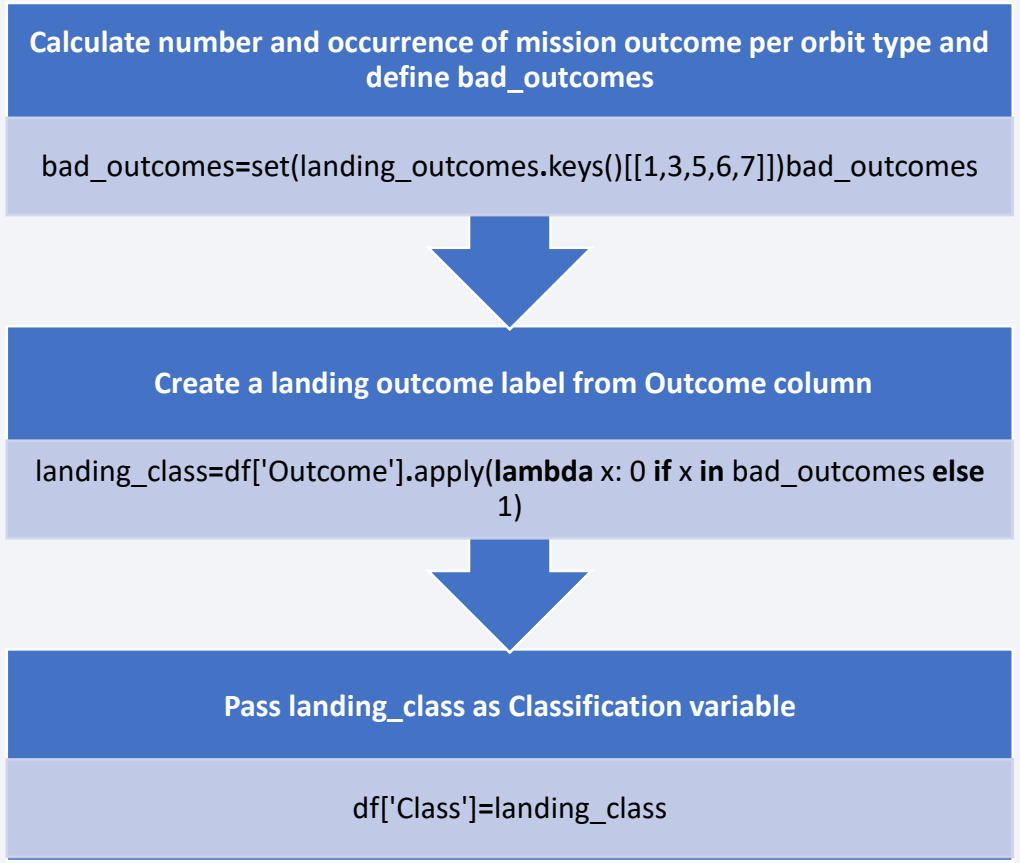

```
for row in first_launch_table.find_all('th'):
    # Get all columns in each row.
    header = extract_column_from_header(row)
    if header is not None and len(header)>0:
        column_names.append(header)
```

Python code explanation in Appendix

Data Wrangling

GitHub link

- The column Outcome indicates if the first stage successfully landed, and we define the bad_count to define classification feature.
- Using the Outcome, create a list where the element is zero if corresponding row in Outcome is in the set bad_outcome



EDA with Data Visualization

GitHub link

- Scatter plots were used to explore relationship between Payload Mass vs. Flight Number, Flight Number vs. Launch Site, Payload Mass vs. Launch Site, Flight Number vs. Orbit type, and Payload Mass vs. Orbit type.
- Bar plot was used to show relationship between success rate and orbit type.
- Line plot was used to show success rate over year over year.

EDA with SQL

GitHub link

1. unique launch sites in the space mission
2. 5 records where launch sites begin with string CCA
3. total payload mass carried by booster launched by NASA(CRS)
4. Average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome in ground pad was achieved
6. names of boosters whchi have success in drone ship and have payload mass greater than 4000 but less than 6000
7. List total number of successful and failure mission outcomes
8. The names of the booster_versions which have carried the maximum payload mass
9. The failed landing_outcomes in drone ship, their booster versions and launch site names for in year 2015
10. Rank the count of landing outcomes between the date 2010-6-4 and 2017-3-20

Build an Interactive Map with Folium

GitHub link

- Interactive map with Folium is focused on analyzing launch site geo and proximities with Folium.
- First mark the launch site locations and their close proximities on an interactive map. In this step, marker object was used to mark the location on the map.
- Then, we explore the map with those markers and try to discover any patterns. In this step, used circle and icon marker for each launch site on the map and indicated success or failure of each launch.
- Finally, explain how to choose an optimal launch site. In this step, lines and distance object is marked on the map to find nearest distance to highway, city or coastline.

Build a Dashboard with Plotly Dash

GitHub link

- Dropdown menu and pie chart shows total success launches by site or total success launches for each site. Since we have four different launch sites, the dropdown menu will allow us to see which one has the largest success count. Then we would select one specific site and check its detailed success rate (class=0 vs. class=1).
- Slider with scatter plot chart shows relationship between Payload Mass, booster version category and success rate. From a dashboard point of view, we want to be able to easily select different payload range and see if we can identify some visual patterns and this is reason why Slider was added here.

