Winning Space Race with Data Science

Fabián Camilo Lozano Calderón
August 2023

Outline

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

Executive Summary

- Summary of methodologies

❑ Data Collection

❑ Data Collection with Web scraping

❑ Data Wrangling

❑ EDA using SQL

❑ EDA Data Visualization using Pandas

❑ Building interactive Dashboards

❑ Machine Learning Predictions

- Summary of all results

❑ EDA Querys and results

❑ Dashboards

❑ Predictive Analysis

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.

This information can be used if an alternate company wants to bid against SpaceX for a rocket launch

- Problems you want to find answers

Predict if the Falcon 9 first stage will land successfully

Section 1

Methodology

Methodology

Executive Summary

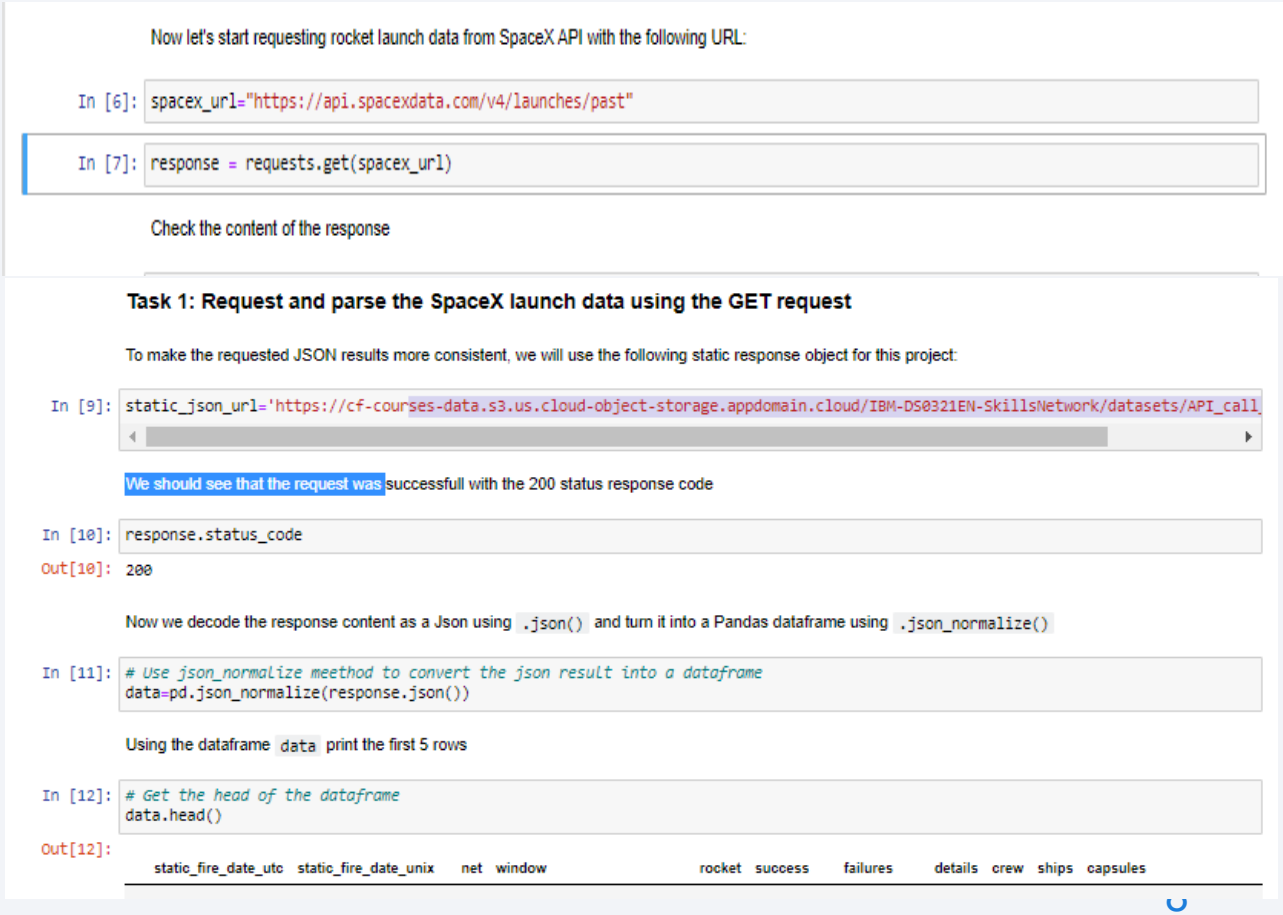
- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- The datasets were collected from the spacex site where they were abstracted through Restful_API. Filtering by rockets, launchpads, cores, launches. Passing that information to dataframes by pandas, then a dealed with missing values was done by replacing with average values.

Data Collection – SpaceX API

- The content was decoded from the response as a Json using .json() and converted to a Pandas dataframe using .json_normalize().
- Add the GitHub URL of the completed SpaceX API calls notebook, https://github.com/familo91/Capston-IBM-DataScience/blob/master/1.DataCollection/jupyter-labs-spacex-data-collection-api.ipynb



Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"
In [7]: response = requests.get(spacex_url)
```

Check the content of the response

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
In [9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call
```

We should see that the request was successfull with the 200 status response code

```
In [10]: response.status_code
Out[10]: 200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
In [11]: # Use json_normalize meethod to convert the json result into a dataframe
         data=pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

```
In [12]: # Get the head of the dataframe
         data.head()
Out[12]:
```

static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules

Data Collection - Scraping

- The extraction of data from the falcon 9 launch table was done by webscraping a wikipedia URL using a beatiful soup object, the columns of the frame and its content were organized by means of the append instruction creating a dataframe.
- Add the GitHub URL of the completed web scraping notebook, https://github.com/familo91/Capston-IBM-DataScience/blob/master/2.Data_Wrangling/labs-jupyter-spacex-Data%20wrangling.ipynb

```
In [68]: 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

            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)
            datatimelist=date_time(row[0])

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

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

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

Data Wrangling

- We used pandas and numpy to find out the success rate, we used functions such as value_counts and mean, we created a column to determine the land succesfully in a boolean way.
- Add the GitHub :https://github.com/familo91/Capston-IBM-DataScience/blob/master/2.Data_Wrangling/labs-jupyter-spacex-Data%20wrangling.ipynb

In [18]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

df['Class']=df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
df['Class'].value_counts()
```

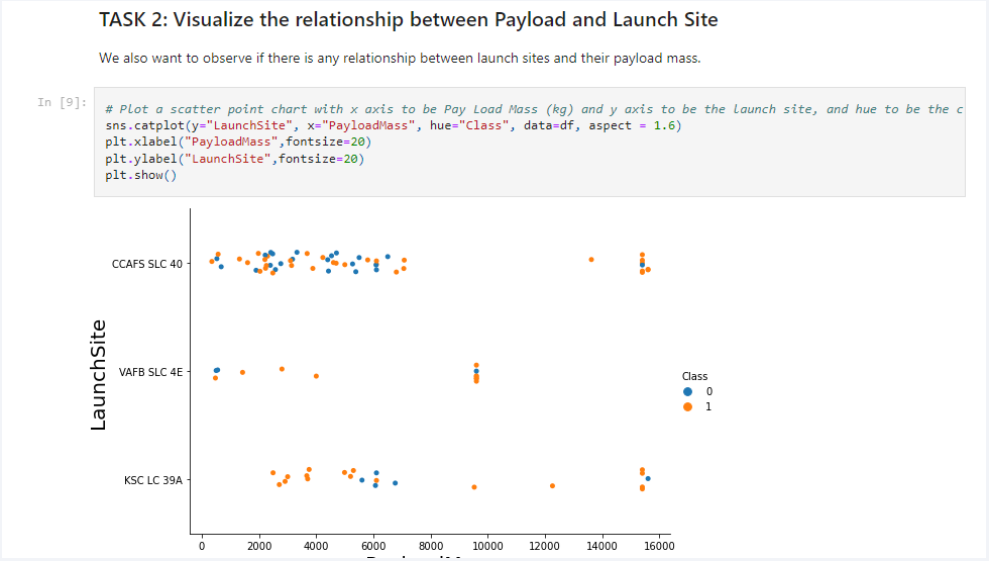
Out[18]:
```
1    60
0    30
Name: Class, dtype: int64
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

In [22]:
```
landing_class=df['Class']
df[['Class']].head(8)
```

EDA with Data Visualization

- The relationship between the number of flight and launches, the success rate of each type of orbit and the annual launch success are verified by pyplot. Using scatter point chart and bar chart
- Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose: https://github.com/familo91/Capston-IBM-DataScience/blob/master/5.%20EDA%20-Visualization%20Tools/jupyter-labs-eda-dataviz%20(1).ipynb



EDA with SQL

- In this case the csv file was uploaded in IBM CLOUD and the database was created and queries were performed with searches for weight, success ground pad and launch dates.

- Add the GitHub URL:https://github.com/familo91/Capston-IBM-DataScience/blob/master/4.EDA-SQL/jupyter-labs-eda-sql-

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)='2017' for year.

