IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Shikha Singh
14th November, 2022

Outline

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Conclusion
- Appendix

Executive Summary

- Summary Methodologies Deployed

  -Collecting Data via API

  -Collecting Data via Web scraping

  -Data Wrangling

  -Exploratory Data Analysis (EDA)with SQL

  -Exploratory Data Analysis (EDA) with Data Visualization

  -Interactive Visual Analytics with Folium

  -Machine Learning Prediction

- Result Summary

  -Exploratory Data Analysis Result

  -Interactive Analytics In Screenshots

  -Predictive Analytics Result

Introduction

- Project background and context
- Problems you want to find answers

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data has been collected using web scraping tools from the SpaceX website and Wikipedia
- Perform data wrangling
  - One hot encoding was used to change type
- 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

- Used get request to the SpaceX API.
- Decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
- Cleaned the data, checked for missing values and fill in missing values where necessary.
- Performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
- The task at hand 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

- Data was collected based on get rquest as instructed and worked on in lab
- I am attaching the link to the lab work below:

    watson link

    github link

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 [13]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

ode

ered": false, "ships": [], "links": {"patch": {"small": "https://images2.imgbox.com
GypSkayF_o.png"}, "reddit": {"campaign": null, "launch": null, "media": null, "recov
ull, "webcast": "https://www.youtube.com/watch?v=0a_00nJ_Y88", "youtube_id": "0a_00n
con-1-rocket-lost-launch.html", "wikipedia": "https://en.wikipedia.org/wiki/DemoSa

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

In [28]:
```
# Calculate the mean value of PayloadMass column
PayloadMass = pd.DataFrame(data_falcon9['PayloadMass'].values.tolist()).mean(1)
print(PayloadMass)
# Replace the np.nan values with its mean value
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
```

```
0    5919.165341
dtype: float64
/tmp/wsuser/ipykernel_164/2137895336.py:10: SettingWithCopyWarnin
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/p
  data_falcon9['PayloadMass'][0] = df_rows.values
```

Out[28]:

FlightNumber	Date	BoosterVersion	Payloa
	[2006-03-24, 2007-03-21, 2008-09-28, 2009-07-13, 2010-06-04, 2012-05-22, 2013-03-01, 2013-09-29,		

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

In [25]:
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_df[launch_df['BoosterVersion'] != 'Falcon 1']
data_falcon9
```

Out[25]:

FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
	[2006-03-24, 2007-03-21, 2008-09-28, 2009-07-13	[None None, None None,	False, False,					-80.577366, -120.610829,	28.5618571, 34.632093,	[False, False, False, False					[167.7431292, 167.7431292, 167.7431292, 167.7431292	[9.0477206, 9.0477206, 9.0477206, 9.0477206

IBM Developer

SKILLS NETWORK

COLLECTING DATA - WEBSCRAPING

- Data was scraped from internet as well.
- The tables were further parsed and converted to data frame using pandas
- I am attaching the link to the database as follows:

  [github link]()

BoosterVersion

Out[16]: []

Now, let's apply `getBoosterVersion` function method to get the booster version

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

In [25]:
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_df[launch_df['BoosterVersion'] != 'Falcon 1']
data_falcon9
```

Out[25]:

FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount
						[None None, None None, None None, None None, None None, None								

Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

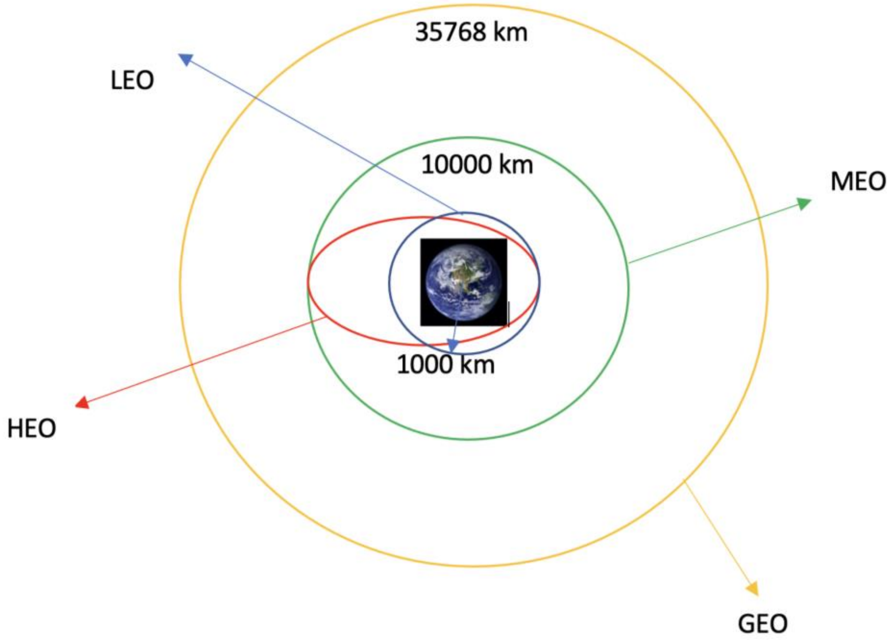
In [22]:
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
```

Data Wrangling

- Performed exploratory data analysis and determined the training labels.
- The number of launches at each site was calculated, and the number and occurrence of each orbits
- Created landing outcome label from outcome column and exported the results to csv

Data Wrangling

- Attaching notebook Link github link



1	0
2	0
3	0
4	0
5	0
6	1
7	1

In [12]:
```
df.head(5)
```

Out[12]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

We can use the following line of code to determine the success rate:

In [13]:
```
df["Class"].mean()
```

Out[13]: 0.6666666666666666

In [14]:
```
df.to_csv("dataset_part_2.csv", index=False)
```

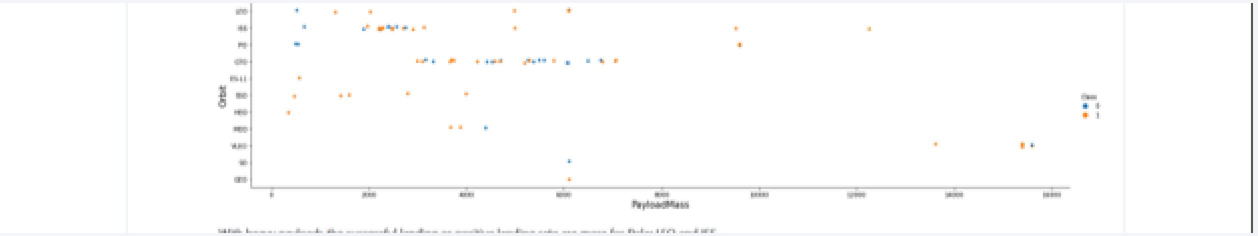
We can now export it to a CSV for the next section,but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.

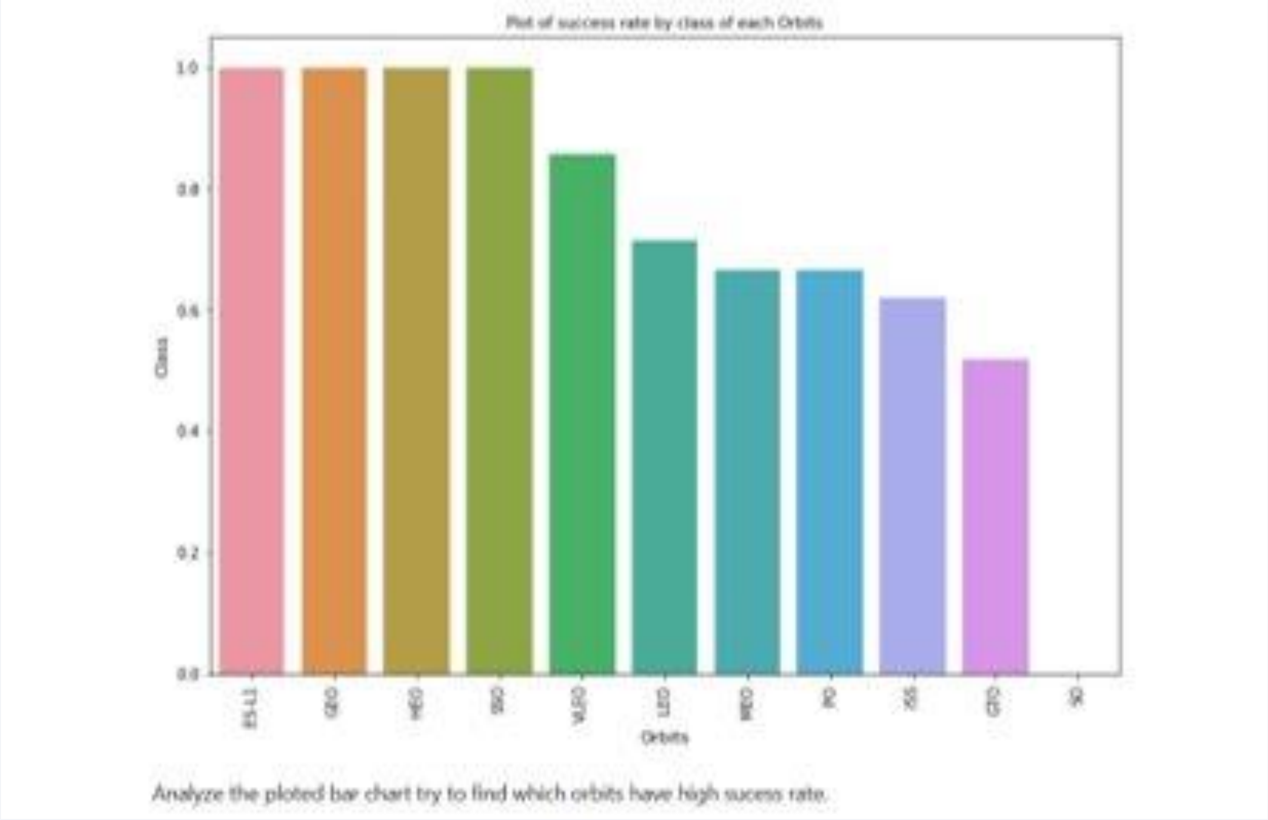
EDA with SQL

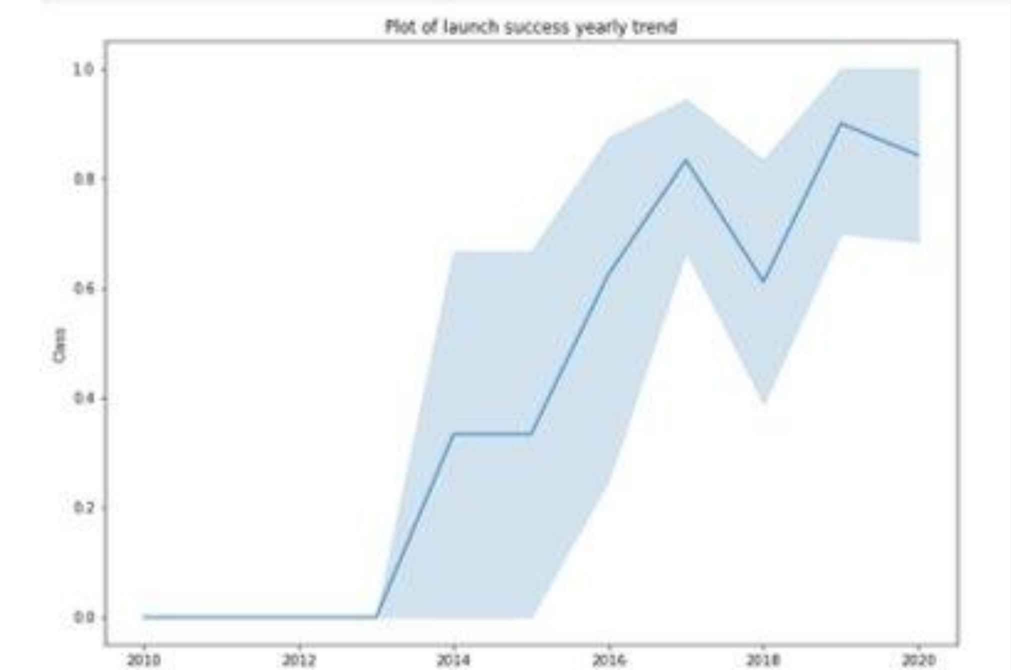
- Performed Exploratory Data Analysis using SQL. All the queries were generated post
- The notebook has been shared [Github link]