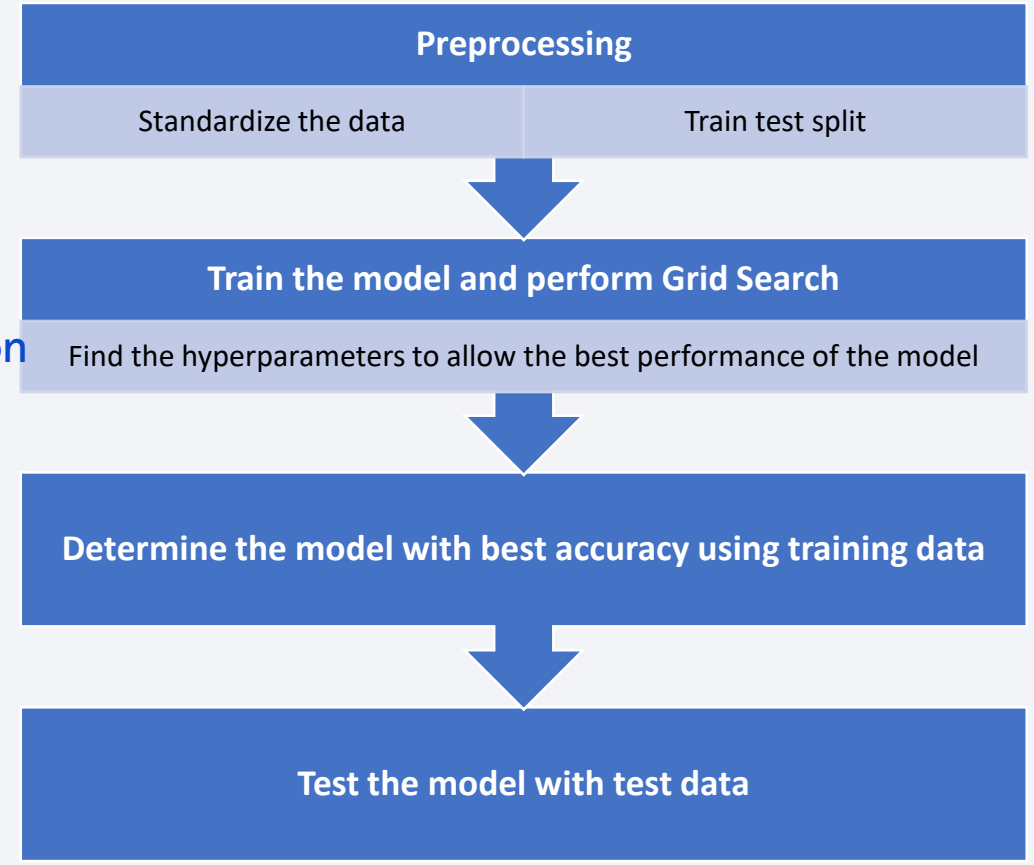
Predictive Analysis (Classification)

GitHub link

- There are 4 classification models considered including Logistic Regression, Support Vector machines, Decision Tree Classifier, and KNN.