In [27]:
```
%sql SELECT substr(date_,6,2) as MONTH,DATE_,BOOSTER_VERSION,LAUNCH_SITE,LANDING_OUTCOME FROM SPACEXTBL WHERE substr(
```

```
 * ibm_db_sa://qxc09986:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb;security=SSL
Done.
```

Out[27]:

MONTH	date_	booster_version	launch_site	landing_outcome
01	2017-01-14	F9 FT B1029.1	VAFB SLC-4E	Success (drone ship)
02	2017-02-19	F9 FT B1031.1	KSC LC-39A	Success (ground pad)
03	2017-03-16	F9 FT B1030	KSC LC-39A	No attempt
03	2017-03-30	F9 FT B1021.2	KSC LC-39A	Success (drone ship)
01	2017-01-05	F9 FT B1032.1	KSC LC-39A	Success (ground pad)
05	2017-05-15	F9 FT B1034	KSC LC-39A	No attempt
03	2017-03-06	F9 FT B1035.1	KSC LC-39A	Success (ground pad)
06	2017-06-23	F9 FT B1029.2	KSC LC-39A	Success (drone ship)
06	2017-06-25	F9 FT B1036.1	VAFB SLC-4E	Success (drone ship)
05	2017-05-07	F9 FT B1037	KSC LC-39A	No attempt
08	2017-08-14	F9 B4 B1039.1	KSC LC-39A	Success (ground pad)

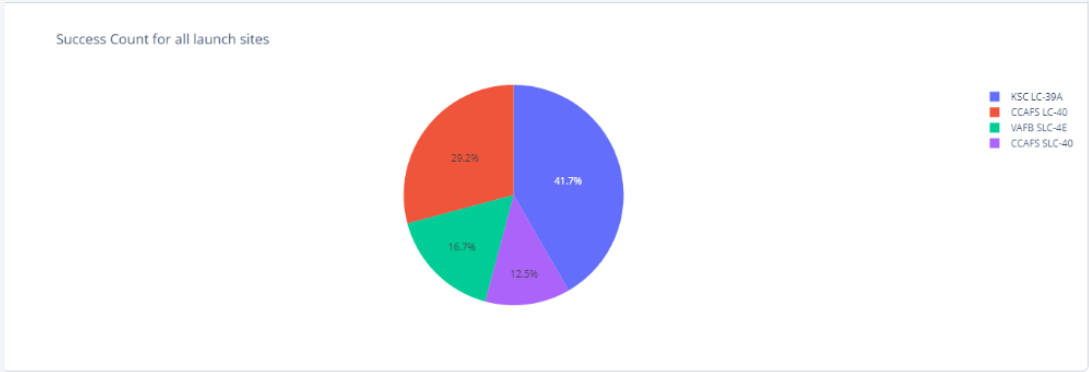
Build an Interactive Map with Folium

- Each one of the launches was positioned according to its longitude and latitude, this allowed us to have clarity on which launches and where they had a higher success rate, finally we calculated the distance of the launches with respect to the coast.
- Add the GitHub URL: https://github.com/familo91/Capston-IBM-DataScience/blob/master/6.Interactive%20Visual%20Analytics/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- A dashboard was created so that each launch could be dynamically filtered by Payload range, success count launch, site, etc.

- Add the GitHub URL:https://github.com/familo91/Capston-IBM-DataScience/blob/master/7.Build%20and%20interactive%20Dash/spacex_dash_app.py



Predictive Analysis (Classification)

- The class column was assigned as variable Y and as X the URL dataframe, the network was trained with 20% of the data, the methods of Logist_reg,SVM,Desicion tree,KNN were used to find the best training parameters and the accuracy according to the score the winner is *Tree* with 0.8767
- Add the GitHub URL:https://github.com/familo91/Capston-IBM-DataScience/blob/master/8.%20Complete_Machine_Learning/module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Out[42]:

	Train Data Accuracy
LogisticRegression	0.846429
SVM	0.848214
KNN	0.848214
Tree	0.876786