    - The names of unique launch sites in the space mission.
    - The total payload mass carried by boosters launched by NASA (CRS)
    - The average payload mass carried 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.

EDA with Data Visualization

- Performed Exploratory Data Analysis
- The notebook has been shared Github link







Build an Interactive Map with Folium

- 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.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- Assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- 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.
- The notebook has been shared Github link

Build a Dashboard with Plotly Dash

- Built an interactive dashboard with Plotly dash
- Plotted pie charts showing the total launches by a certain sites
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The notebook has been shared github link

Predictive Analysis (Classification)

- The data was loaded using numpy and pandas, then it was transformed, and the data was split into training and testing.
- Different machine learning models were built and tuned different hyperparameters using GridSearchCV.
- Accuracy was used as the metric for the model, Model was further improved using feature engineering and algorithm tuning.
- Best performing classification model was thus found.
- The notebook has been shared github link

Results

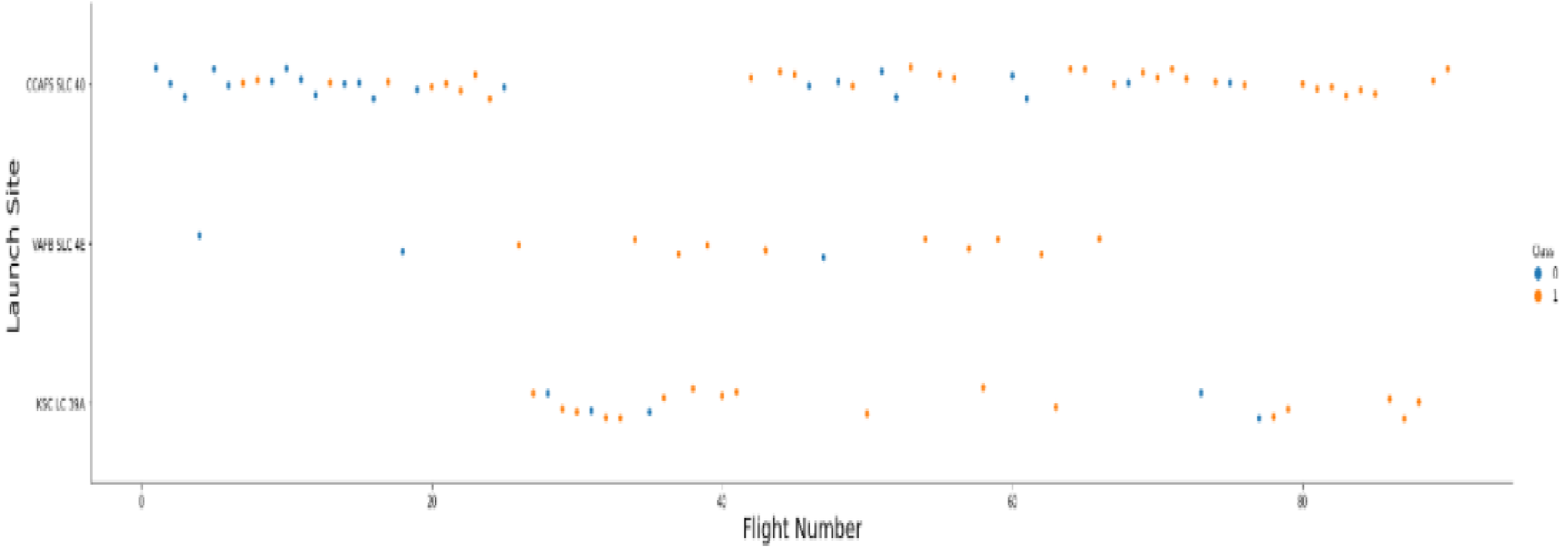
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

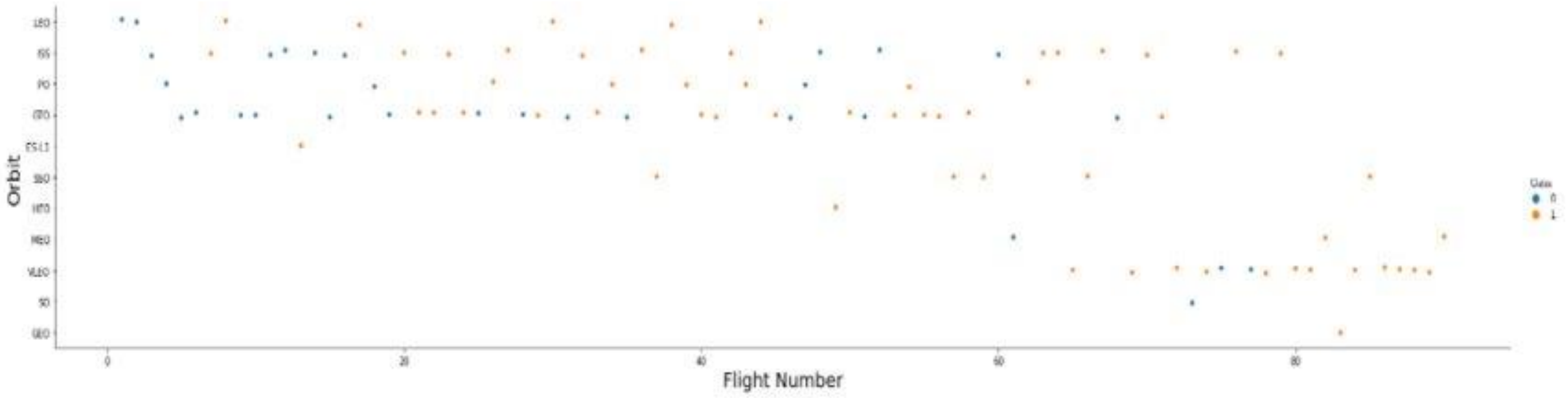
Payload vs. Launch Site

The greater the payload mass for launch site CCAFS SLC 40 the higher is the success rate



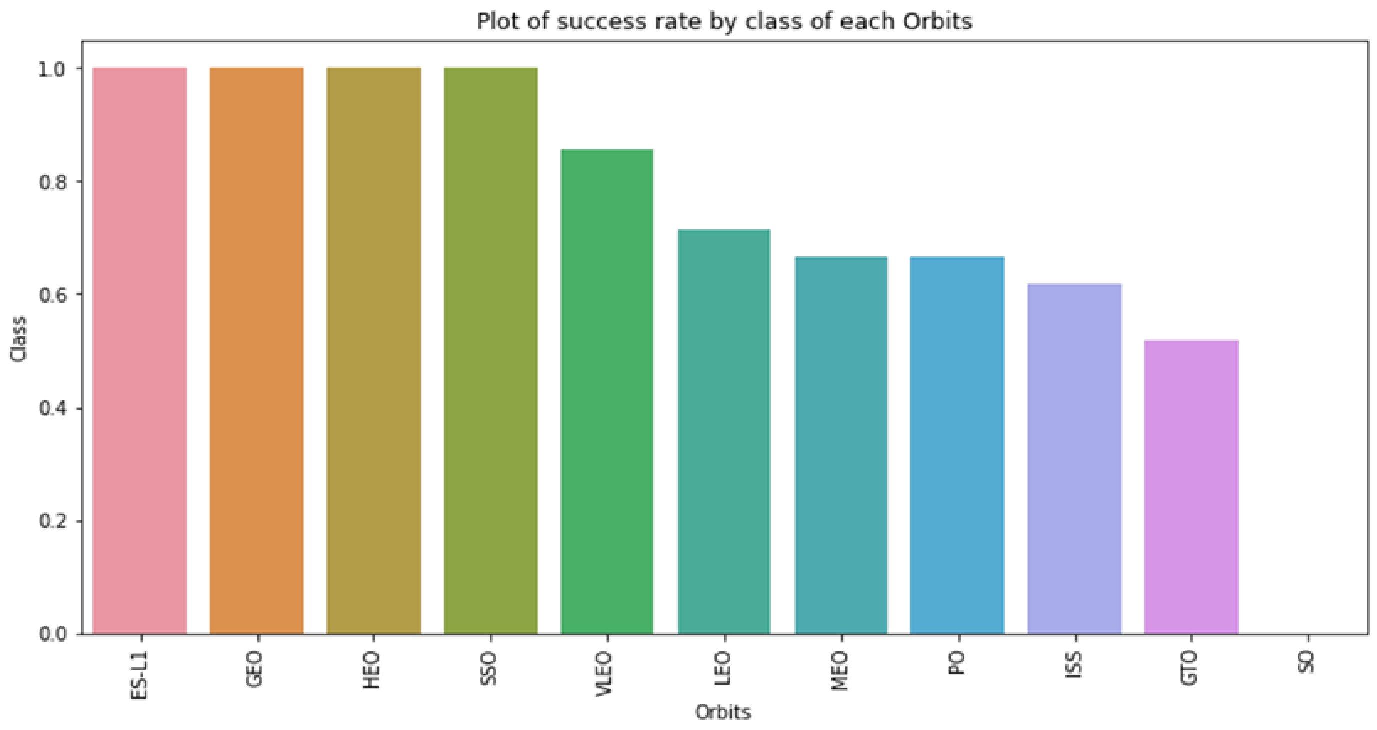
Flight Number vs. Orbit Type

- the plot shows flight number vs orbit type.
- In LEO orbit, success is related to the number of flights whereas
- GTO orbit, however, has no relationship between flight number and the orbit.



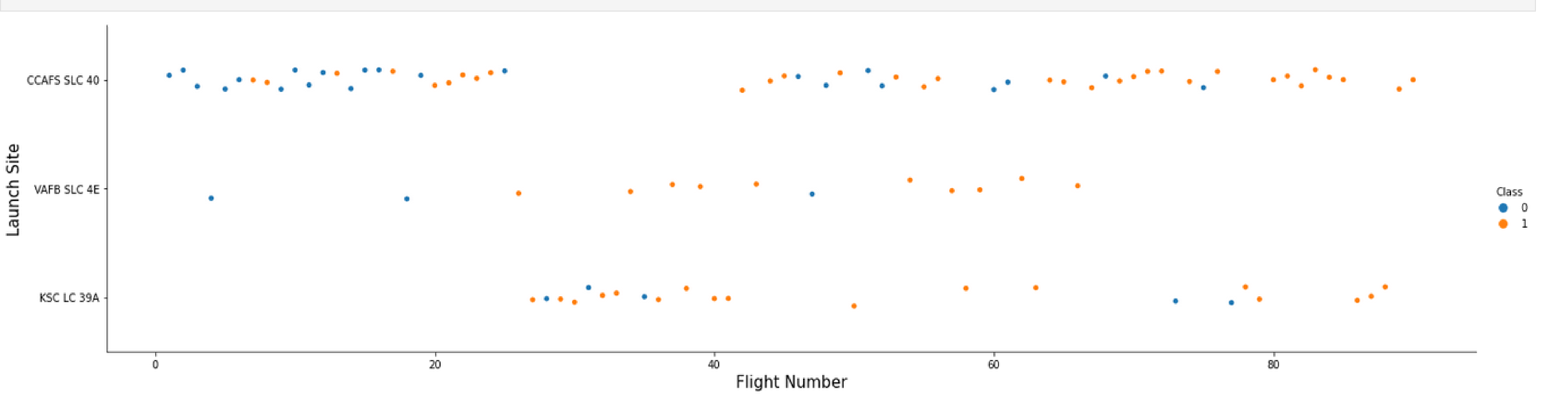
Success Rate vs. Orbit Type

- It is evident that ES-L1, GEO, HEO, SSO, VLEO had the highest success rate compared to the rest



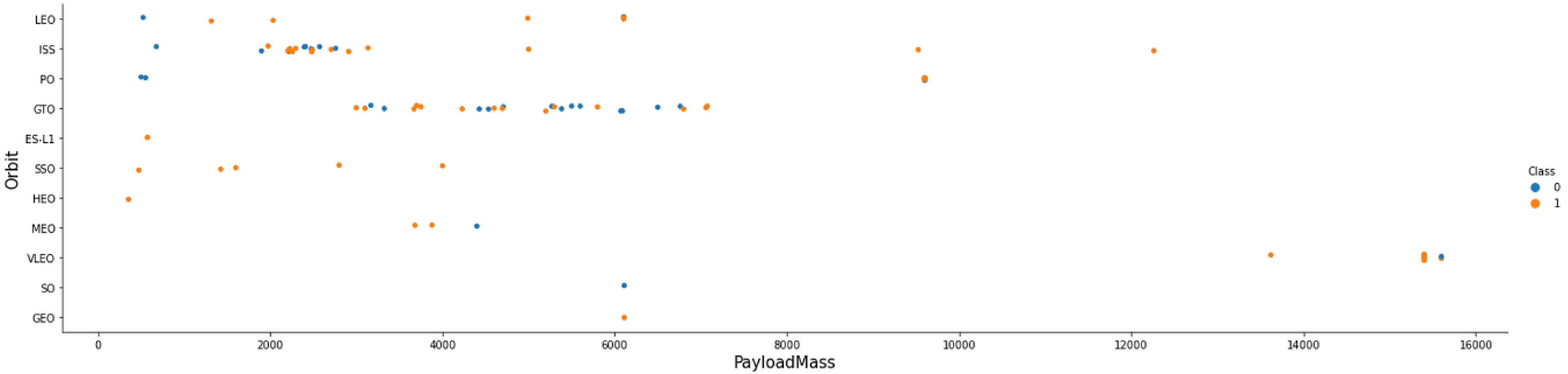
Payload vs. Launch Site

- For the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000)



Payload vs Orbit Type

- Heavy payloads have successful landing with Polar, Leo and ISS



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.

All Launch Site Names

- Unique launch sites data presented below with names

