IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Mithun Bobade
27th March 2023

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 and Matplotlib
  - Interactive Visual Analytics with Folium and Ploty Dash
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result with Classification

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 factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia
  - Perform data wrangling
  - One-hot encoding was applied for 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
  - GridSerach CV is applied for parameter tuning

Data Collection

- The data was collected using following methods
  - Data collection was done using get request to the SpaceX API.
  - Then cleaned the data and checked for missing null values and fill in missing values with mean
  - 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

Data Collection – SpaceX API

- SpaceX API request is used to get the data and some data cleaning for null values

- GitHub URL of the completed SpaceX API calls notebook: https://github.com/mithun119/DataScience/blob/main/IBM-Data-Science-SpaceX/1.%20jupyter-labs-spacex-data-collection-api.ipynb

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

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

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

Check the content of the response

```
[8]: print(response.content)
```

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:

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

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

```
[10]: response.status_code
```

```
[10]: 200
```

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

```
[15]: # 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

```
[16]: # Get the head of the dataframe
data.head()
```

```
]: # Calculate the mean value of PayloadMass column
payloadmassavg = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, payloadmassavg, inplace=True)
```

Data Collection - Scraping

- We applied web scrapping to Wikipedia page of Falcon 9 launch records with BeautifulSoup

- GitHub URL for Web scraping notebook: https://github.com/mithun119/DataScience/blob/main/IBM-Data-Science-SpaceX/2.%20jupyter-labs-webscraping.ipynb

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url).text
```

Create a `BeautifulSoup` object from the HTML `response`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response, 'html.parser')
```

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

```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, plea

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all("table")
print(html_tables)
```

Data Wrangling

- We performed some Exploratory Data Analysis (EDA) to find patterns in the data and determine what would be the label for training supervised models.
- We found null values count for each column, Unique Launch sites and landing outcome column and success rate.
- GitHub URL : https://github.com/mithun119/DataScience/blob/main/IBM-Data-Science-SpaceX/3.%20IBM-DS0321EN-SkillsNetwork_labs_module_1_L3_labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Comp
4E), Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Lau

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the nu

```
# Apply value_counts() on column LaunchSite
df.LaunchSite.value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