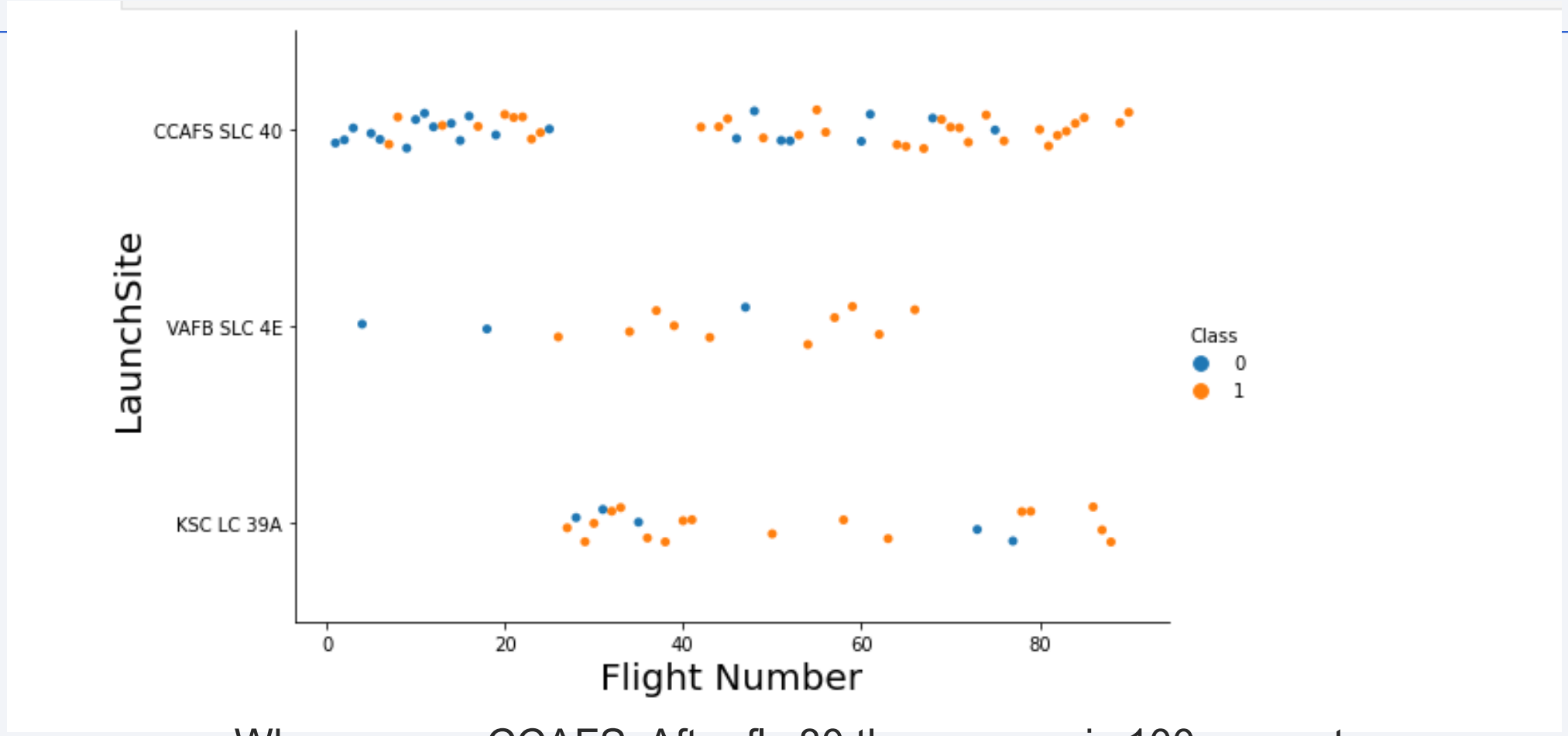
Results

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

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

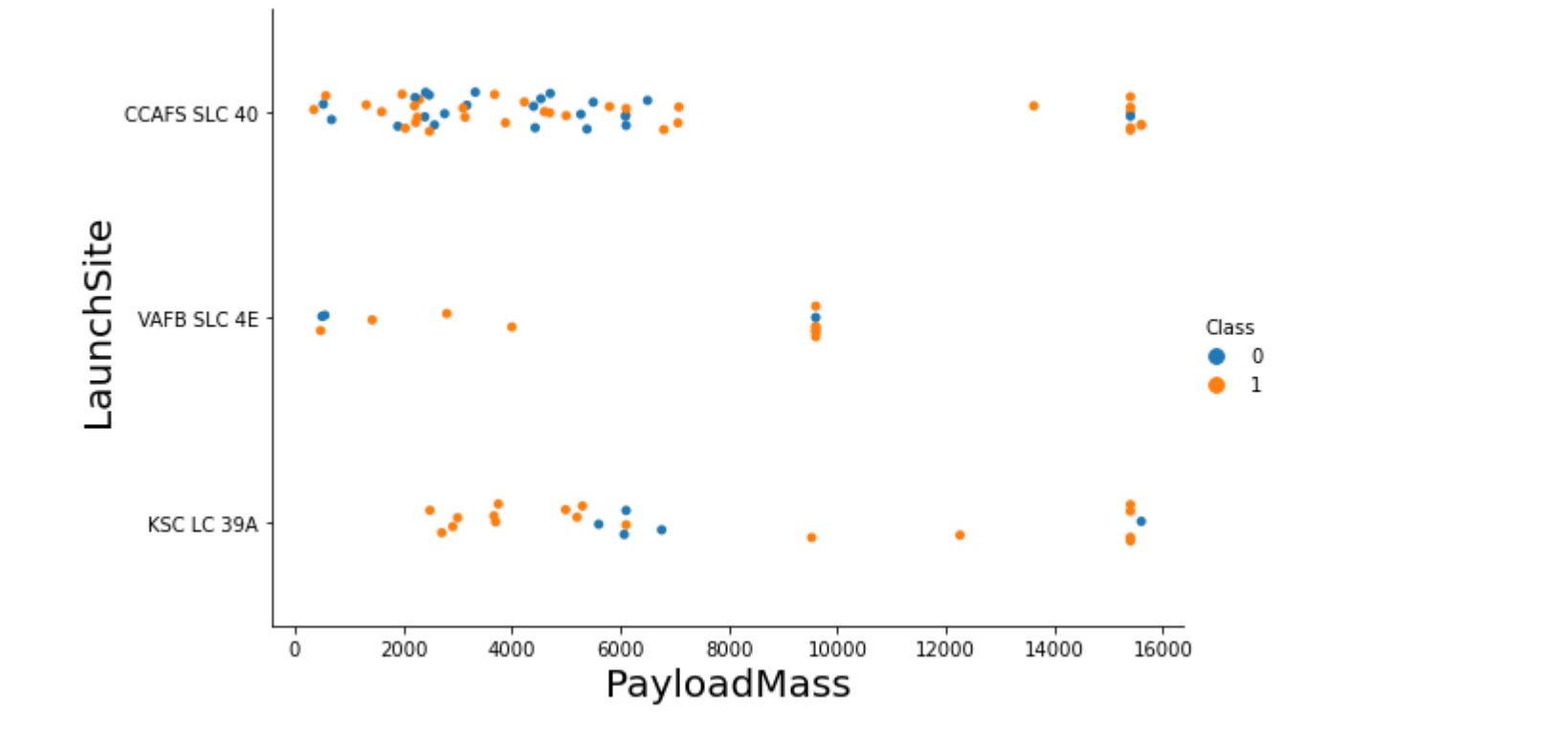


Whe can see CCAFS: After fly 80 the success is 100 percent

Whe can see VAFB: After fly 50 the success is 100 percent

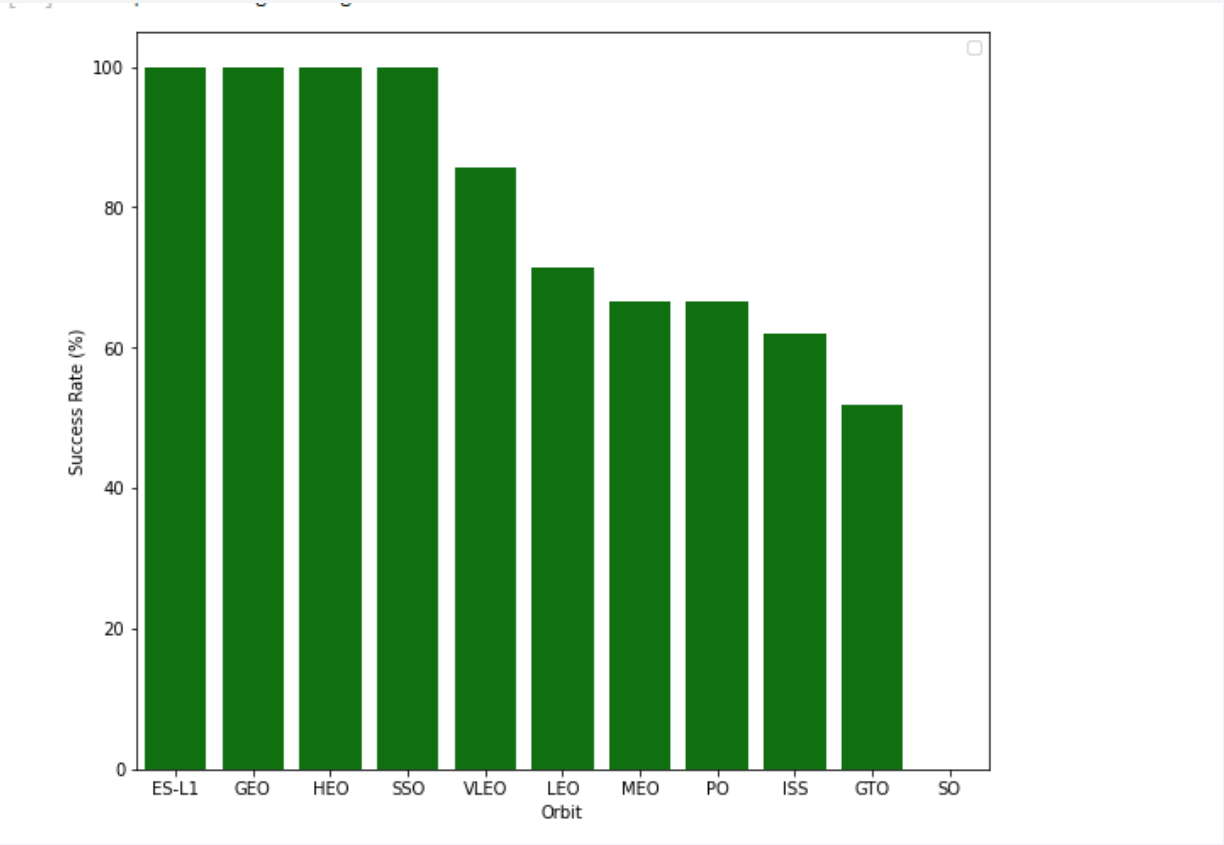
Whe can see KSC LC Before fly 80 the success is 100 percent

Payload vs. Launch Site



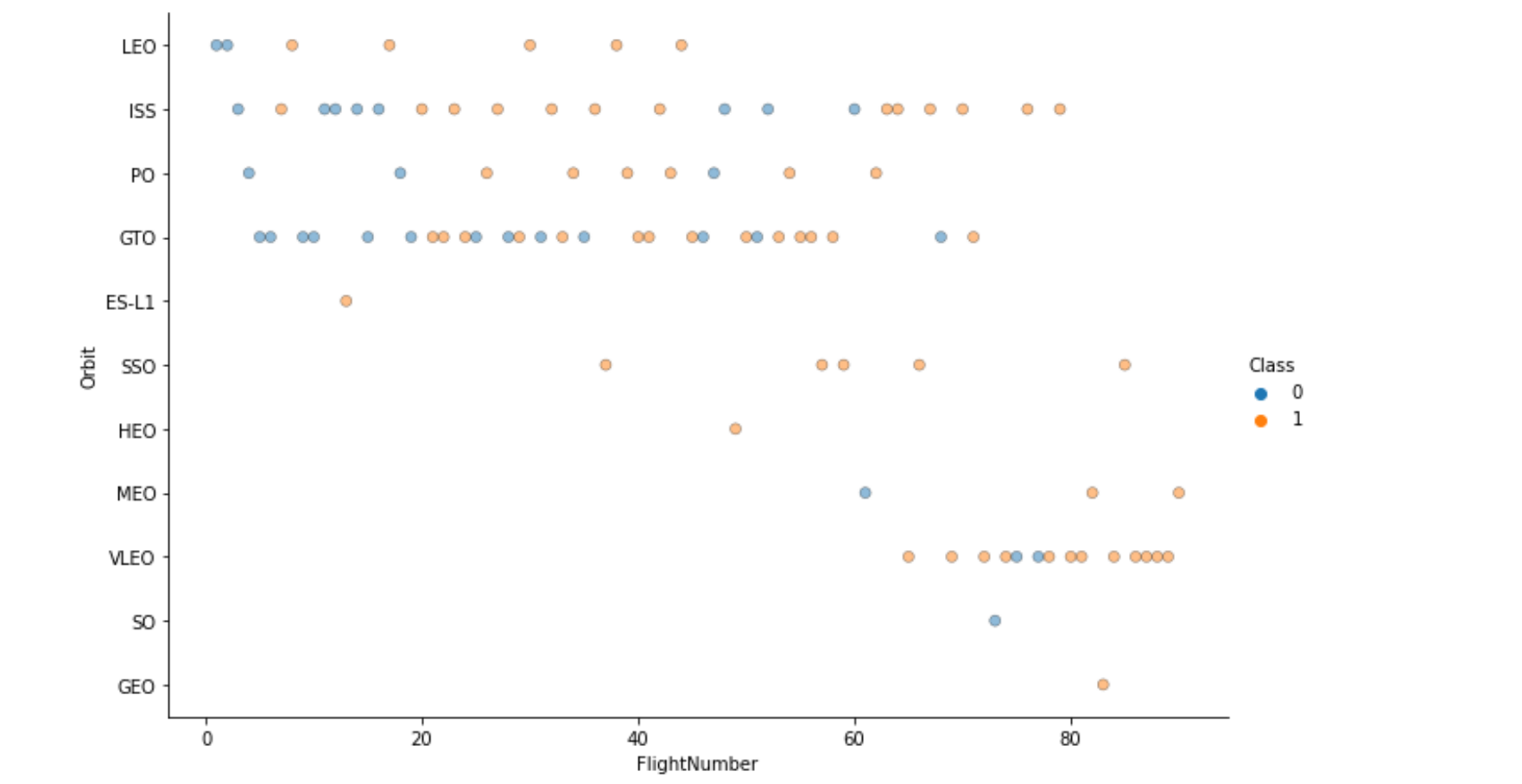
- Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type