```
[35]: %sql Select Unique Launch_Site from SpaceX

 * ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/I
   ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/I
Done.
```

[35]:

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`
- This gives the results, used a simple where and select clause with % like

Task 2

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

In [38]:

```
%%sql Select * from SpaceX
where Launch_Site like 'CCA%'
limit 5
```

```
 * ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/BLUDB
   ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[38]:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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)
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-12	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170	GTO	SES	Success	No attempt

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- This is a nested query which gives the result as stated below

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

In [43]:
```
%%sql Select customer, sum(payload_mass__kg_) as "Total Payload Mass" from
        (Select customer, payload_mass__kg_ from SpaceX
        where customer LIKE 'NASA (CRS)')
        GROUP BY CUSTOMER
```

```
 * ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/BLUDB
   ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[43]:

customer	Total Payload Mass
NASA (CRS)	22007

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- The result for this was also obtained using Nested query

Task 4

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

In [44]:
```
%%sql Select booster_version    , AVG(payload_mass__kg_) as "AVERAGE Payload Mass" from
        (Select booster_version          , payload_mass__kg_ from SpaceX
        where booster_version    LIKE 'F9 v1.1')
        GROUP BY booster_version
```

```
 * ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/BLUDB
   ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[44]:

booster_version	AVERAGE Payload Mass
F9 v1.1	3676

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- The query was performed using nested query and the minimum function

In [48]:

```
%%sql Select MIN(DATE) as "FIRST SUCCESS" FROM
        (SELECT DATE FROM SPACEX
        WHERE landing__outcome LIKE 'Success%')
```

```
 * ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/BLUDB
   ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[48]:

FIRST SUCCESS
2016-06-05

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

- This was done by setting simple conditions and the result has been shown below

In [50]:
```
%%sql SELECT booster_version, payload_mass__kg_, landing__outcome from SPACEX
where 4000 < payload_mass__kg_ and payload_mass__kg_ < 6000 and landing__outcome = 'Success (drone ship)'
```

```
 * ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/BLUDB
   ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[50]:

booster_version	payload_mass__kg_	landing__outcome
F9 FT B1022	4696	Success (drone ship)
F9 FT B1031.2	5200	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- The results have been posted below; this is a simple group by mission outcomes

List the total number of successful and failure mission outcomes

In [51]:

```
%%sql SELECT mission_outcome, count(mission_outcome) as "Total" FROM SPACEX
Group by mission_outcome
```

```
 * ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/BLUDB
   ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[51]:

mission_outcome	Total
Success	44
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- It is obtained by using nested query again

In [52]:
```
%%sql SELECT Unique booster_version, payload_mass__kg_    from SPACEX
where payload_mass__kg_          = (Select max(payload_mass__kg_        ) from SPACEX)
```

```
 * ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/BLUDB
   ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[52]:

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- This is the result of failure in 2015