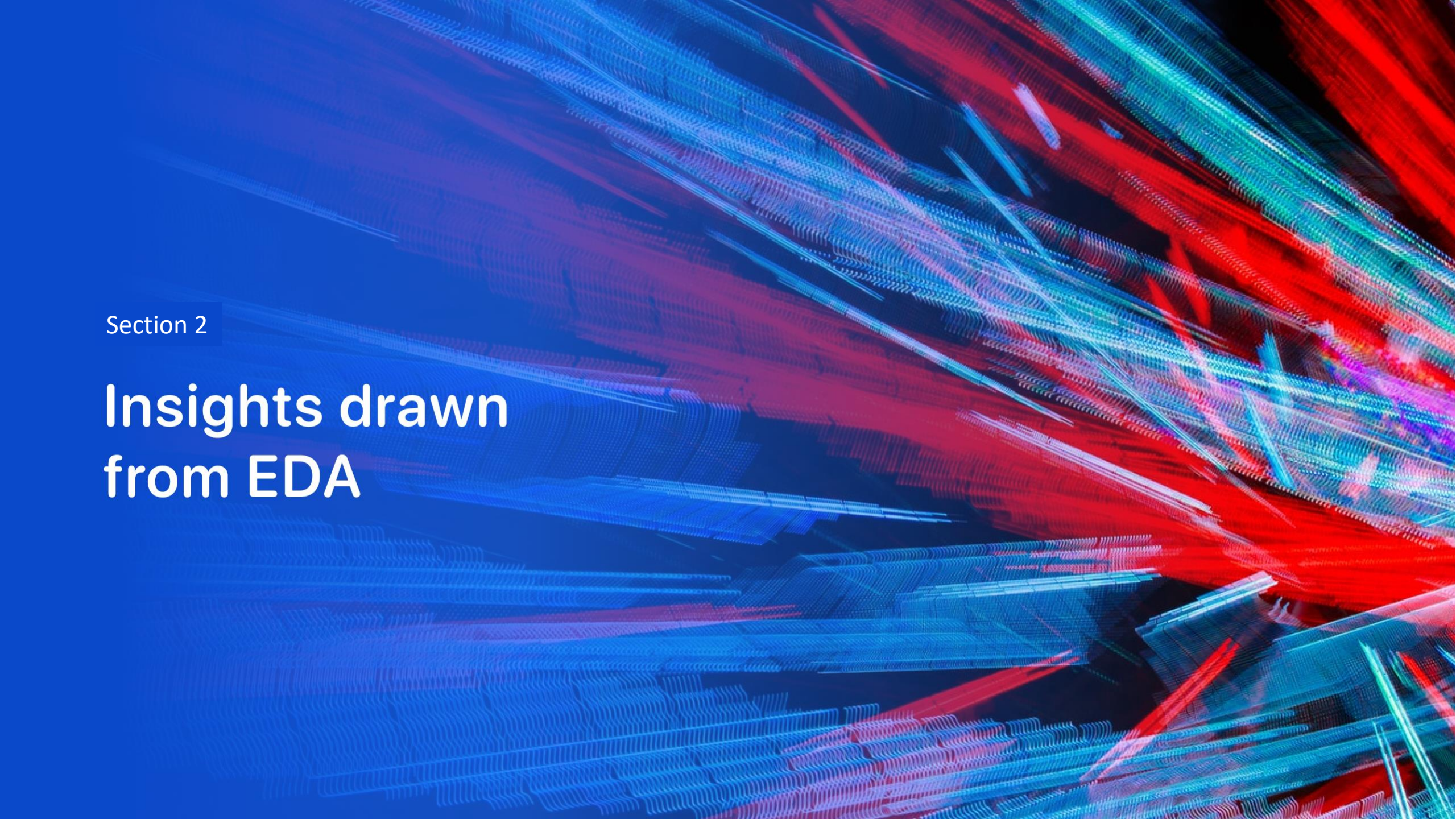
Predictive Analysis (Classification

- See python code in Appendix

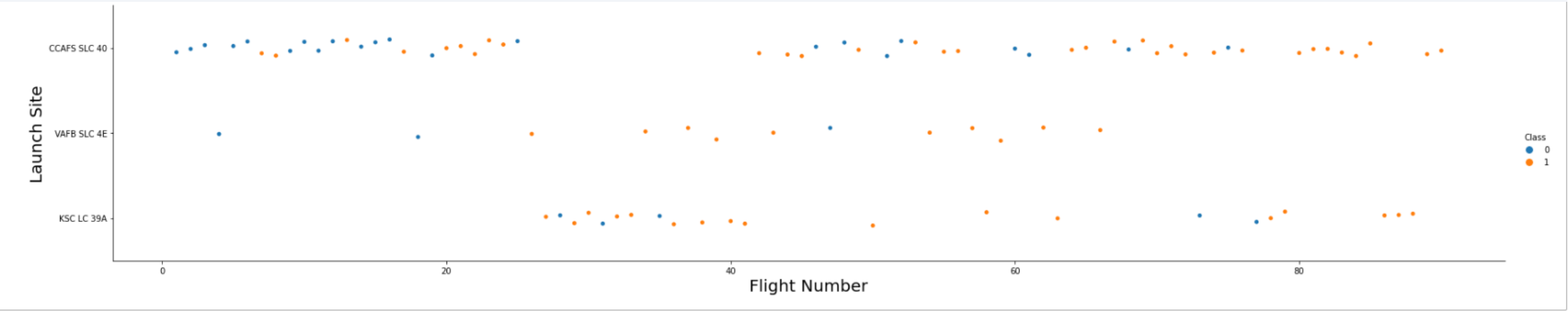


Results

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

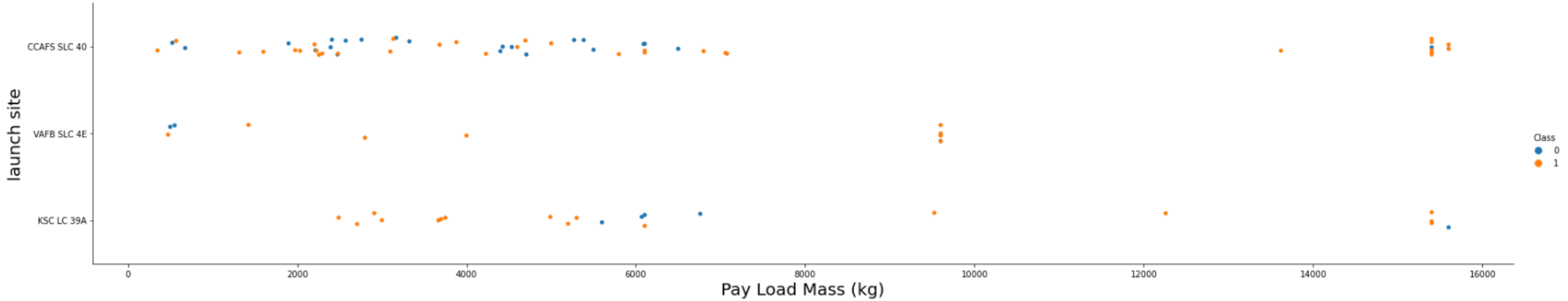

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



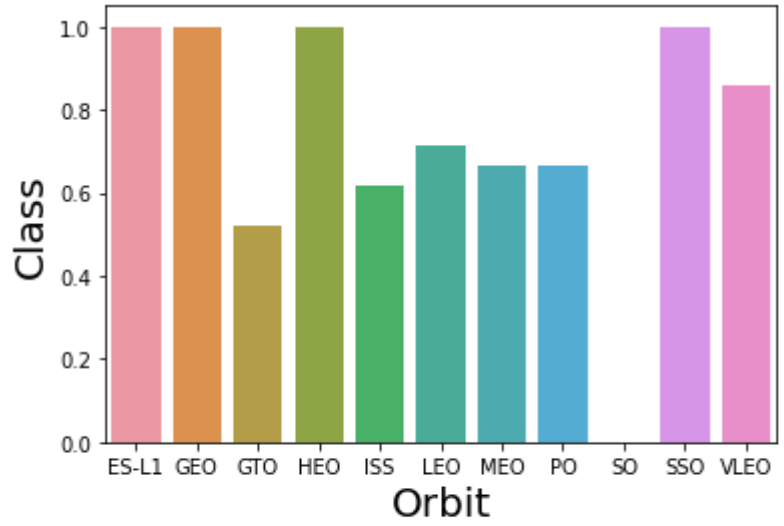
Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.
We see that as the flight number increases, the first stage is more likely to