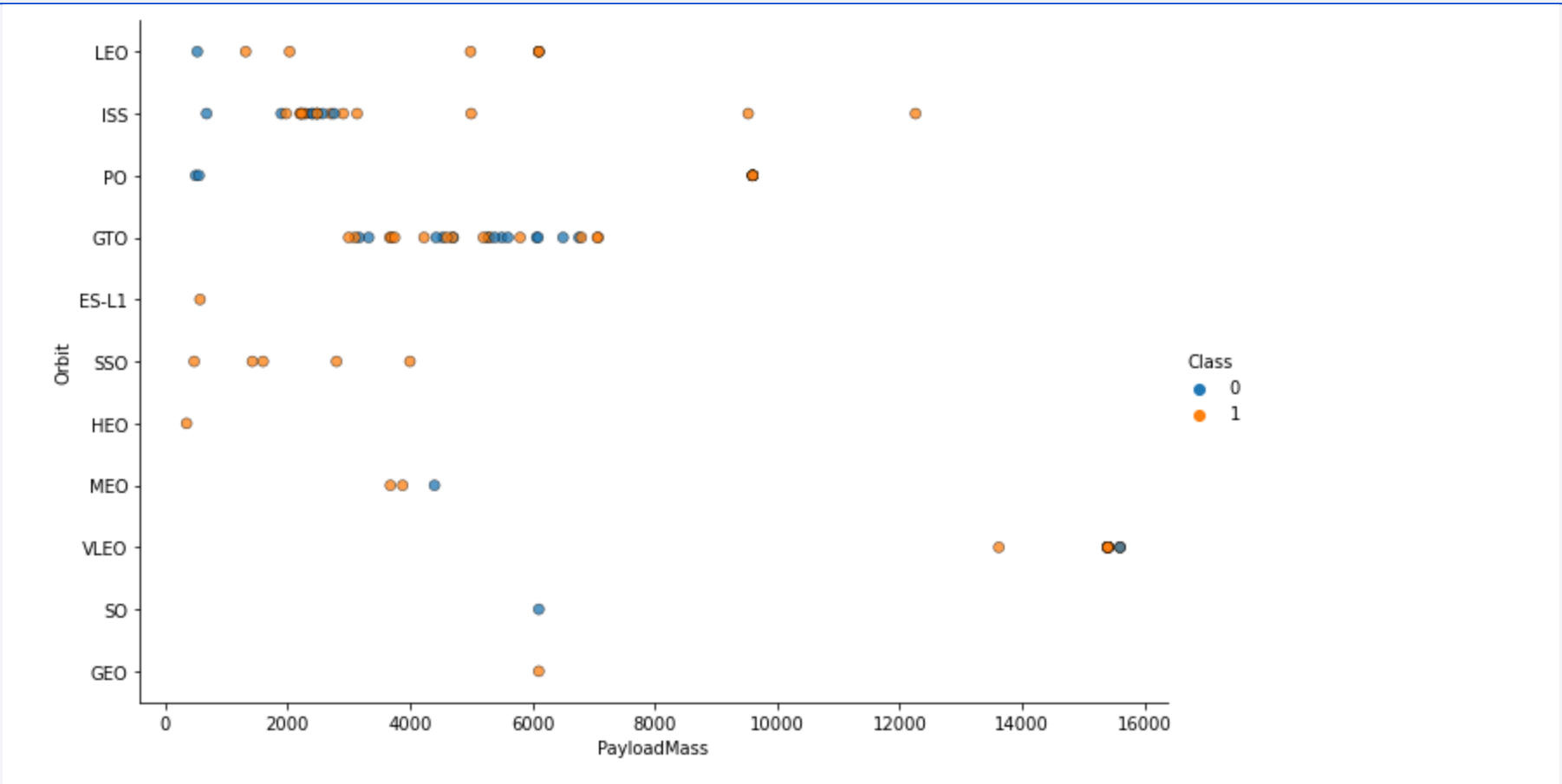
- the orbits with 100% success rate are ESL1, GEO, HEO, SSO, the lowest success rate is GTO.

Flight Number vs. Orbit Type



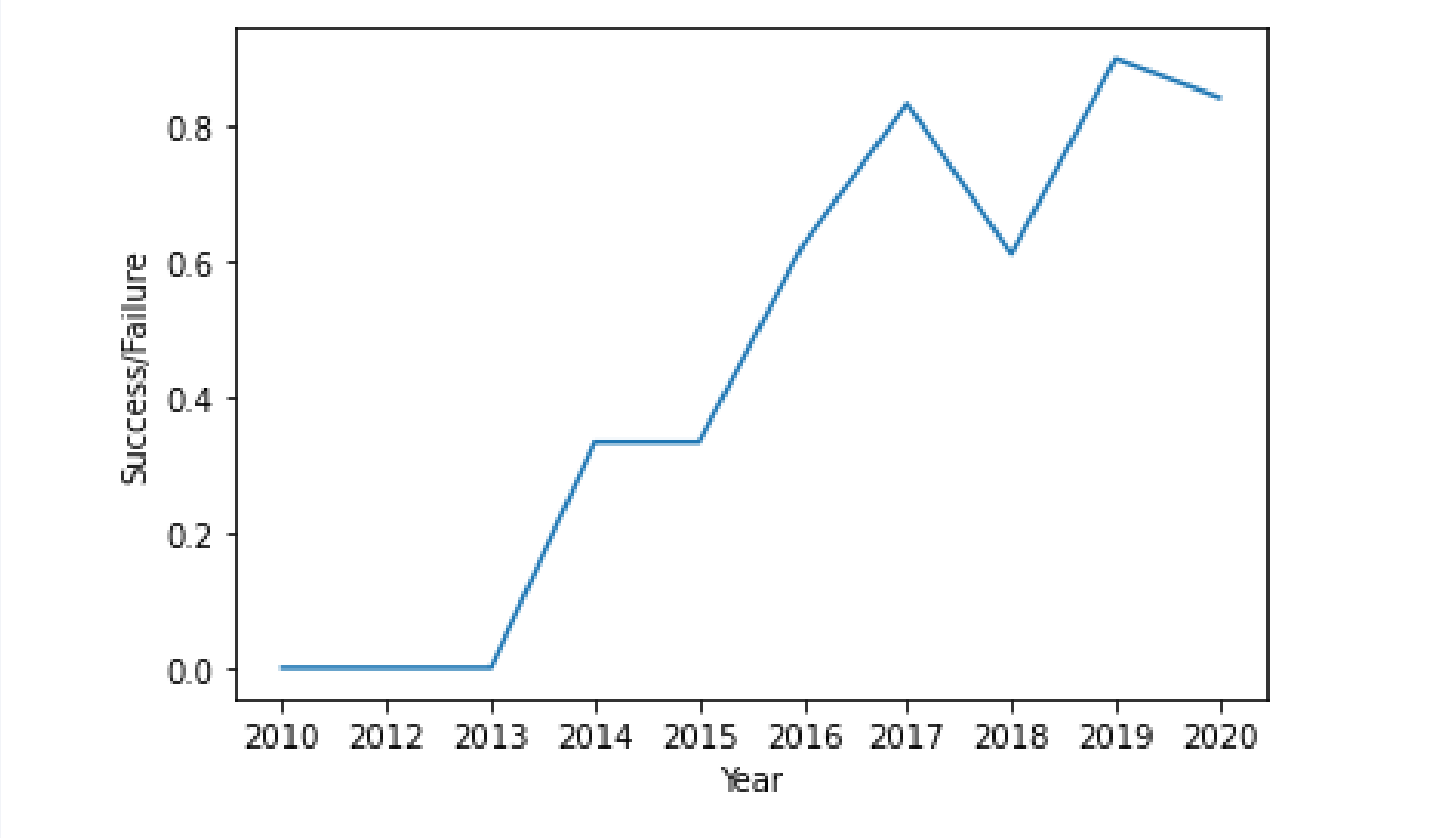
You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

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

```
In [4]: %sql select Unique(LAUNCH_SITE) from SPACEXTBL;
 * ibm_db_sa://qxc09986:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb;security=SSL
Done.
```

Out[4]:

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

- For this case I used a select with unique search for not repeat the result

Launch Site Names Begin with 'KSC'

Task 2

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

In [5]:
```
%sql select LAUNCH_SITE from SPACEXTBL WHERE (LAUNCH_SITE) LIKE 'KSC%' LIMIT 5;
```

```
 * ibm_db_sa://qxc09986:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb;security=SSL
Done.
```

Out[5]:

launch_site
KSC LC-39A
KSC LC-39A
KSC LC-39A
KSC LC-39A
KSC LC-39A

For these exercise you have to use LIKE to search the letters and a Limit for the quantity of searchs

Total Payload Mass

Task 3

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

```
In [6]: %sql select SUM(PAYLOAD_MASS__KG_) as sum from SPACEXTBL WHERE (CUSTOMER) = 'NASA (CRS)'
```

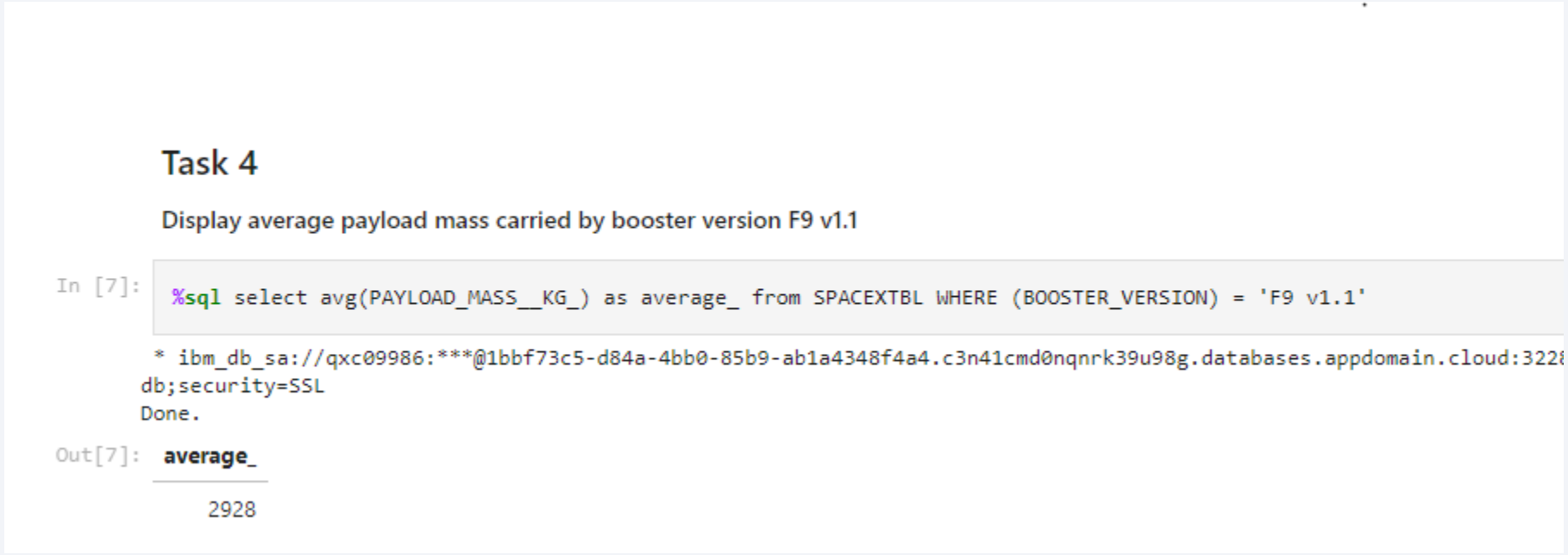
```
 * ibm_db_sa://qxc09986:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb;security=SSL
Done.
```

Out[6]:

SUM
45596

- You have to use math operation of sum with clause in column customer, 45596 KG is the total payload used for the customer NASA

Average Payload Mass by F9 v1.1



Task 4

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

In [7]:
```
%sql select avg(PAYLOAD_MASS__KG_) as average_ from SPACEXTBL WHERE (BOOSTER_VERSION) = 'F9 v1.1'
```

```
 * ibm_db_sa://qxc09986:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:322
db;security=SSL
Done.
```

Out[7]:

average_
2928

- The average payload mass carried by booster version F9 v1.1 is 2928Kg

First Successful Ground Landing Date

Task 5

List the date where the succesful landing outcome in drone ship was acheived.