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

```
Class
1    60
0    30
dtype: int64
```

```
df.head(5)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitu
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	28.561
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	28.561
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	28.561
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	34.632
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	28.561

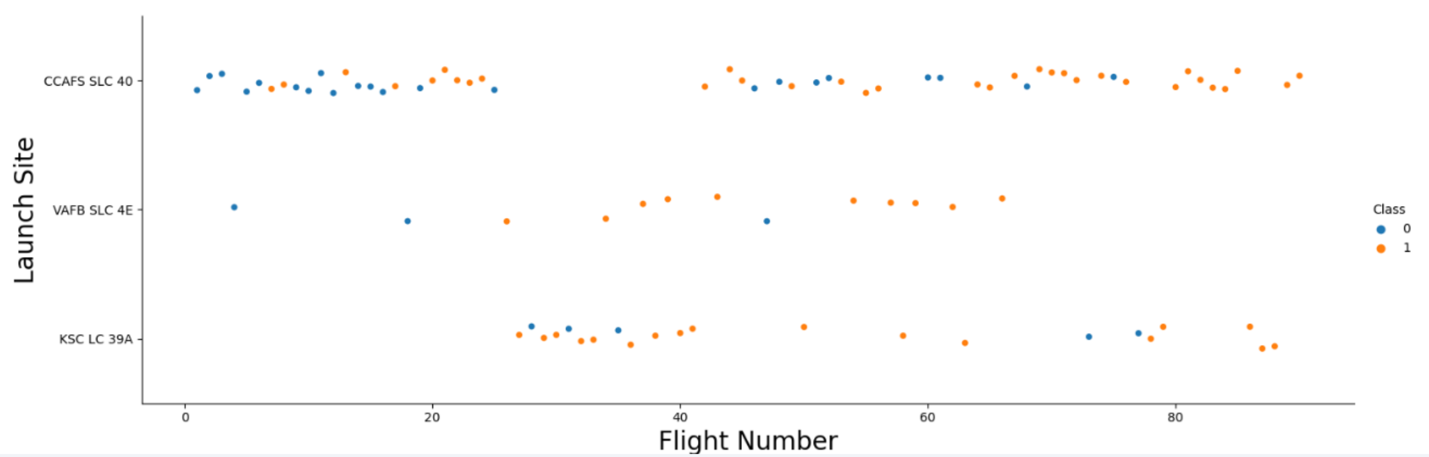
We can use the following line of code to determine the success rate:

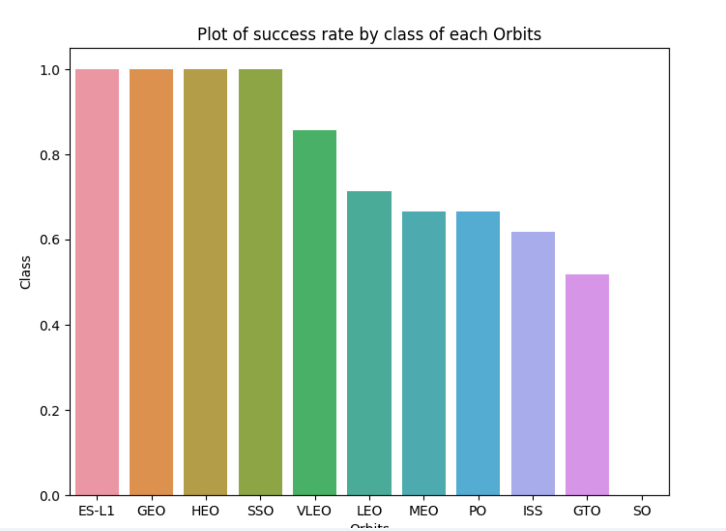
```
df["Class"].mean()
```

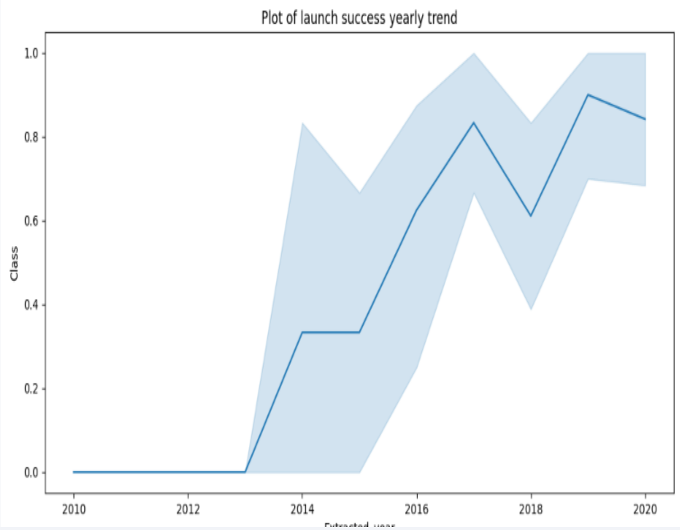
```
0.6666666666666666
```

EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number, launch Site, payload, Orbit type and yearly trend
- GitHub URL https://github.com/mithun119/DataScience/blob/main/IBM-Data-Science-SpaceX/5.%20IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb







EDA with SQL

- We applied EDA with SQL to get insight from the data. We ran following queries to find out relations and insights:

  Names of the unique launch sites in the space mission.

  Launch sites begin with the string 'CCA'

  Total payload mass carried by boosters launched by NASA (CRS)

  first successful landing outcome in ground pad was achieved.