land successfully. At begining, most of the launches are at CCAFS SLC 40, and success rate is very low then launches moved to KSC LC 39 A, the success rate got higher.

Payload vs. Launch Site



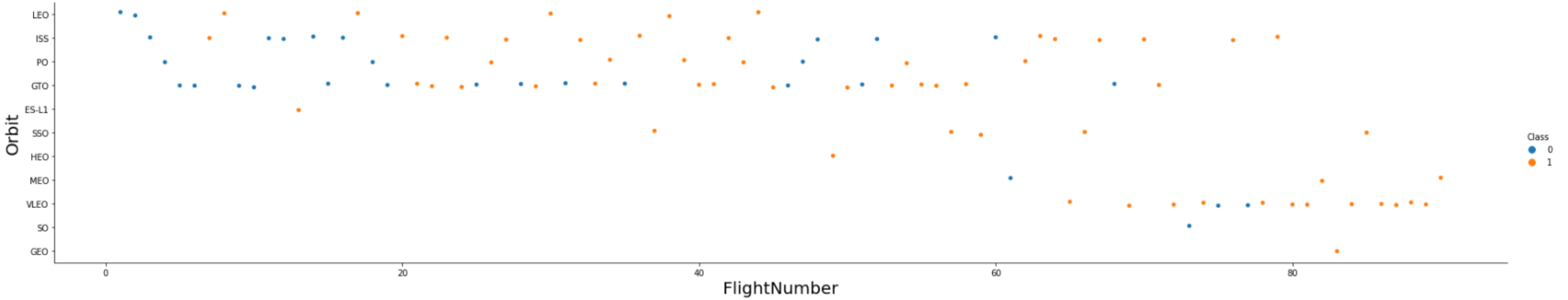
Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type



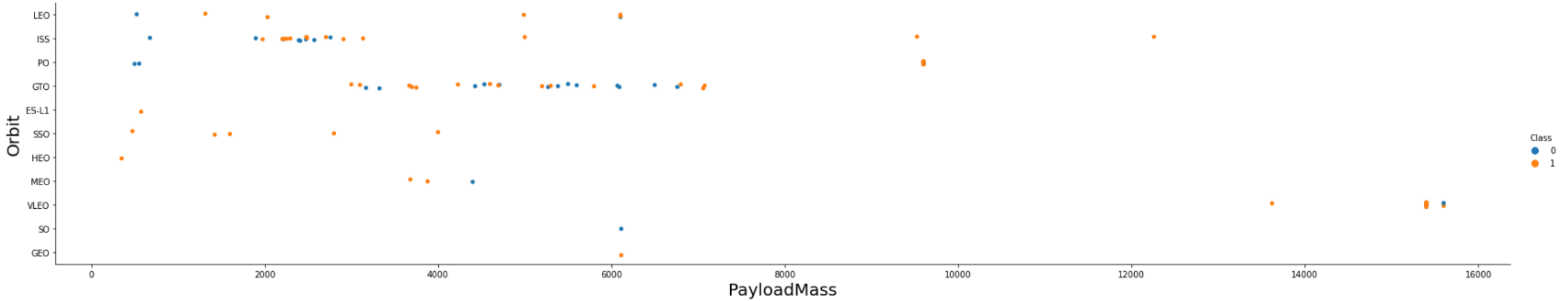
Analyze the ploted bar chart try to find which orbits have high sucess rate. ES-L1, GEO, HEO, SSO

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.