Hint:Use min function

In [10]:
```
%sql SELECT min(date_) from SPACEXTBL where landing_outcome = 'Success (drone ship)'
```

```
 * ibm_db_sa://qxc09986:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb;security=SSL
Done.
```

Out[10]:

1
2016-05-27

- The successful landing outcome was 2016-05-27

Successful Drone Ship Landing with Payload between 4000 and 6000

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

In [12]:
```
%sql select BOOSTER_VERSION from SPACEXTBL where LANDING_OUTCOME='Success (ground pad)'and PAYLOAD_MASS__KG_ BETWEEN 40
```

```
 * ibm_db_sa://qxc09986:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/blu
db;security=SSL
Done.
```

Out[12]:

booster_version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

- For this query is necessary use the function between in the conditional

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

In [13]:
```
%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS OUTCOME FROM SPACEXTBL GROUP BY MISSION_OUTCOME
```

```
 * ibm_db_sa://qxc09986:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb;security=SSL
Done.
```

Out[13]:

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

- The total of success was 99 , the failure was 1, for this query a group was used.

Boosters Carried Maximum Payload

```
In [17]: %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
 * ibm_db_sa://qxc09986:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb;security=SSL
Done.
```

Out[17]:

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

- For this query Max function was used for filter booster version

2017 Launch Records

In [27]:
```
%sql SELECT substr(date_,6,2) as MONTH,DATE_,BOOSTER_VERSION,LAUNCH_SITE,LANDING_OUTCOME FROM SPACEXTBL WHERE substr(date_,
```

```
 * ibm_db_sa://qxc09986:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb;se
curity=SSL
Done.
```

Out[27]:

MONTH	date_	booster_version	launch_site	landing_outcome
01	2017-01-14	F9 FT B1029.1	VAFB SLC-4E	Success (drone ship)
02	2017-02-19	F9 FT B1031.1	KSC LC-39A	Success (ground pad)
03	2017-03-16	F9 FT B1030	KSC LC-39A	No attempt
03	2017-03-30	F9 FT B1021.2	KSC LC-39A	Success (drone ship)
01	2017-01-05	F9 FT B1032.1	KSC LC-39A	Success (ground pad)
05	2017-05-15	F9 FT B1034	KSC LC-39A	No attempt
03	2017-03-06	F9 FT B1035.1	KSC LC-39A	Success (ground pad)
06	2017-06-23	F9 FT B1029.2	KSC LC-39A	Success (drone ship)
06	2017-06-25	F9 FT B1036.1	VAFB SLC-4E	Success (drone ship)
05	2017-05-07	F9 FT B1037	KSC LC-39A	No attempt
08	2017-08-14	F9 B4 B1039.1	KSC LC-39A	Success (ground pad)
08	2017-08-24	F9 FT B1038.1	VAFB SLC-4E	Success (drone ship)
07	2017-07-09	F9 B4 B1040.1	KSC LC-39A	Success (ground pad)
09	2017-09-10	F9 B4 B1041.1	VAFB SLC-4E	Success (drone ship)
11	2017-11-10	F9 FT B1031.2	KSC LC-39A	Success (drone ship)
10	2017-10-30	F9 B4 B1042.1	KSC LC-39A	Success (drone ship)
12	2017-12-15	F9 FT B1035.2	CCAFS SLC-40	Success (ground pad)
12	2017-12-23	F9 FT B1036.2	VAFB SLC-4E	Controlled (ocean)

- For this query, function must be used **substr(Date, x, y) to filter month**

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

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

In [62]:
```
%sql SELECT LANDING_OUTCOME, COUNT(*) AS COUNT_LAUNCHES FROM SPACEXTBL WHERE LANDING_OUTCOME='Failure (drone ship)' OR LAND
```

```
 * ibm_db_sa://qxc09986:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb;se
curity=SSL
Done.
```

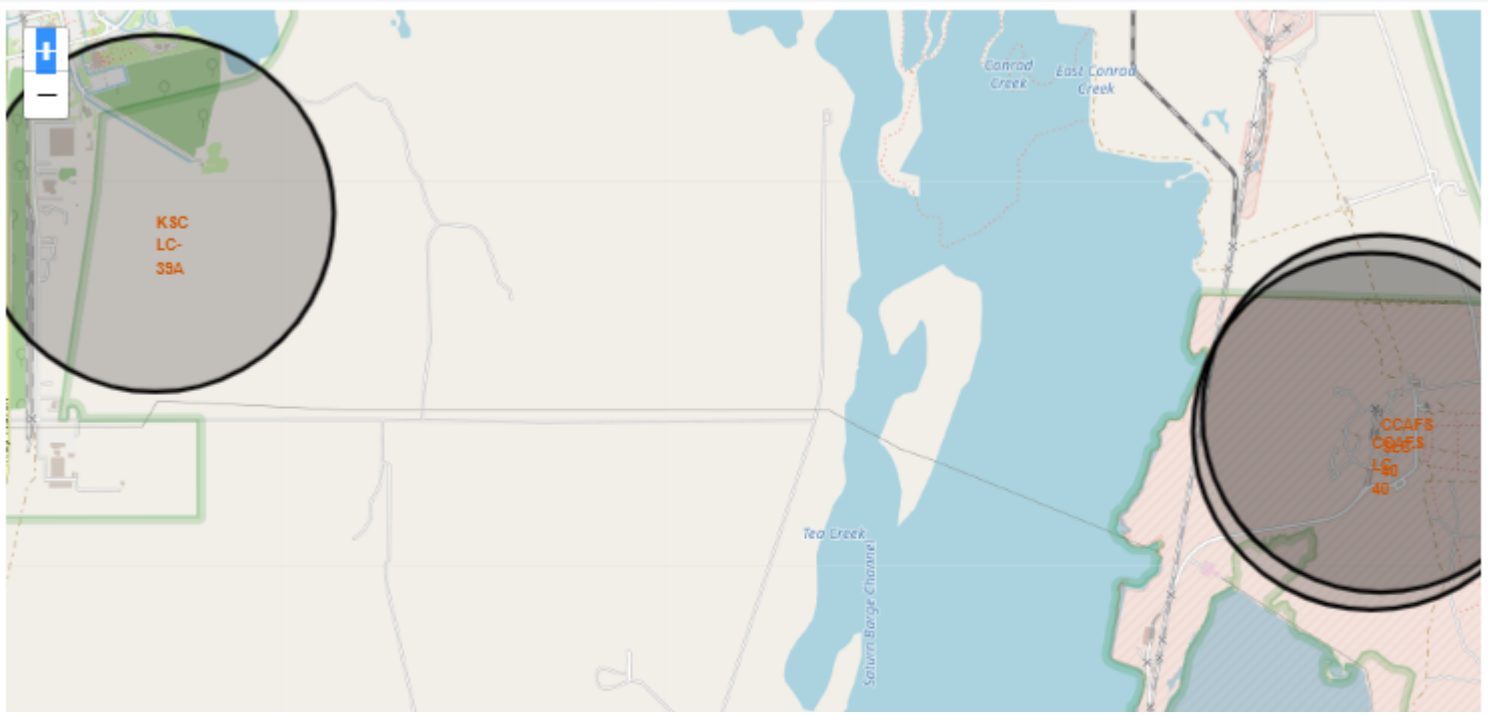
Out[62]:

landing_outcome	count_launches
Failure (drone ship)	5
Success (ground pad)	5

- For this query filter by date with the same function as last example, and filter by column failure (drone ship) and success (ground pad) name counting with desc function