In [54]:

```
%%sql Select landing__outcome, booster_version,  launch_site, DATE from SPACEX
where landing__outcome = 'Failure (drone ship)' and Year(DATE) =  2015
```

```
 * ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/BLUDB
   ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[54]:

landing__outcome	booster_version	launch_site	DATE
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015-10-01

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

- Used a simple group by and total with date to obtain results

In [56]:
```
%%sql select landing__outcome, count(landing__outcome) as "Total" from Spacex
where DATE between '2010-06-04' and '2017-03-20'
group by landing__outcome
order by "Total" desc
```

```
 * ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/BLUDB
   ibm_db_sa://lpg38777:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[56]:

landing__outcome	Total
No attempt	7
Failure (drone ship)	2
Success (drone ship)	2
Success (ground pad)	2
Controlled (ocean)	1
Failure (parachute)	1

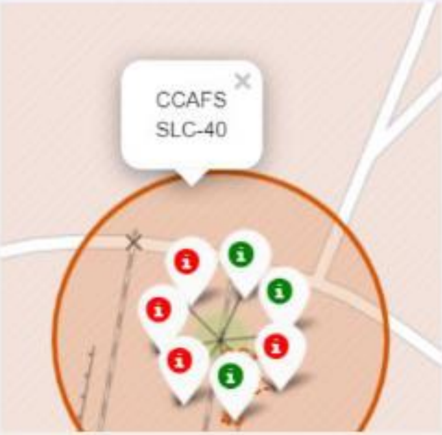
Section 3

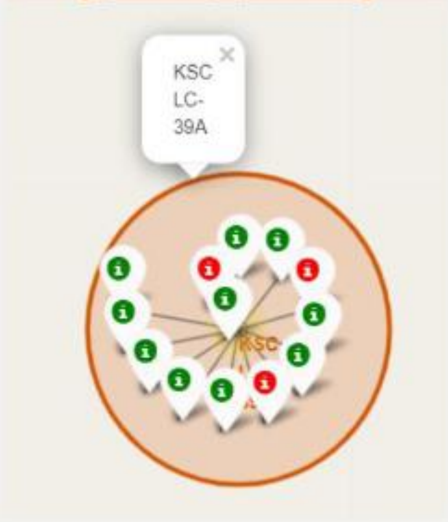
Launch Sites Proximities Analysis

All launch sites global map markers

- [github link to folium runs]()