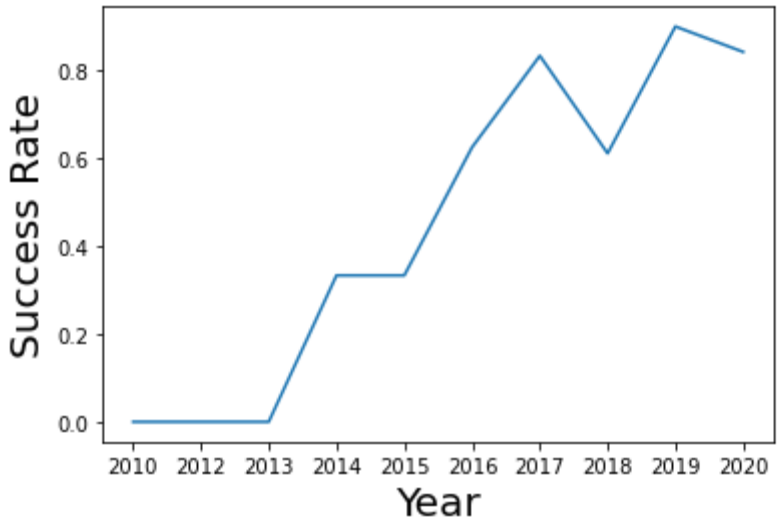
Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

- Find the names of the unique launch sites
- There are 4 unique launch sites.

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

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

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- total payload carried by boosters from NASA is 45,596 KG

total_payload_mass
45596

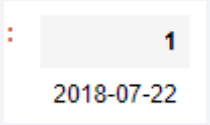
Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- average payload mass carried by booster version F9 v1.1 is 2,928 KG

booster_version	avg_payload_mass
F9 v1.1	2928

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- first successful landing outcome on ground pad is 2018-7-22



Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

- the names of boosters are listed as shown below.

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Present your query result with a short explanation here

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Present your query result with a short explanation here

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

2015 Launch Records

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

- Present your query result with a short explanation here

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

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

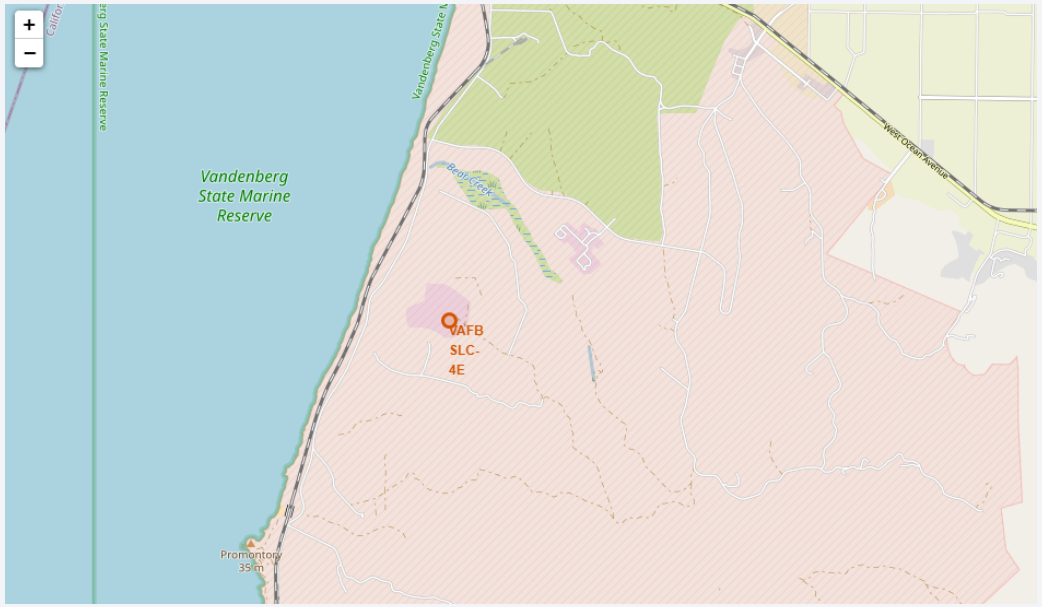
- Present your query result with a short explanation here

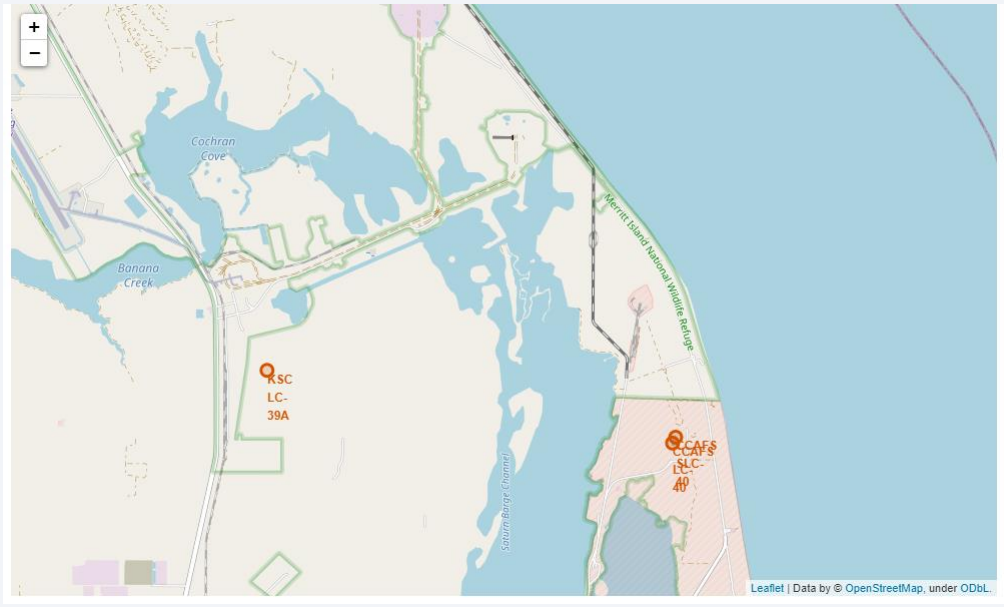
landing__outcome	COUNT
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