Section 3
Launch Sites
Proximities Analysis

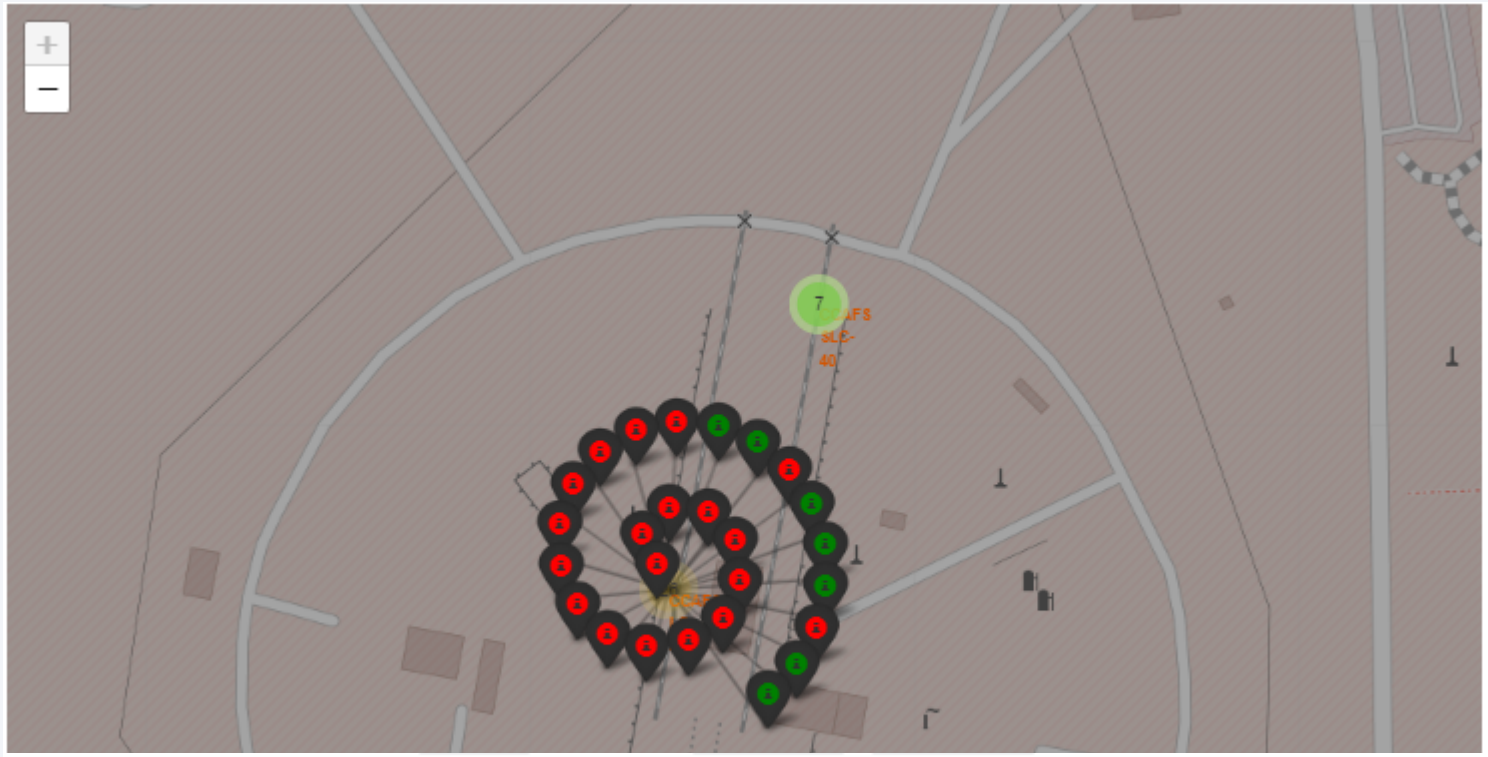
<Launches sites>




```
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as a p
for index ,x in launch_sites_df.iterrows():
    coor= [x['Lat'],x ['Long']]
    folium.Circle(coor, radius=1000, color='#000000', fill=True).add_child(folium.Popup(x['Launch Site'])).add_to(site_map)
    folium.map.Marker(coor, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0), html='<div style="font-size: 12; color:#d35400;"><
site_map
```

- With folium.circle and folium.map.Marker using the coordinates is possible draw in the map the launches sites

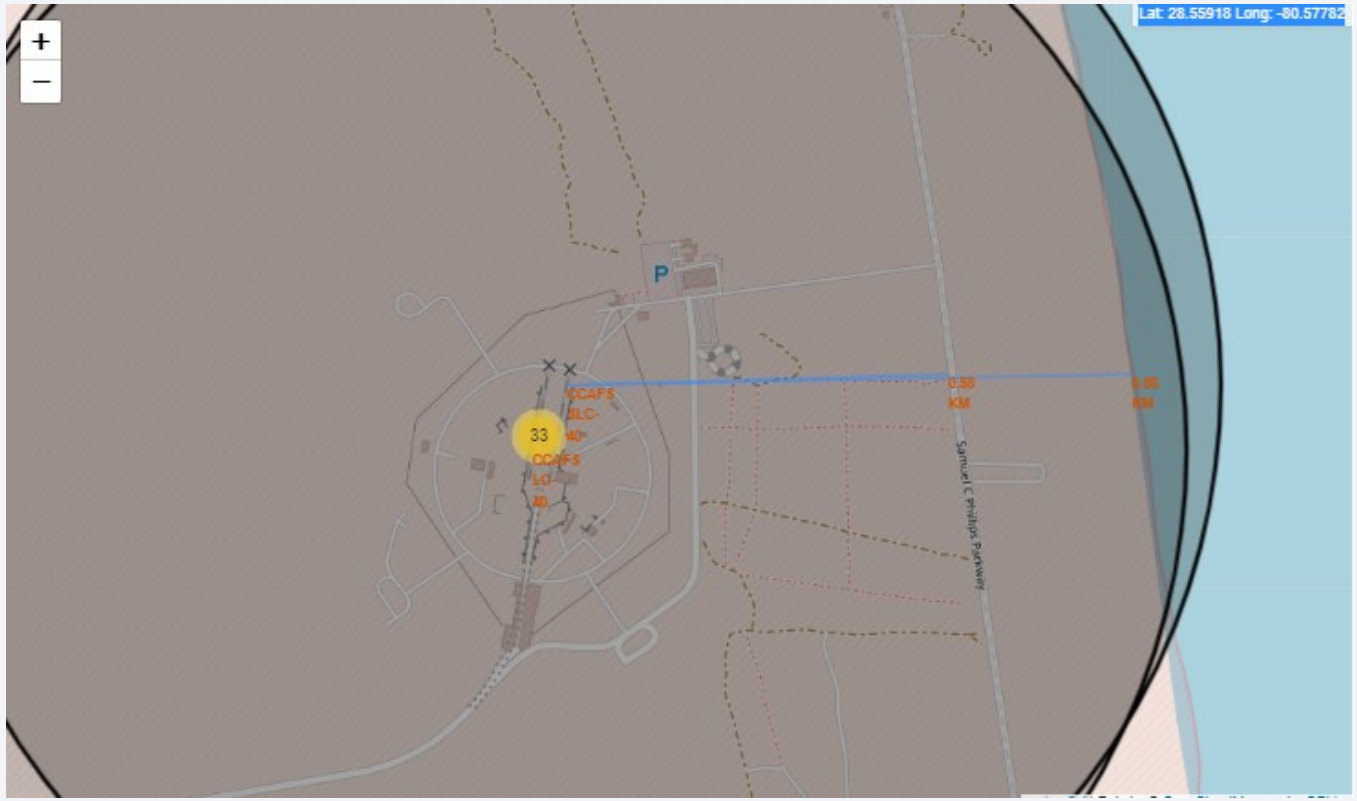
<Success/Failed Launches>



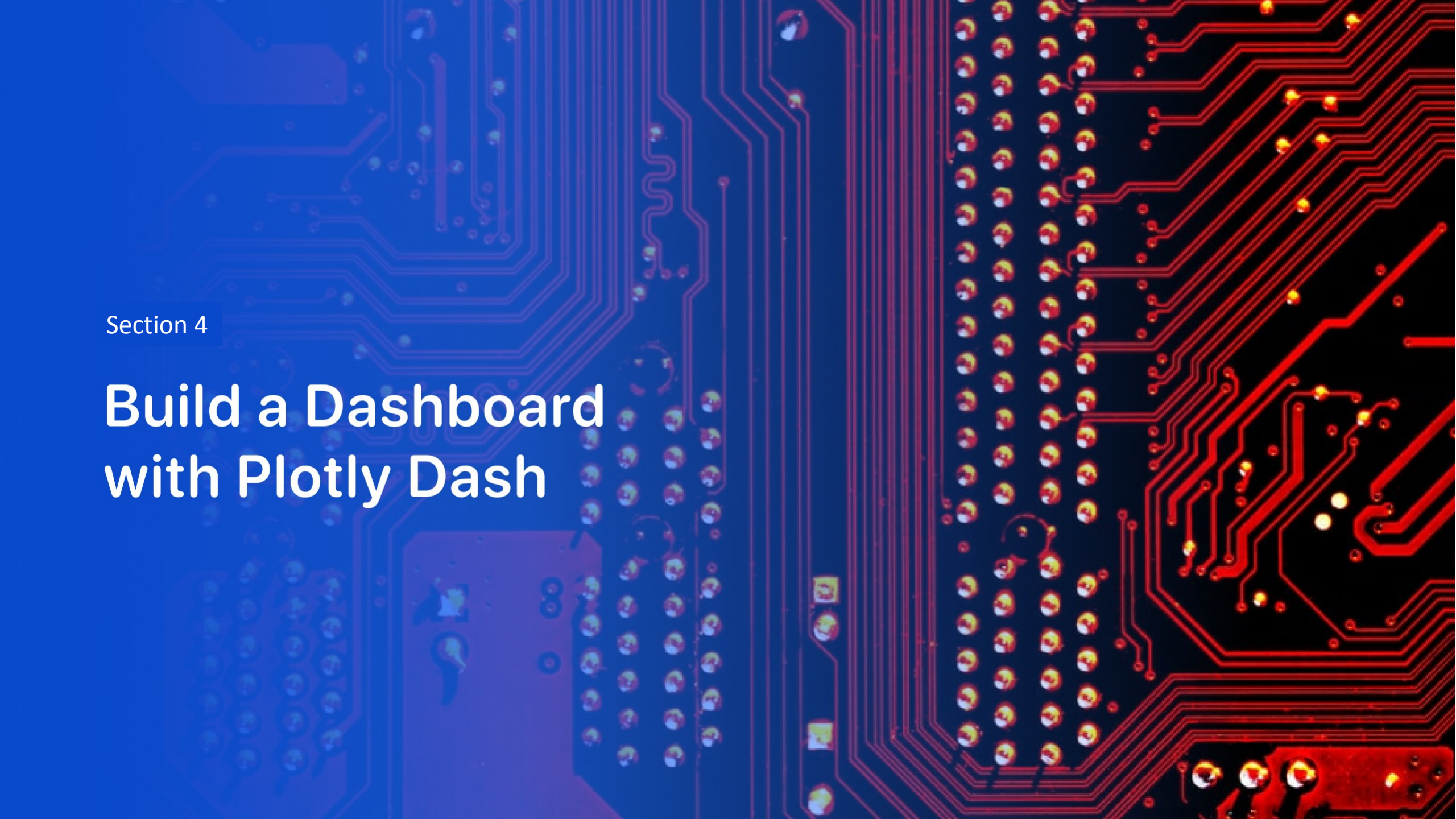

```
In [38]: # Add marker_cluster to current site_map
site_map.add_child(marker_cluster)

# for each row in spacex_df data frame
# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch was successed or failed,
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
for index, record in spacex_df.iterrows():
    # TODO: Create and add a Marker cluster to the site map
    # marker = folium.Marker(...)
    coordinate= [record['Lat'],record['Long']]
    folium.map.Marker(coordinate, icon=folium.Icon(color='black',icon_color=record['marker_color'])).add_to(marker_cluster)
site_map
```