  Names of the boosters which have success in drone ship and have payload > 6000

  Insight of total number of successful and failure mission outcomes

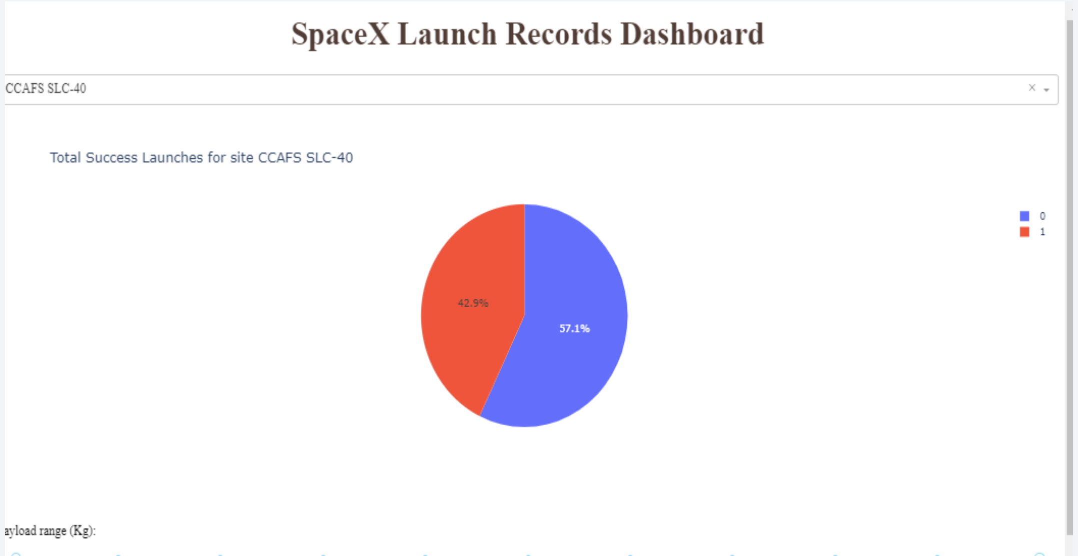
Add the GitHub URL : SQL EDA Notebook *(all sql queries)*

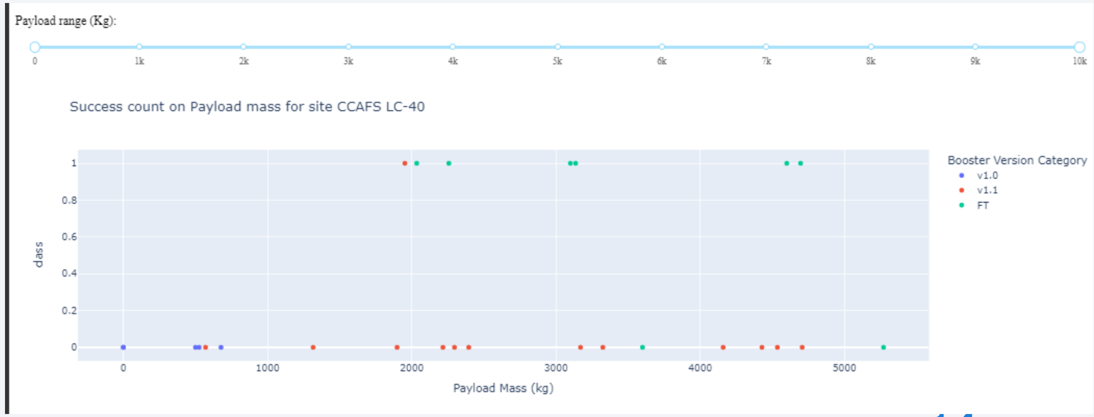
Build an Interactive Map with Folium

- We marked all launch sites and added map objects such as markers, circles 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.
- We also mapped the distance between launch site and other proximities.
- GitHub URL : https://github.com/mithun119/DataScience/blob/main/IBM-Data-Science-SpaceX/6.%20IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- We built interactive pie charts to show success rate of each launch site and scatter chart with outcome and payload.
- We added dropdown to choose launch site and display respective charts
- GitHub URL: https://github.com/mithun119/DataScience/blob/main/IBM-Data-Science-SpaceX/7.%20SpaceX_Dashboard.py





Predictive Analysis (Classification)

- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We got beast score and score of each model for comparison
- GitHub URL: https://github.com/mithun119/DataScience/blob/main/IBM-Data-Science-SpaceX/8.%20IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Find the method performs best:

[78]:
```python
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)