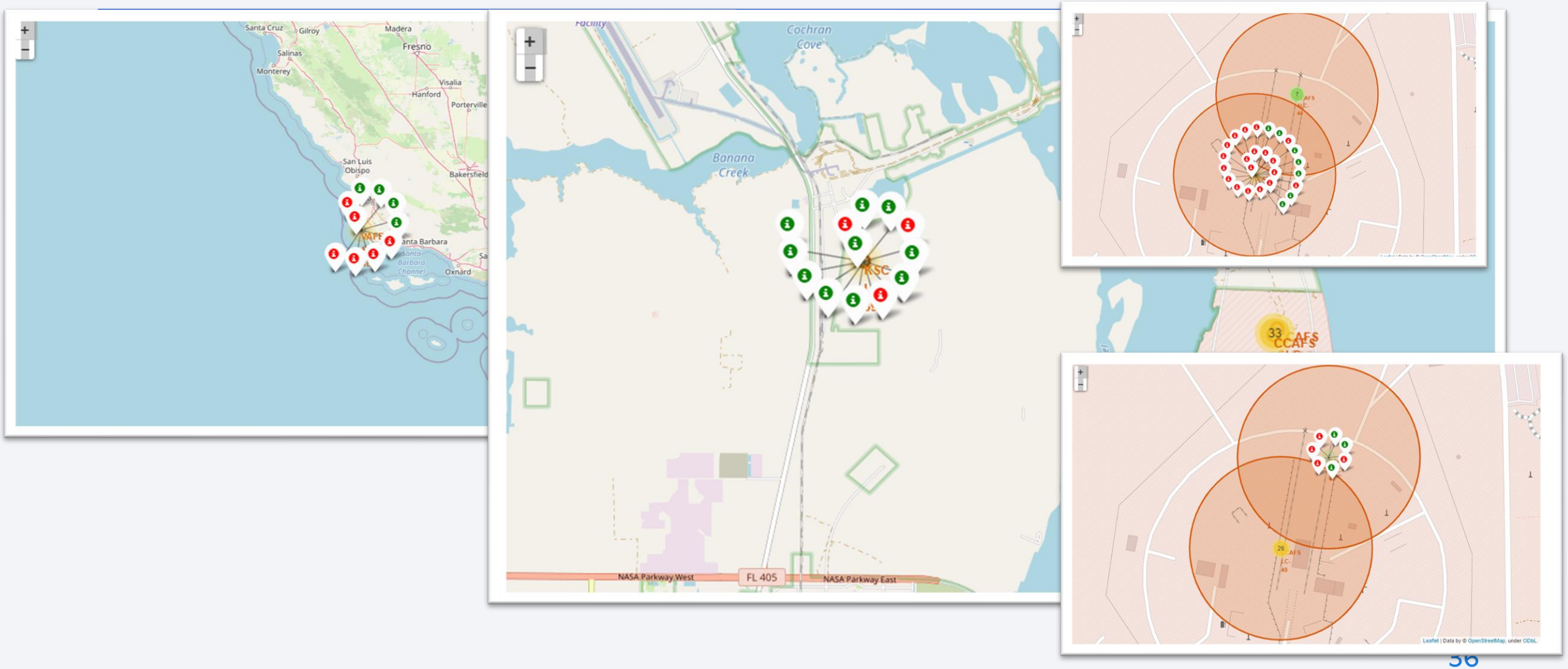
Mark all launch sites on a map



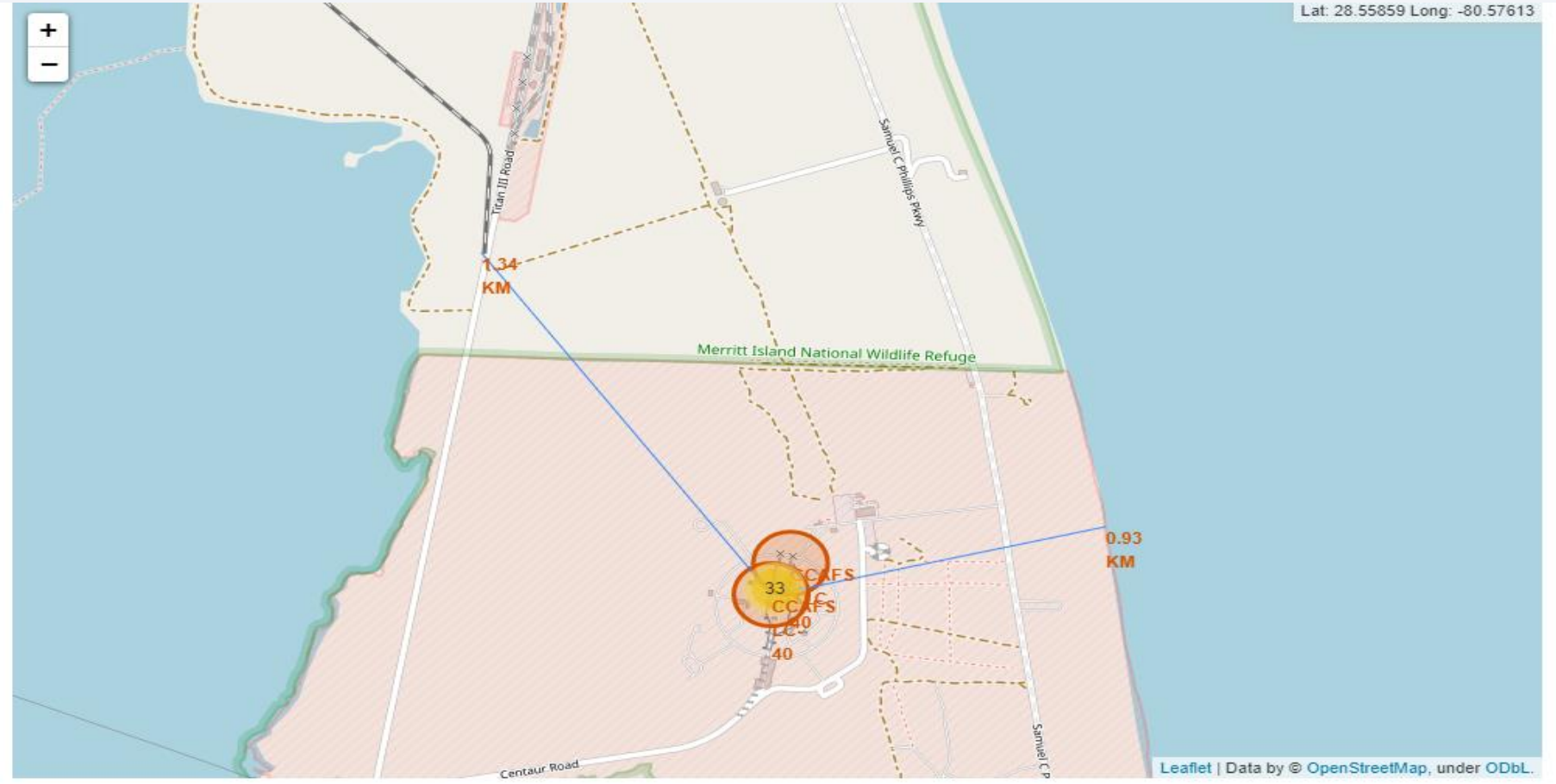


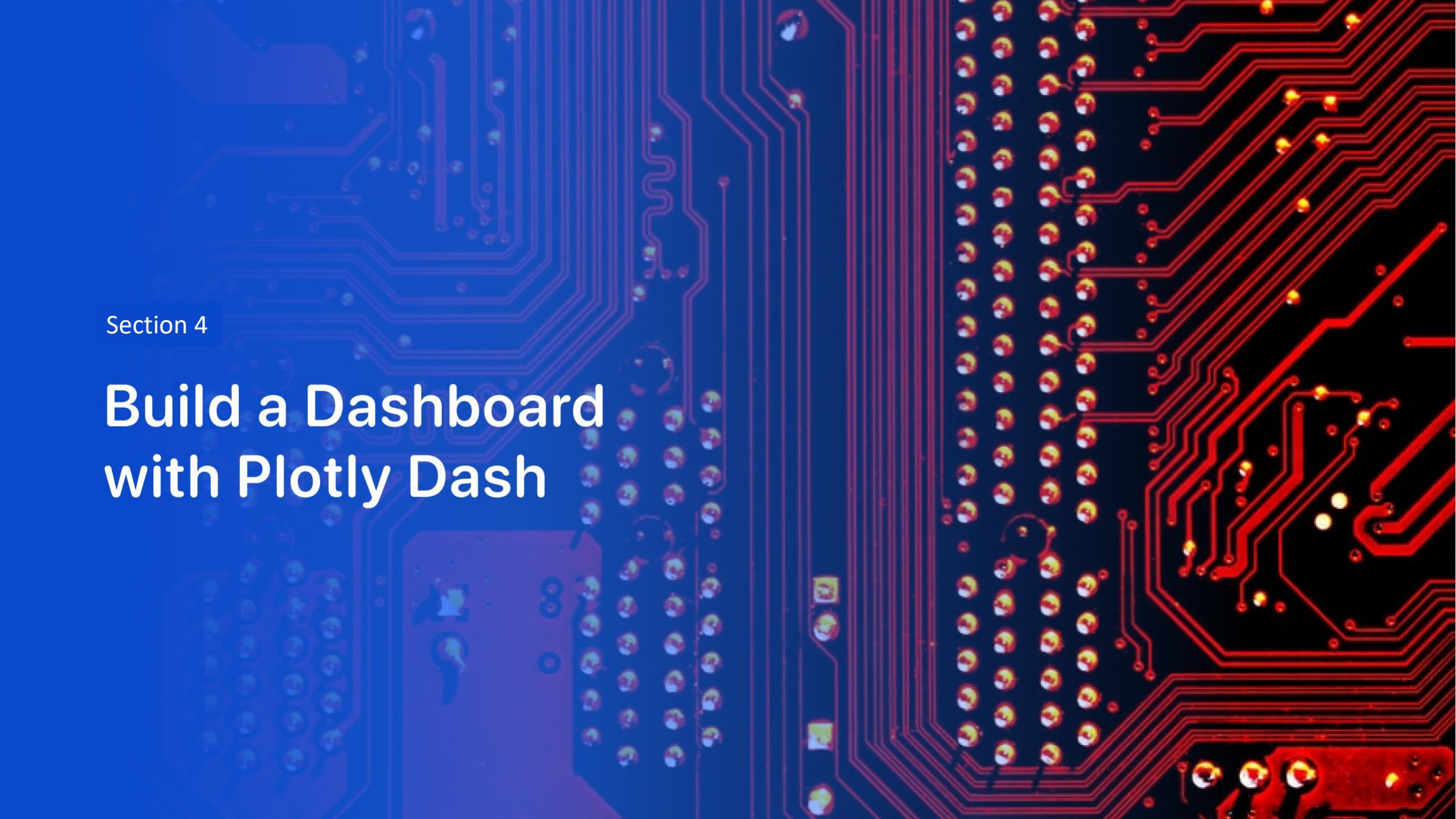
- All launch sites are in proximity to the Equator line and are very close proximity to the coast