From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates

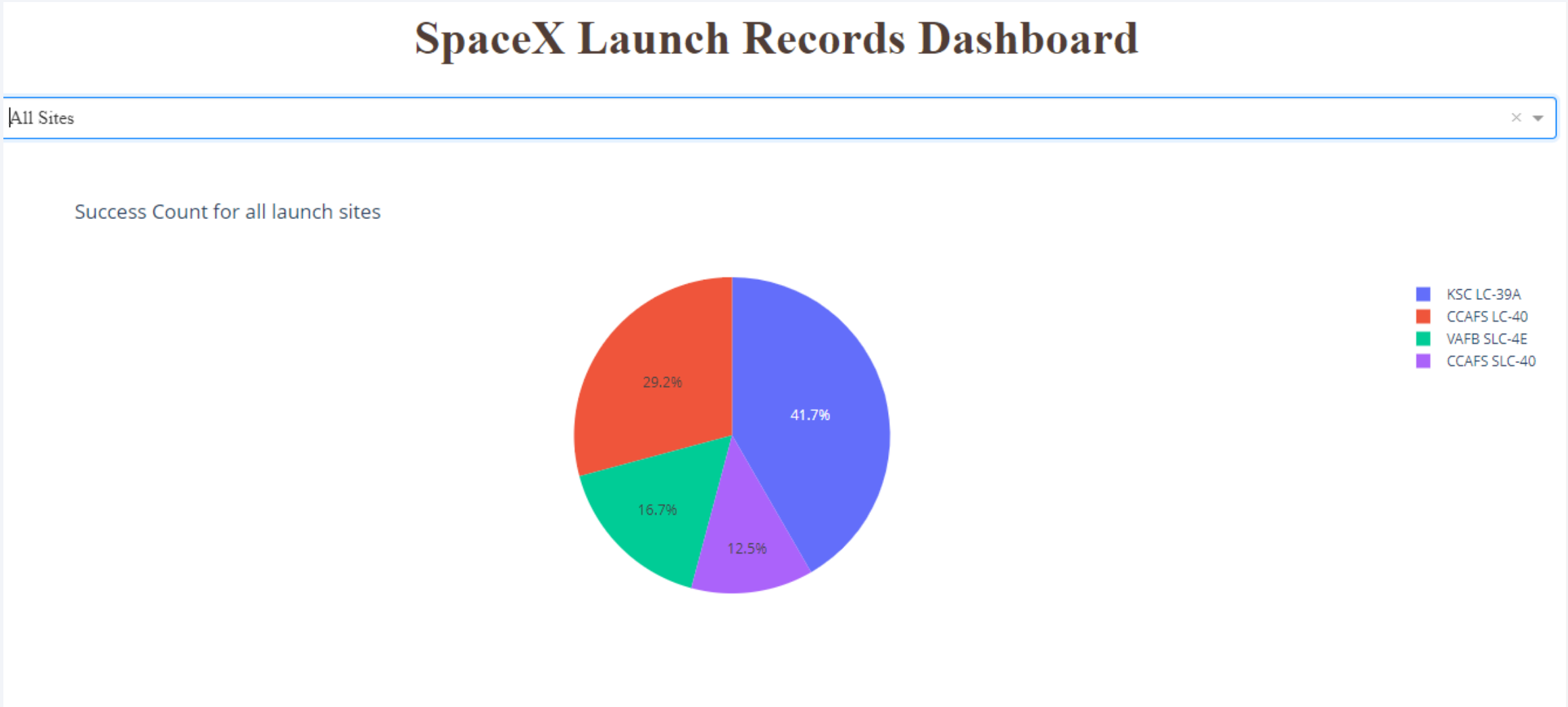
<Distance between launch sites>



Calculations can be made with other distance coordinates, in this case the calculation to the coastline was 0.89 km.

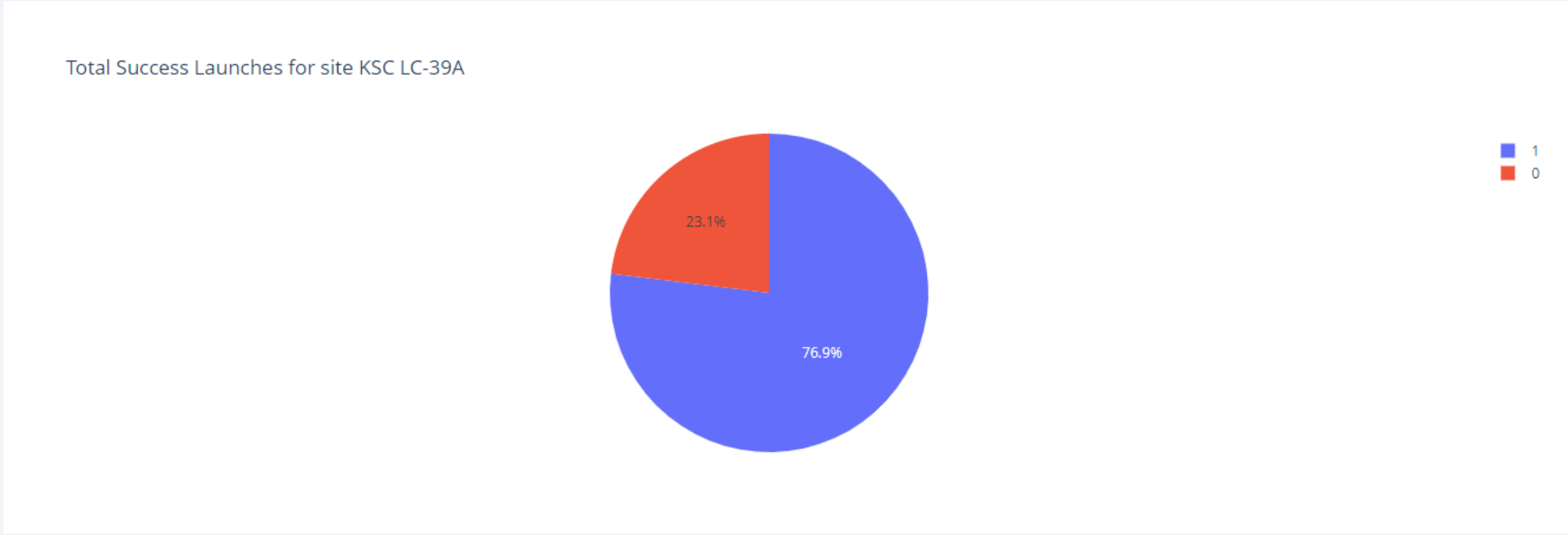

Section 4
Build a Dashboard with Plotly Dash

<Success Count>



- KSCLC39 had the major number of success count, followed by CCAFS LC-40

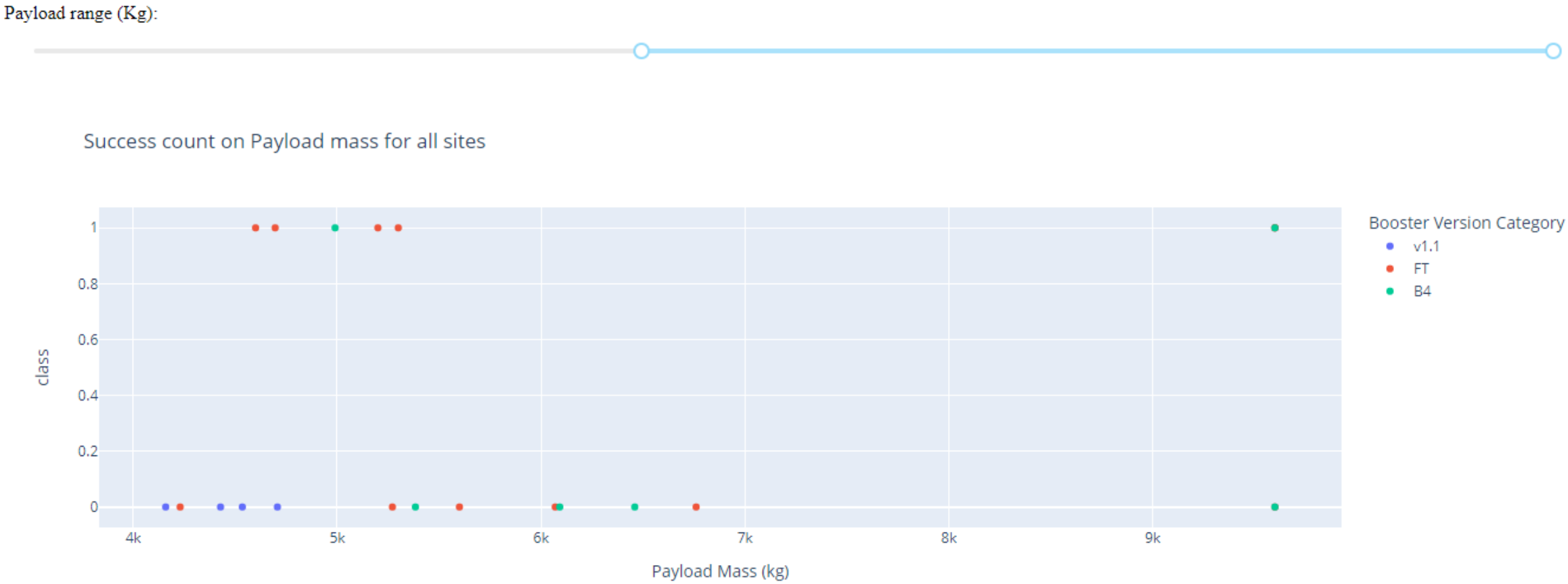
<Total Success Launches KSC-39A>



The percent of success for KSC-LC 39A was 79.6%

< Payload Mass vs Booster Version >

<Payload Mass vs Booster Version>



Above 5000kg the unique version that found was V1.1 ,FT, B4. For 10000 kg the version that support this load is B4

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The model of best classification of accuracy is tree .