print('Score for Logistic_Reg :', logreg_cv.score(X_test, Y_test))
print('Score for SVM :', svm_cv.score(X_test, Y_test))
print('Score for Decision Tree :', tree_cv.score(X_test, Y_test))
print('Score for KNN :', knn_cv.score(X_test, Y_test))

print('Score for all algoritham is simillar')
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
Score for Logistic_Reg : 0.8333333333333334
Score for SVM : 0.8333333333333334
Score for Decision Tree : 0.8333333333333334
Score for KNN : 0.8333333333333334
Score for all algoritham is simillar
```

Results

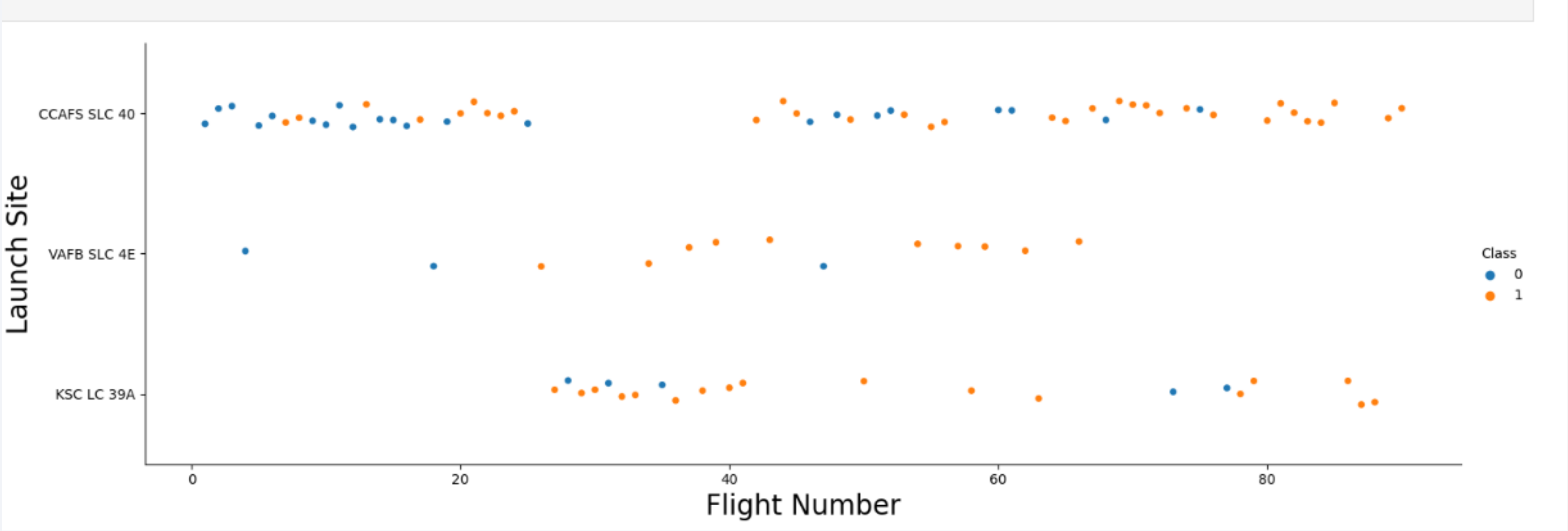
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

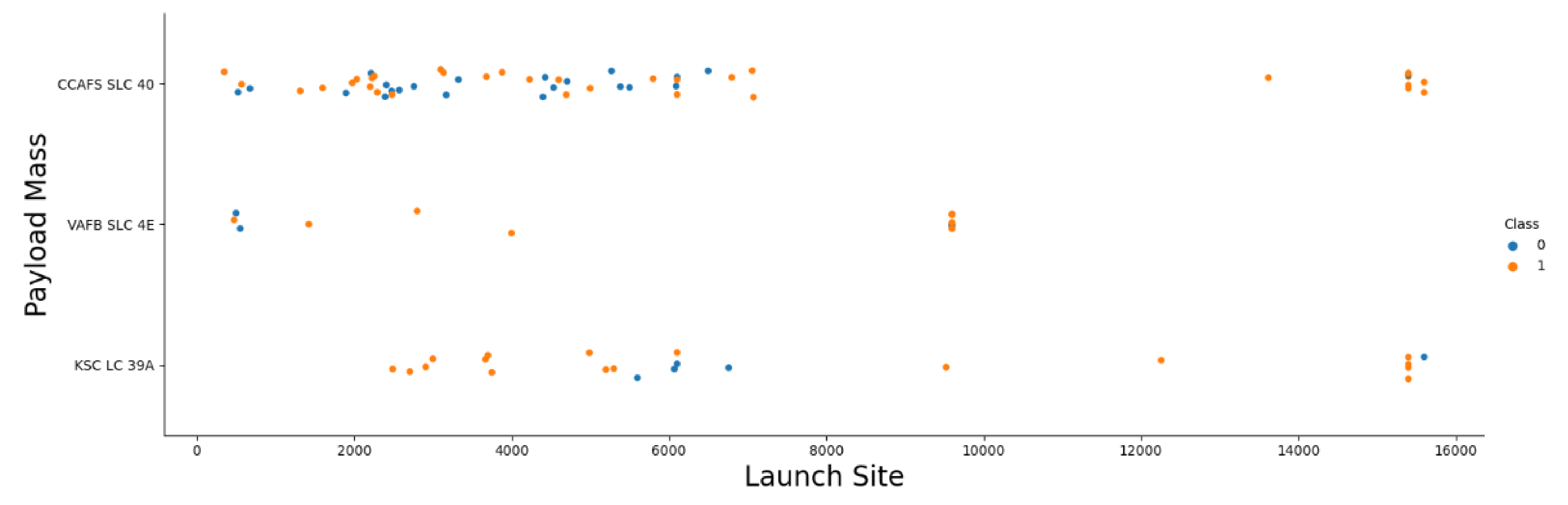
Flight Number vs. Launch Site

- Following scatter plot of Flight Number vs. Launch Site indicates that as flight number increases success rate also increases. For VAFB SLC 4E , after flight number 50, success rate is 100% and same for other two sites its after flight number 80.