Markers showing launch sites with color labels





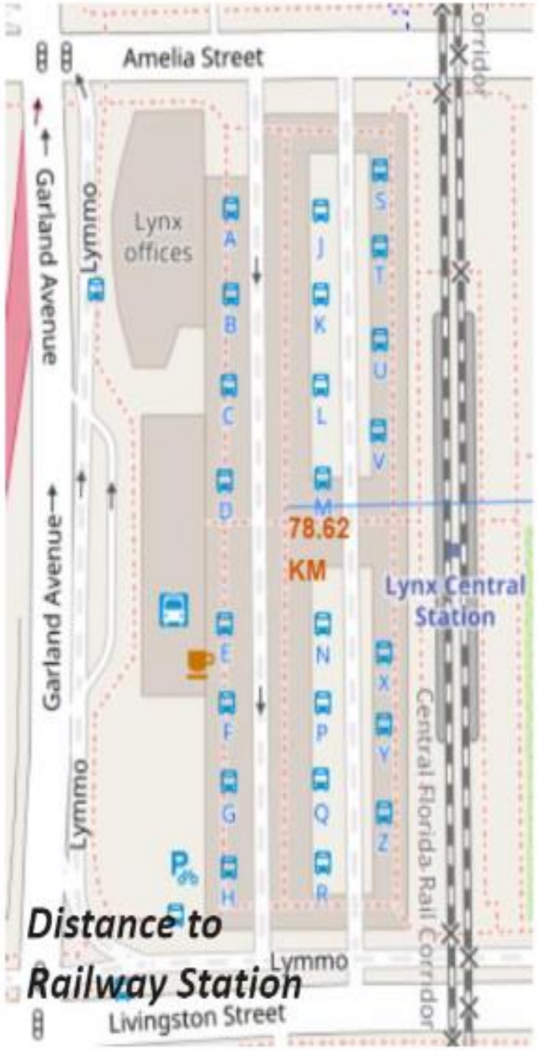
California Launch Site

37

Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

Launch Site distance to landmarks

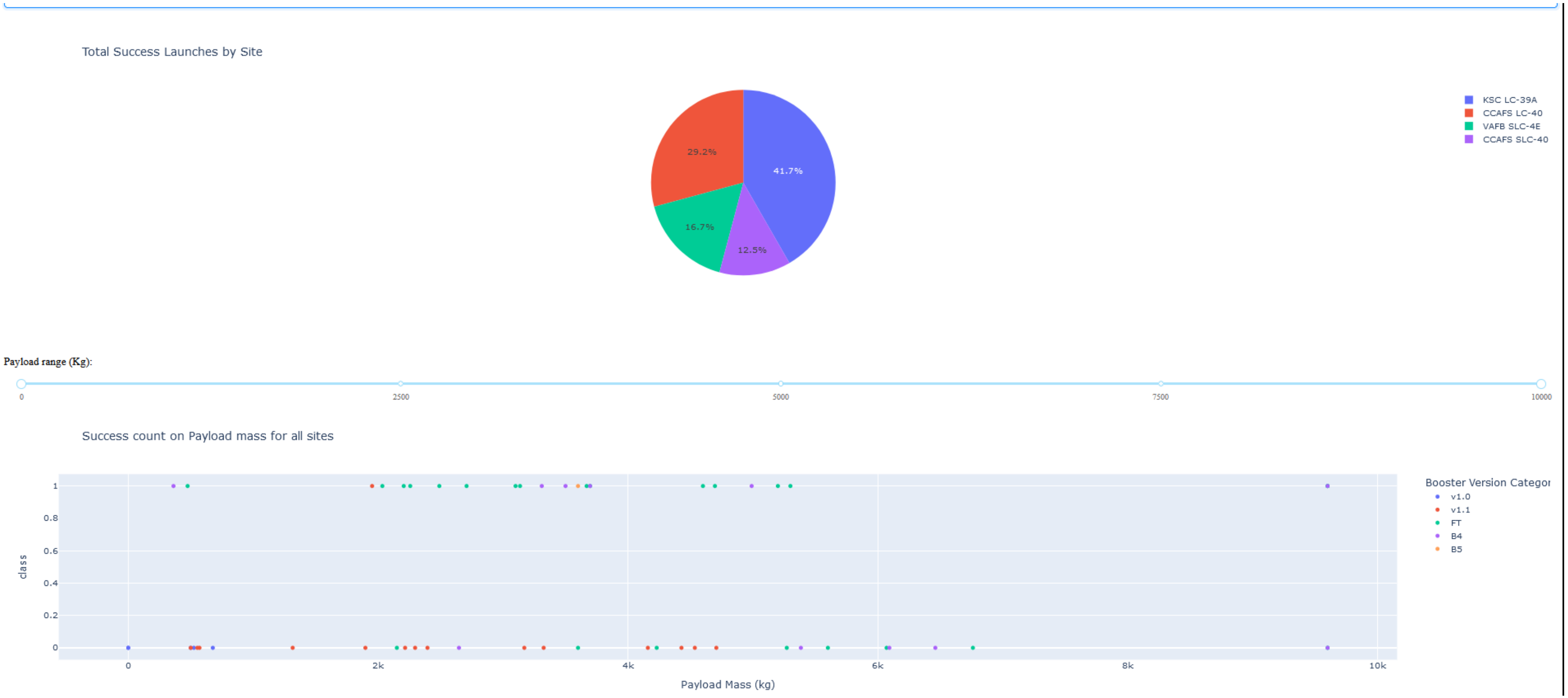


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

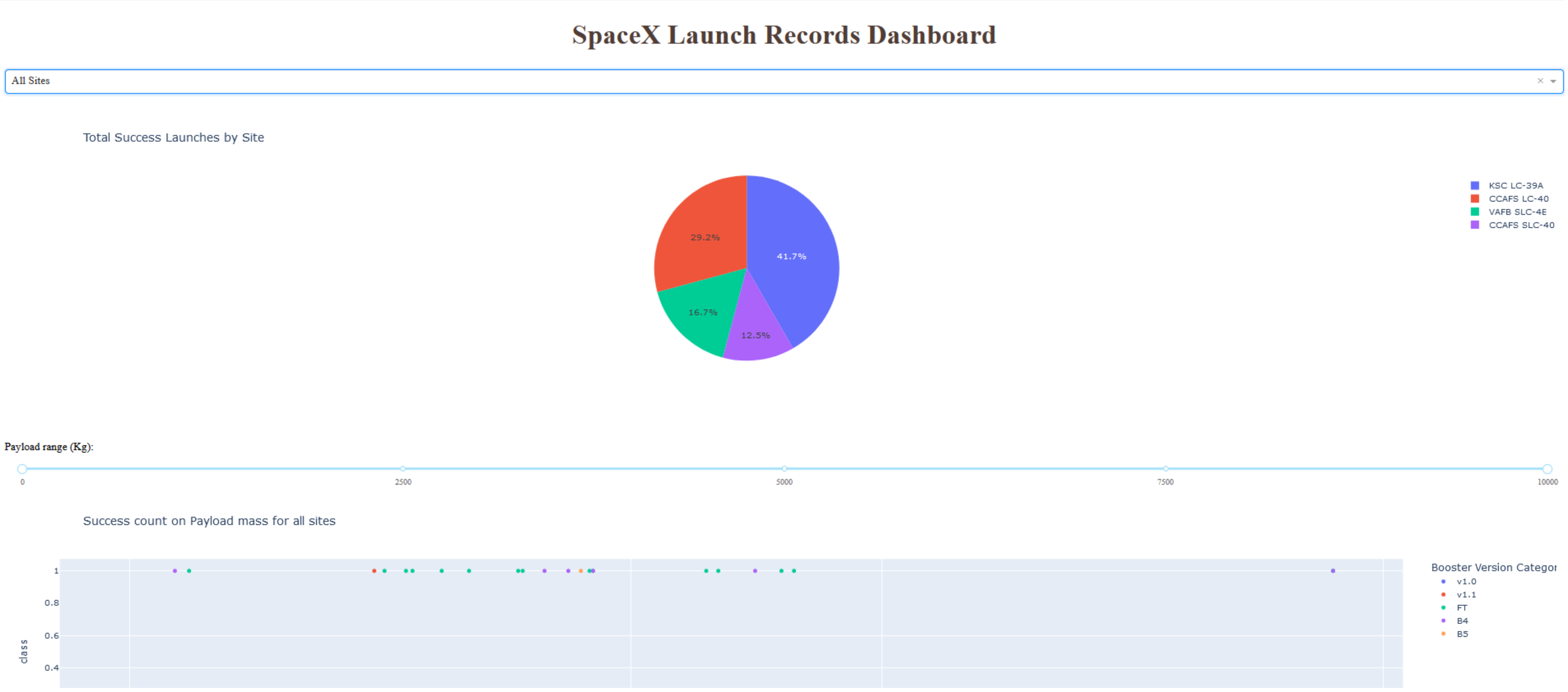
Section 4

Build a Dashboard with Plotly Dash

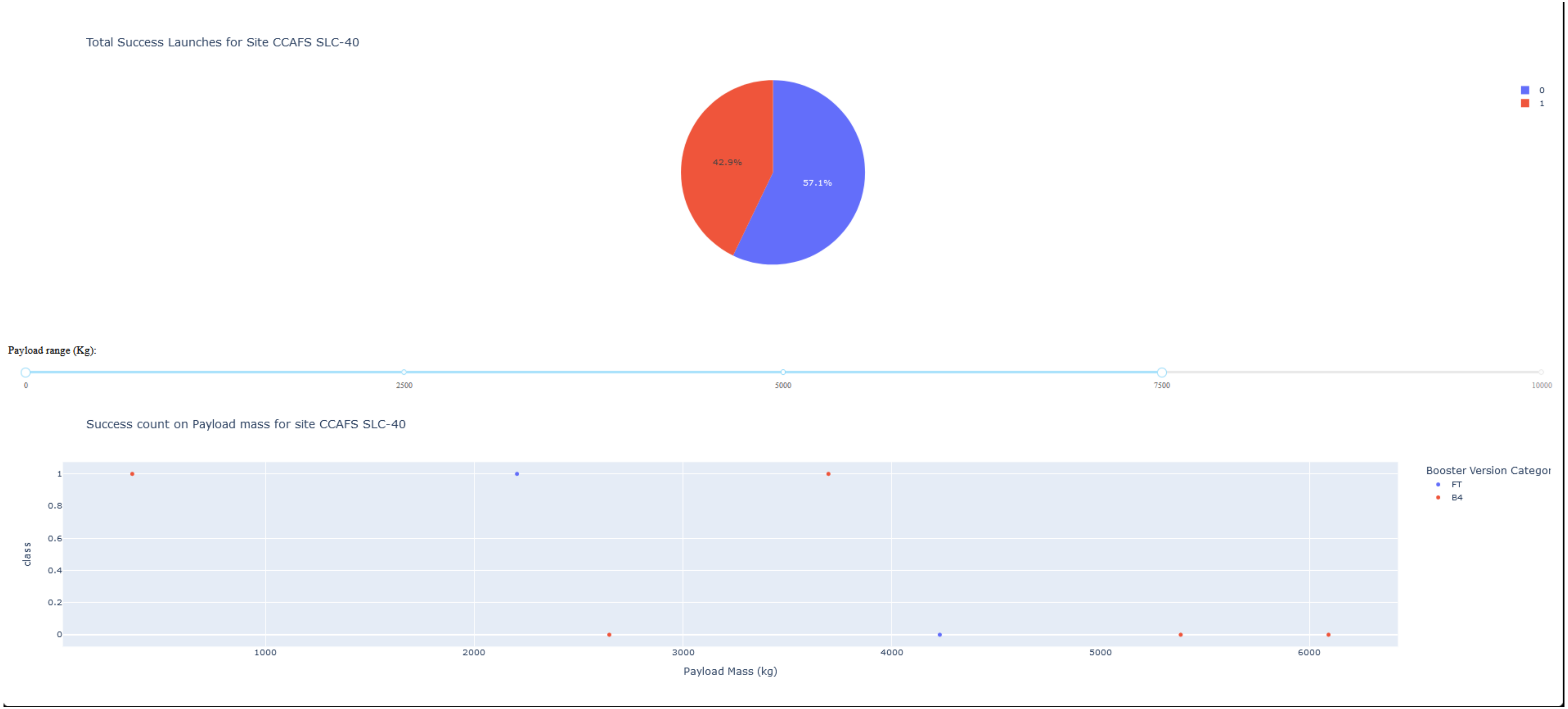
DASHBOARD TAB 1



DASHBOARD TAB 2



DASHBOARD TAB 3



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- It was found that :
- Best Performing method is DecisionTree with score 0.8732142857142856
- Best parameters are : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
- The accuracy for landing prediction seems to be quite high across
- github link

Find the method performs best:

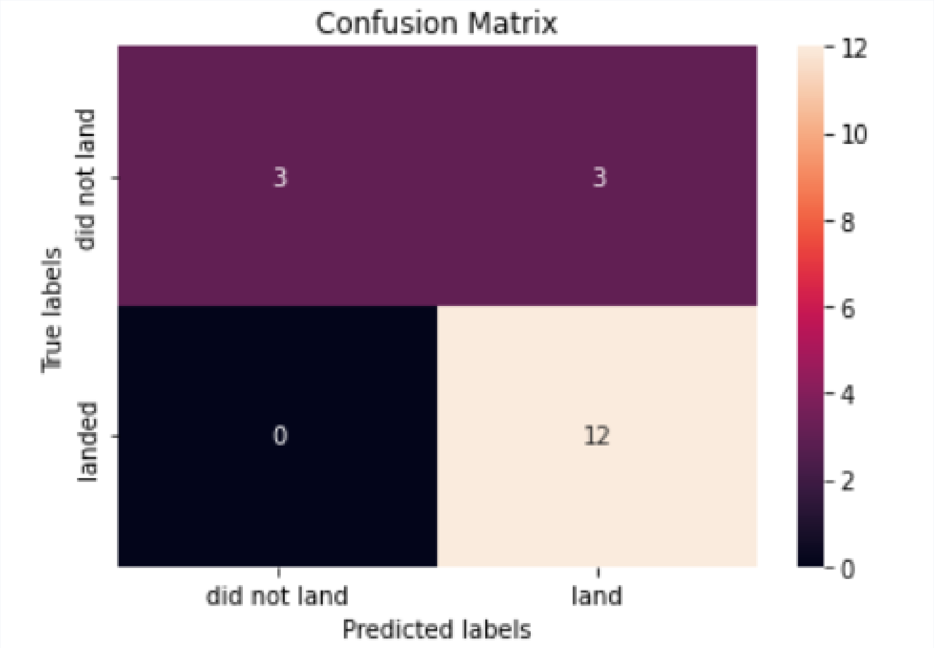
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best Performing method is', bestalgorithm,'with score', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best parameters is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best parameters is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best parameters is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best parameters is :', svm_cv.best_params_)
```

```
Best Performing method is DecisionTree with score 0.8732142857142856
Best parameters is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

- Large flight frequency ensures greater success rates.
- The success rate in launching has an upward trend from 2013 onwards
- Orbits ES-L1, GEO, HEO, SSO, VLEO has the most success rate.
- KSC LC-39A has the most successful launches of all sites.
- The decision tree classifier is the best model for this task in prediction using Machine Learning.

Thank you!