Mark the success/failed launches for each site on the map

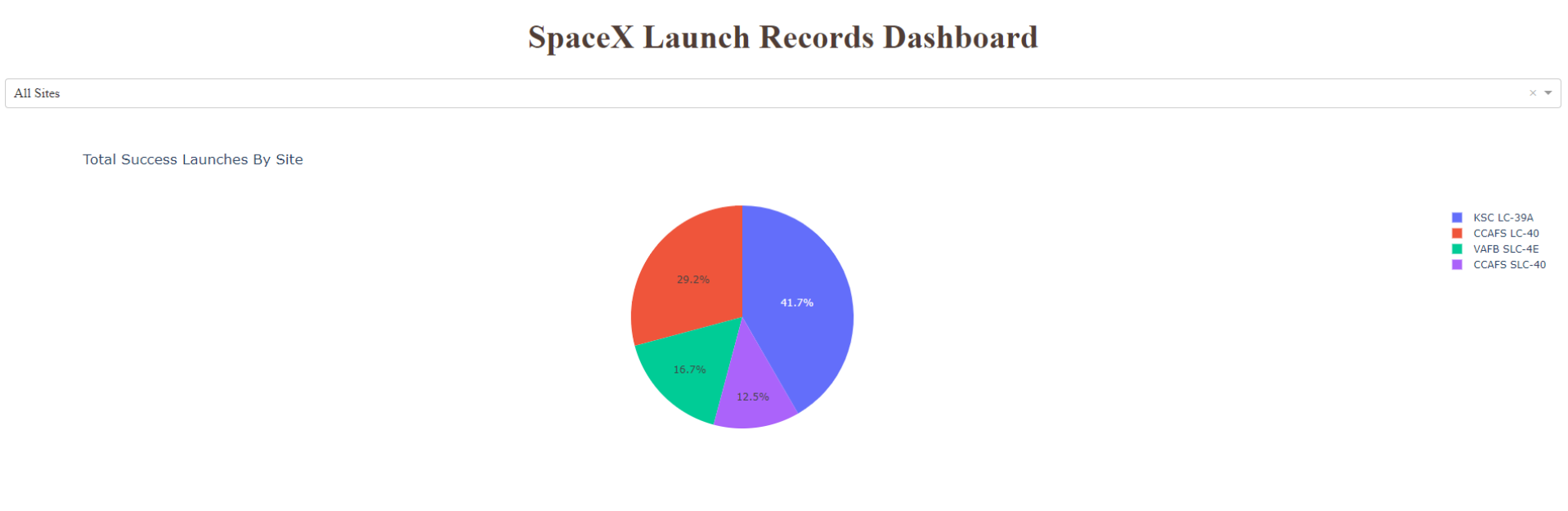


Calculate the distances between a launch site to its proximities



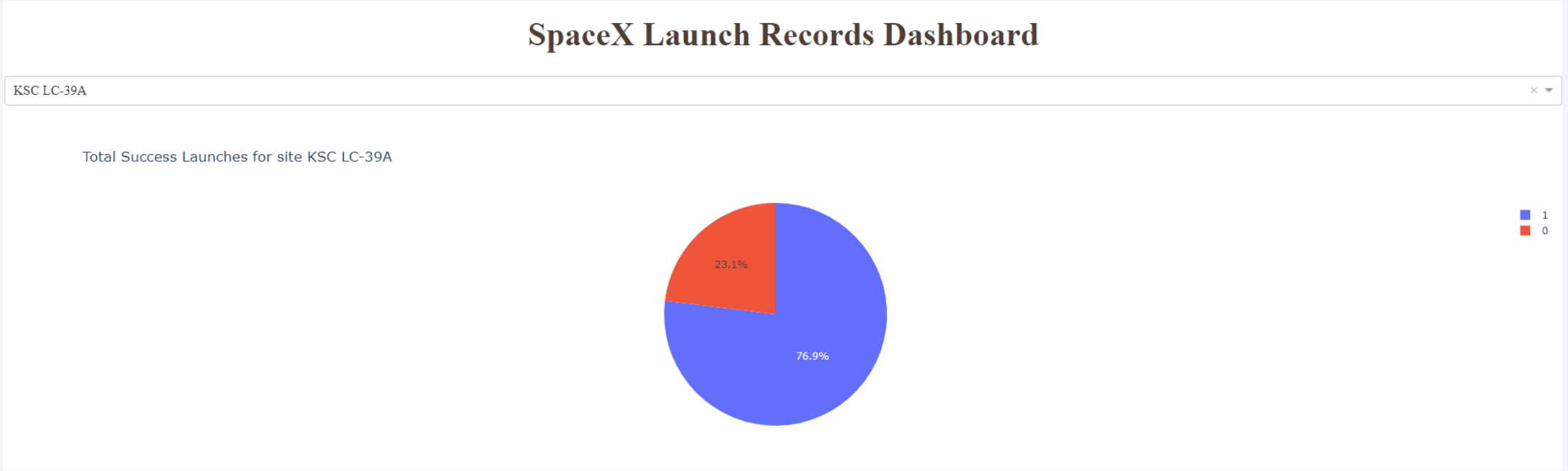

Section 4
Build a Dashboard with Plotly Dash

Launch success count for all sites



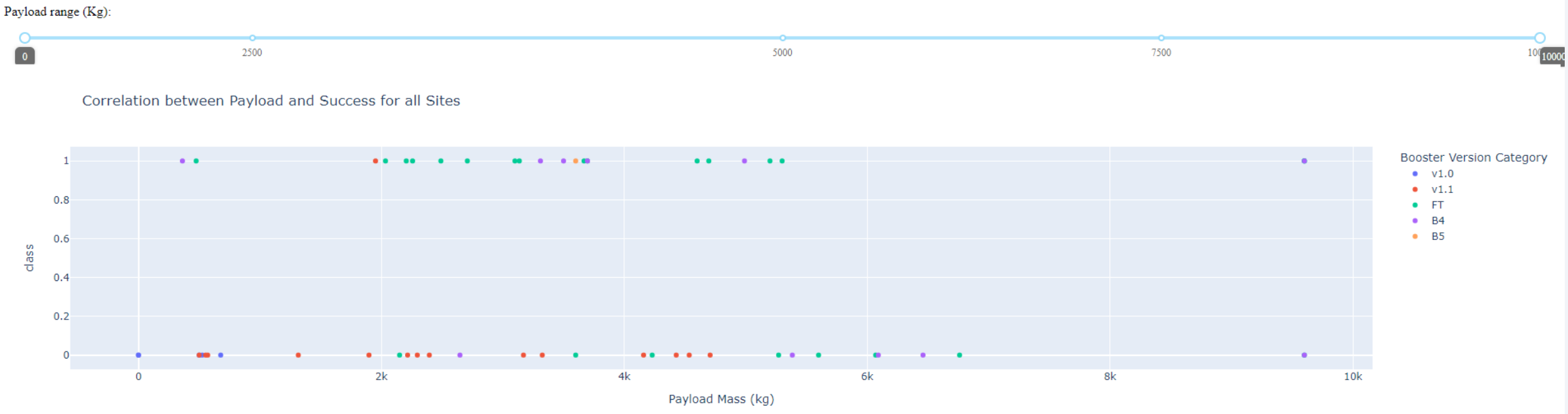
- The dropdown menu allows user to get success rate of all sites when All Sites is selected.
- Among all sites, KSC LC-39A (41.7%) has the best success launch rate followed by CCAPS LC-40 (29.2%), VAFB SLC-4E(16.7%) and CCAFS SLC-40 (12.5%)

Launch site with highest launch success ratio



- The dropdown menu allows user to get success rate of one site if any site is selected.
- KSC LC-39A has the highest launch success ratio of 76.9%.