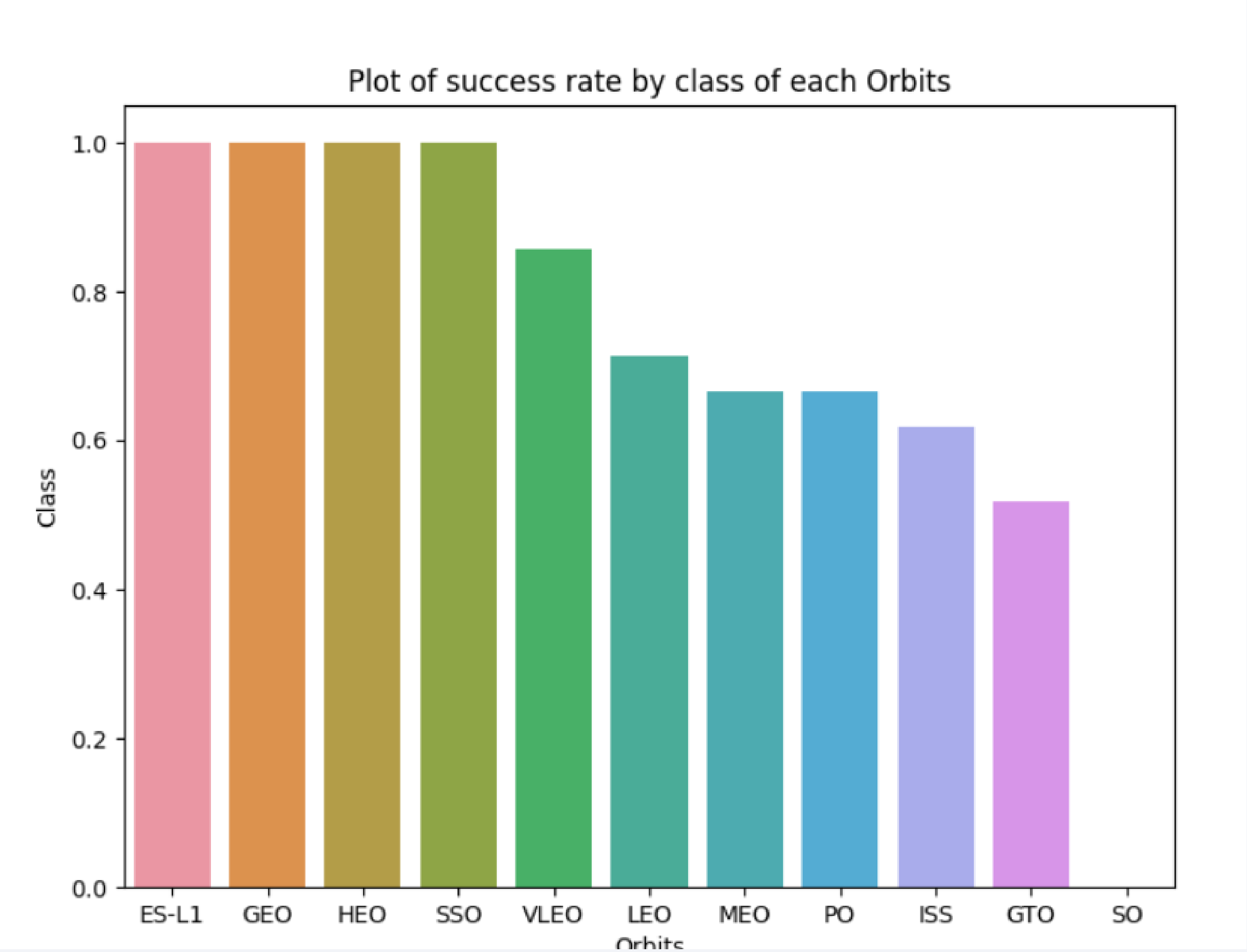
Payload vs. Launch Site

- Following scatter plot of Payload vs. Launch Site shows that VAFB-SLC launchsite there are no rockets launched for heavy payload mass(greater than 10000).



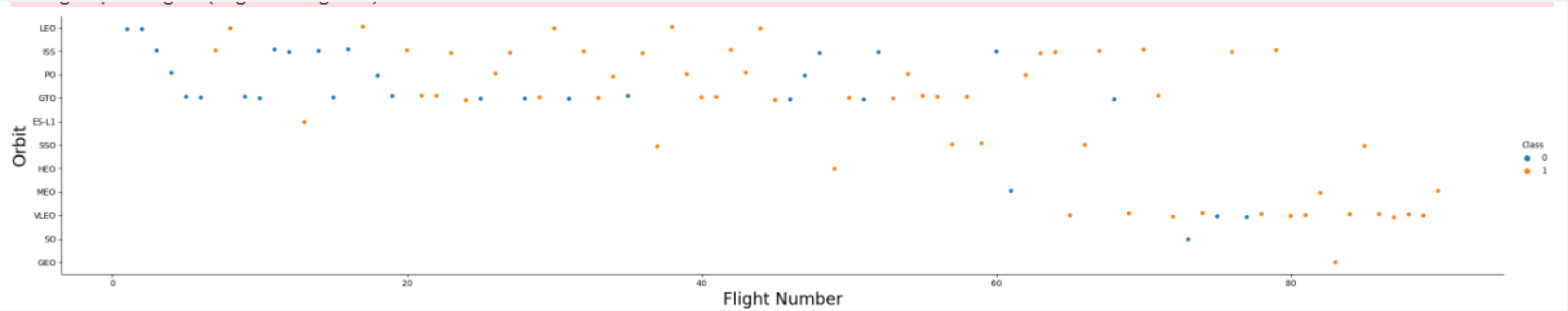
Success Rate vs. Orbit Type

- Bar chart for the success rate of each orbit type shows that ES-L1, GEO, HEO, SSO, and VLEO are the Orbits that have high success rate. The SO has the least success rate amongst the orbits.



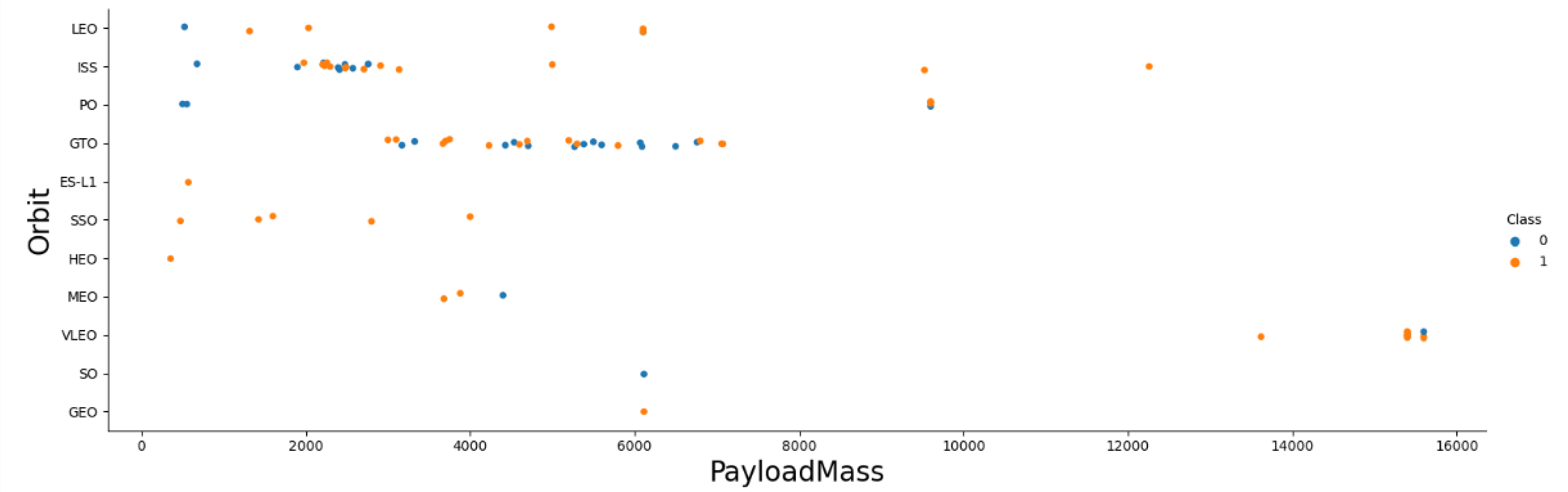
Flight Number vs. Orbit Type

- Scatter point of Flight number vs. Orbit type shows 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.



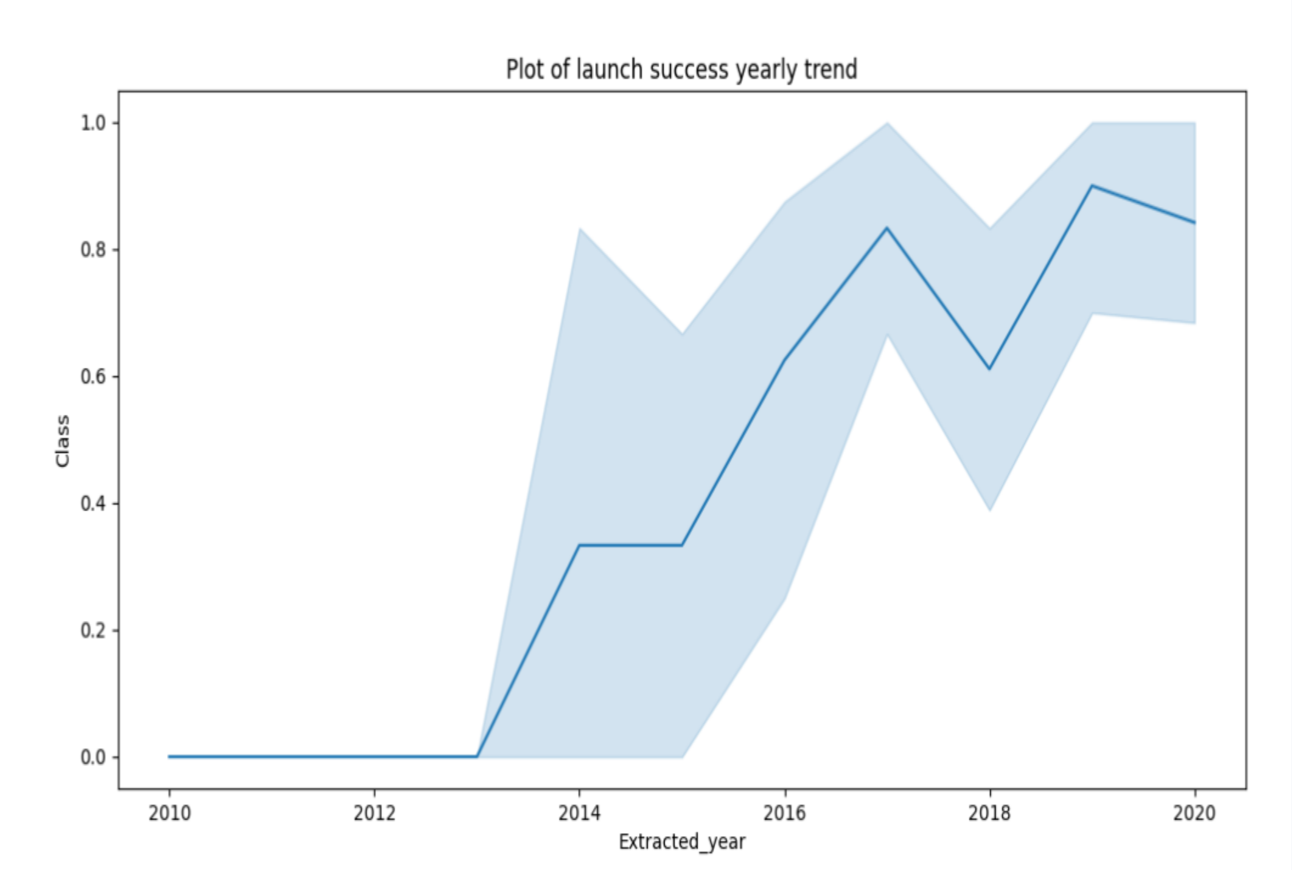
Payload vs. Orbit Type

- Scatter point of payload vs. orbit type shows that With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.



Launch Success Yearly Trend

- Line chart shows that the success rate since 2013 kept on increasing till 2020



All Launch Site Names

- Unique launch sites with **DISTINCT** keyword

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

In [27]:
```
%sql select Distinct(LAUNCH_SITE) from SPACEXTBL;
```

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

Out[27]:

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

Launch Site Names Begin with 'CCA'

- Launch sites begin with `CCA` with **Like** Query.

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

In [22]:
```
%sql select * from SPACEXTBL where upper(Launch_Site) like 'CCA%' limit 5;
```

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

Out[22]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Total payload carried by boosters from NASA with Where filter and Sum function.

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

In [11]:
```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer = 'NASA (CRS)';
```

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

Out[11]:

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- Average payload mass carried by booster version F9 v1.1 with Avg function

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

In [13]:
```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version = 'F9 v1.1';
```

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

Out[13]:

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- Date of the first successful landing outcome on ground pad with use of MIN function on date and Where filter on outcome.

Hint:Use min function