TASK 12

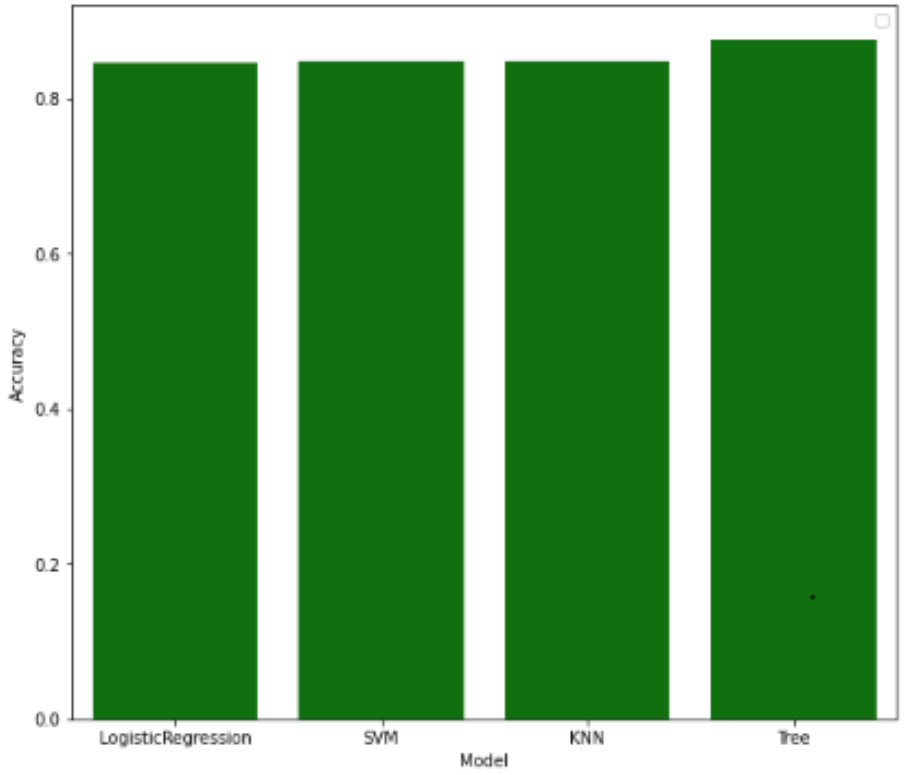
Find the method performs best:

In [48]:
```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_,'SVM':svm_cv.best_s
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])

score_df = pd.DataFrame.from_dict(algorithms, orient='index', columns=['Train Data Accuracy'])
score_df.sort_values(['Train Data Accuracy'], inplace=True)
score_df.head(5)

score_df = score_df.reset_index()
score_df.rename(columns = {'index': 'Algorithm'}, inplace = True)
score_df.head(5)
```

```
Best Algorithm is Tree with a score of 0.8767857142857143
```

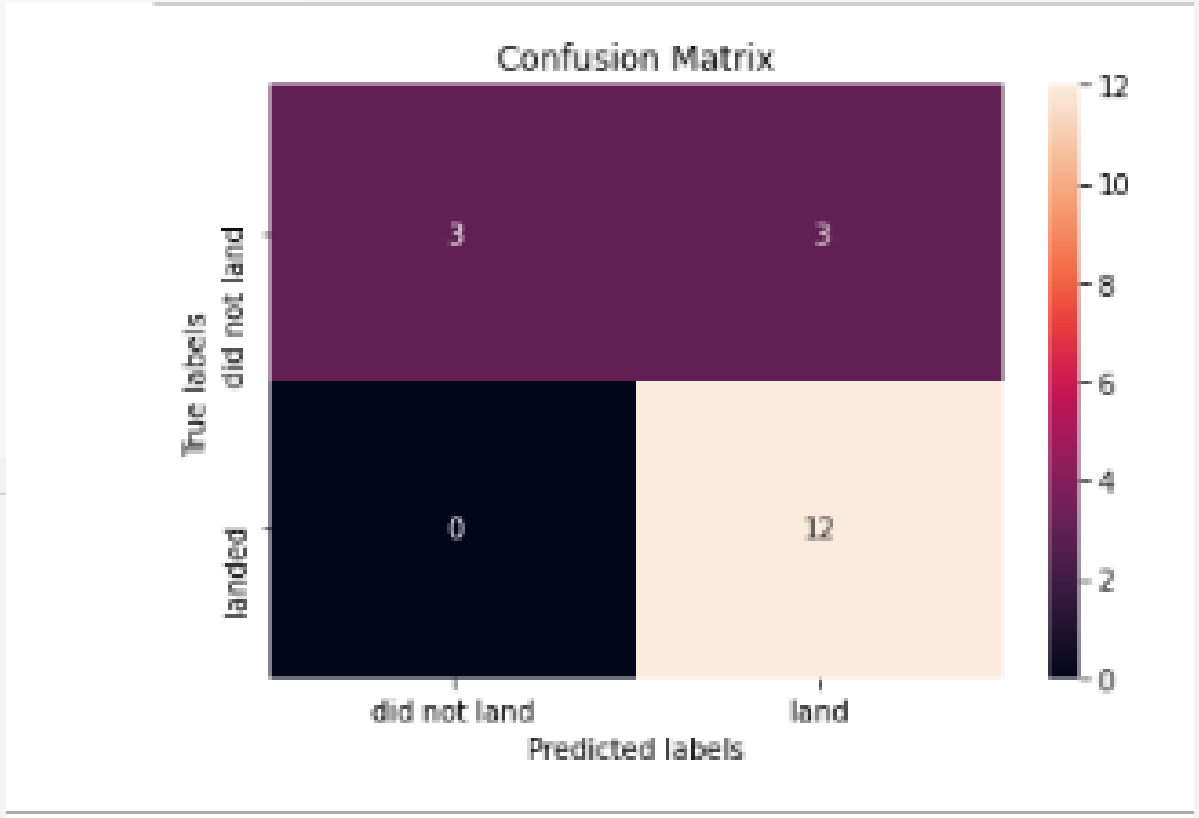
Out[48]:

	Algorithm	Train Data Accuracy
0	LogisticRegression	0.846429
1	SVM	0.848214
2	KNN	0.848214
3	Tree	0.876786



Confusion Matrix

```
#tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf':

parameters = {'criterion': ['gini'],
      'splitter': ['random'],
      'max_depth': [4],
      'max_features': ['auto'],
      'min_samples_leaf': [2],
      'min_samples_split': [2]}

tree = DecisionTreeClassifier()

tree_cv = GridSearchCV(tree, parameters, cv= 10)
#Fit the training data into the GridSearch object
tree_cv.fit(X_train, Y_train)

yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
plt.show()
```

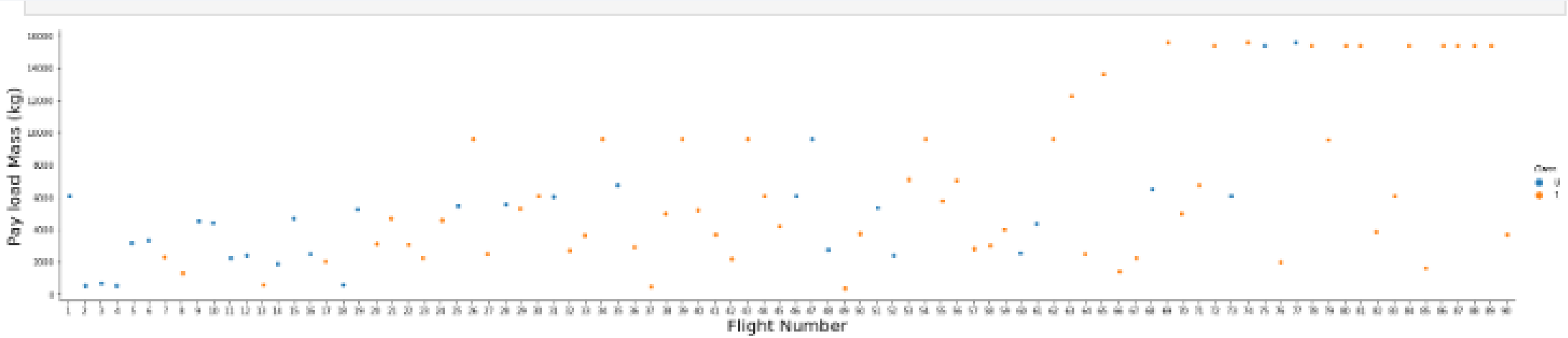


- With the best parameters, we can see that the sum of 20% of training dates is 18, we have 12 True positives, 3 True negatives, 3 falses Positives, 0 False negative

Conclusions

- The Falcon 9 first stage will land successfully
- For 10000 kg the version that support this load is B4
- The percent of success for KSC-LC 39A was 79.6%
- KSCLC39 had the major number of success count, followed by CCAFS LC-40
- You can observe that the sucess rate since 2013 kept increasing till 2020
- The orbits with 100% success rate are ESL1, GEO, HEO, SSO, the lowest success rate is GTO
- The model of best classification of accuracy is tree .
- We had a model that can predict the success of stage with X inputs

Appendix



Thank you!