Payload vs. Launch Outcome scatter plot for all sites

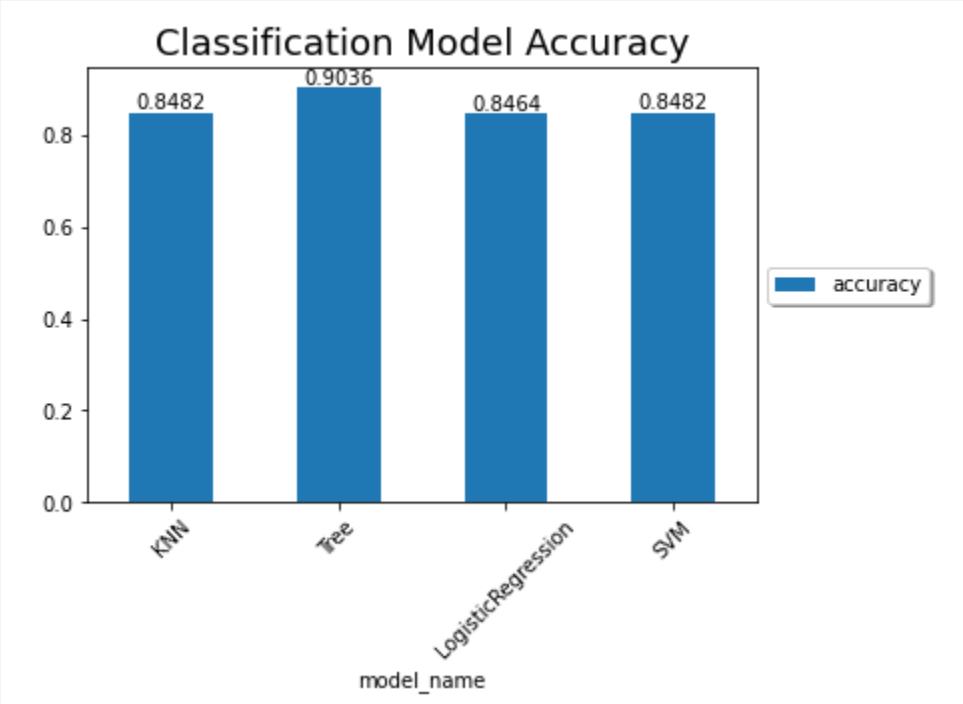


- Payload within range 2000~ 6000 has comparatively higher success rate.

Section 5

Predictive Analysis (Classification)

Classification Accuracy




```
Best Model is Tree with a score of 0.9036
Best Params is : {'criterion': 'gini',
                  'max_depth': 14,
                  'max_features': 'auto',
                  'min_samples_leaf': 4,
                  'min_samples_split': 2,
                  'splitter': 'random'}
```

Confusion Matrix

This confusion matrix of tree model with score of .9036

Due to the unbalanced data set, we need to consider F1 as well.

TN: 2 records of failure to land was predicted correctly;

FP: 4 records of success to land was wrongly predicted as failure

FN: 0

TP: 12 records of success landing was predicted correctly.

Accuracy= (TP+TN)/Total=(12+2)/18=.7778

Precision=TP/(TP+FP)=12/16=.75

Recall = TP/(TP+FN)=12/12=1

F1 score = 2*(precision*recall)/(precision + recall)=.857

Conclusions

- ES-L1, GEO, HEO, SSO has highest success rate
- Since 2013, we see a promising increasing success rate.
- Payload Mass is negatively correlated with success rate. When payload mass goes beyond 6K kg, we saw comparatively lower success rate.
- The best Machine learning model we can use to make prediction on success or failure launch is Decision tree model.

Appendix

Python code

- Data collection - scraping from HTML
- Purpose: Append list of data into dict by:
    1. enumerate table_number, table
    2. Check row.th and mark flag with flight_number
    3. Using flag to append list to dict

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            flight_number
            launch_dict['Flight No.'].append(flight_number)
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
#             print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            #print(date)
            launch_dict['Date'].append(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            #print(time)
            launch_dict['Time'].append(time)

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            print(bv)
            launch_dict['Version Booster'].append(bv)

            # Launch Site
            # TODO: Append the bv into launch_dict with key `Launch Site`
            launch_site = row[2].a.string
            #print(launch_site)
            launch_dict['Launch site'].append(launch_site)

            # Payload
            # TODO: Append the payload into launch_dict with key `Payload`
            payload = row[3].a.string
            #print(payload)
            launch_dict['Payload'].append(payload)
```

Appendix

Python code

- Predictive Analysis (Classification) logistic regression as an example
- Purpose: use GridSearchCV to obtain best parameters for the training model
    1. Initiate the model
    2. Call GridSearchCV by passing model, parameters and define cross validation folds
    3. Fit the GridSearchCV model with training data
    4. Call GridSearchCV .best_params_ and .best_score_ to get best parameters and score for CV

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

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

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv=GridSearchCV(lr,parameters,cv=10)
logreg_cv.fit(X_train,Y_train)
```

```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params\_` and the accuracy on the validation data using the data attribute `best_score\_`.

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

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

Appendix

Python code

- Predictive Analysis (Classification) collect and pick the best model
- Purpose: pick the best model from all models built:
  - logistic regression, Decision Tree, KNN, SVM

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_,'SVM':svm_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
if bestalgorithm == 'SVM':
    print('Best Params is :',svm_cv.best_score_)
```

```
Best Algorithm is Tree with a score of 0.9035714285714287
Best Params is : {'criterion': 'gini', 'max_depth': 14, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
```

Appendix

Python code

- Predictive Analysis (Classification) collect and pick the best model
- Purpose: visualize the model results
  - logistic regression, Decision Tree, KNN, SVM

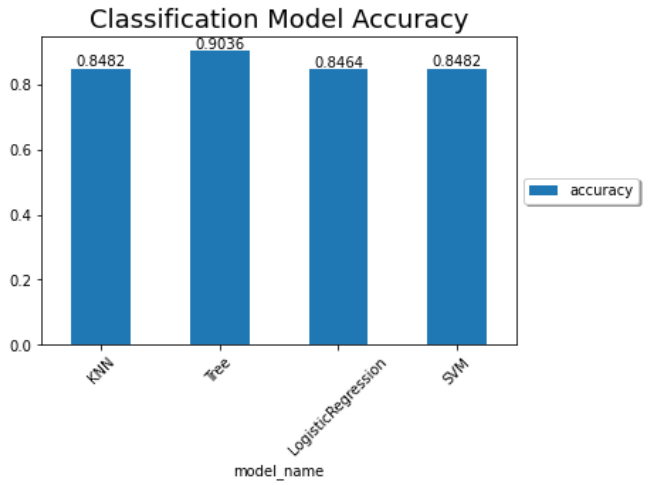
```
algorithms_df = pd.DataFrame(algorithms,index=[0]).T.reset_index()
algorithms_df = algorithms_df.rename(columns={'index':'model_name',0:'accuracy'})
algorithms_df['accuracy']=algorithms_df['accuracy'].round(4)
algorithms_df
```

	model_name	accuracy
0	KNN	0.8482
1	Tree	0.9036
2	LogisticRegression	0.8464
3	SVM	0.8482

```
ax=algorithms_df.plot.bar(x='model_name',y='accuracy')
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5), fancybox=True, shadow=True, ncol=3)
# iterate through the axes containers
for c in ax.containers:
    labels = [f'{v.get_height()}' for v in c]
    ax.bar_label(c, labels=labels, label_type='edge')

plt.xticks(rotation = 45)
plt_title= 'Classification Model Accuracy'
plt.title(plt_title, fontsize=18)
```

```
Text(0.5, 1.0, 'Classification Model Accuracy')
```



Thank you!