```
In [28]: %sql SELECT MIN(DATE) FROM SPACEXTBL WHERE "Landing _Outcome" = 'Success (ground pad)';

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

Out[28]:

MIN(DATE)
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

- Names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 from Between filter

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

```
%sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 AND "Landing _Outcome" = 'Success (drone ship)';
```

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

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

Total Number of Successful and Failure Mission Outcomes

- Total number of successful and failure mission outcomes with use of COUNT and GROUP BY aggregate function.

List the total number of successful and failure mission outcomes

In [32]:
```
%sql SELECT MISSION_OUTCOME, COUNT(*) FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

Out[32]:

Mission_Outcome	COUNT(*)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- Name of the booster which have carried the maximum payload mass with use of subquery

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

In [33]:
```
%sql SELECT BOOSTER_VERSION,PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

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

Out[33]:

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

2015 Launch Records

- Failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015 with use of LIKE on date and Where filter on Lading outcome.

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

In [35]:

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE "Landing _Outcome"='Failure (drone ship)' AND DATE LIKE '%2015%';
```

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

Out[35]:

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 with use of COUNT and GROUP BY

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

In [42]:
```
%sql SELECT "Landing _Outcome", COUNT("Landing _Outcome") FROM SPACEXTBL WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017' GROUP BY "Landing _Outcome"
```

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

Out[42]:

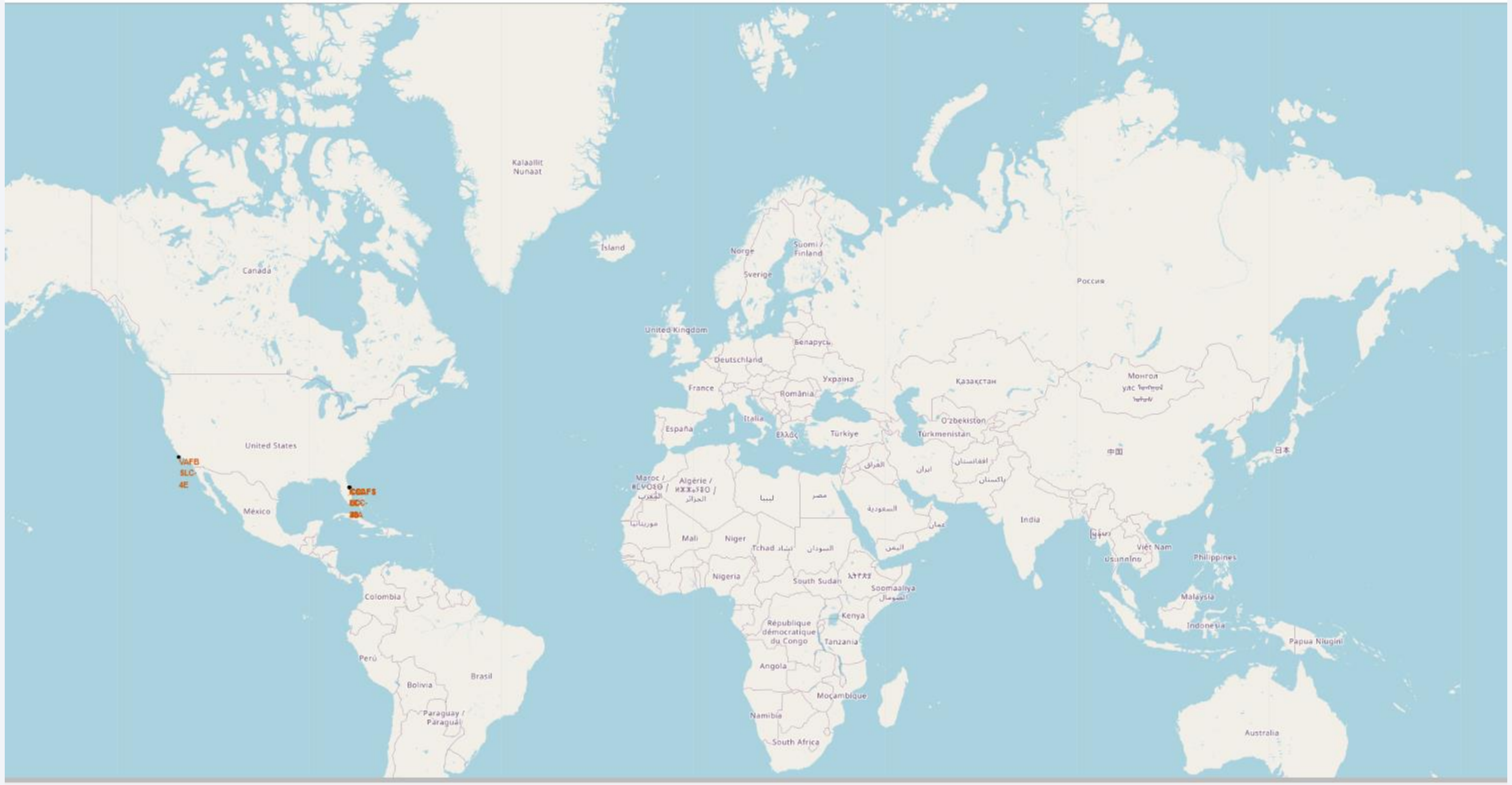
Landing _Outcome	COUNT("Landing _Outcome")
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

Section 3

Launch Sites Proximities Analysis

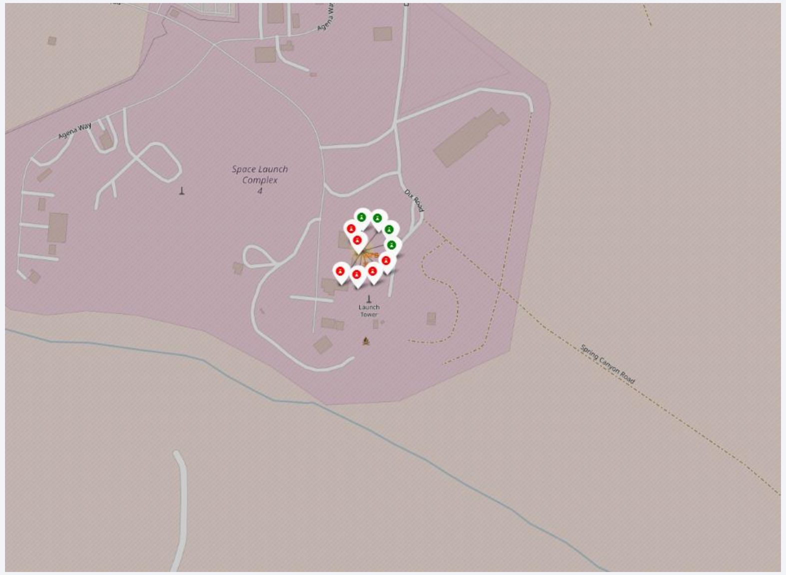
Global Map of Launch Sites

- All three Launch sites are near USA coastline.



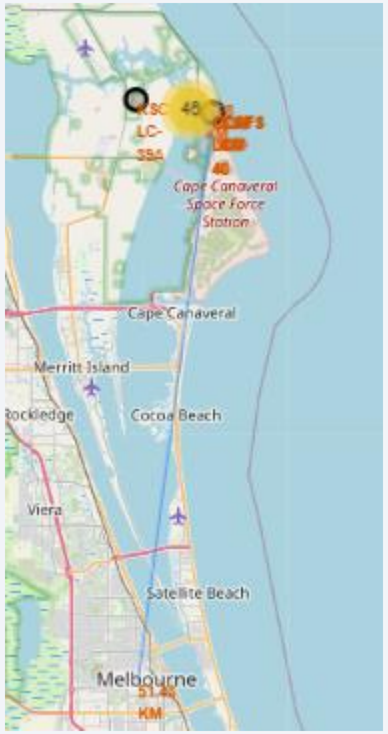
Color labeled Markers on launch site

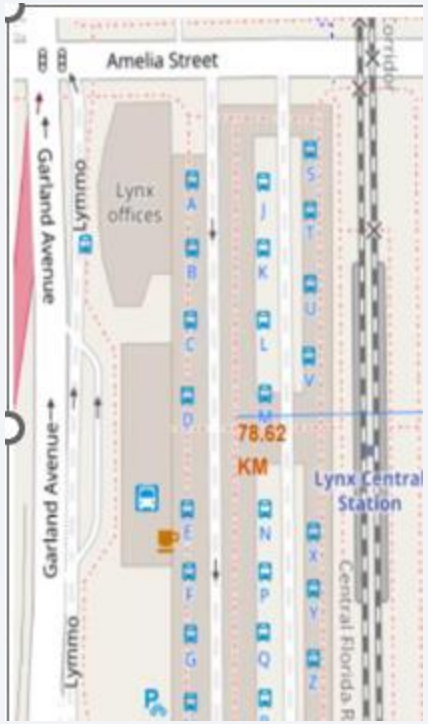
- Green marks shows successful launches and red mark shows failures.



Launch Site distance to landmarks

- From distance we come to know that launch sites are near to coast line and far from cities.



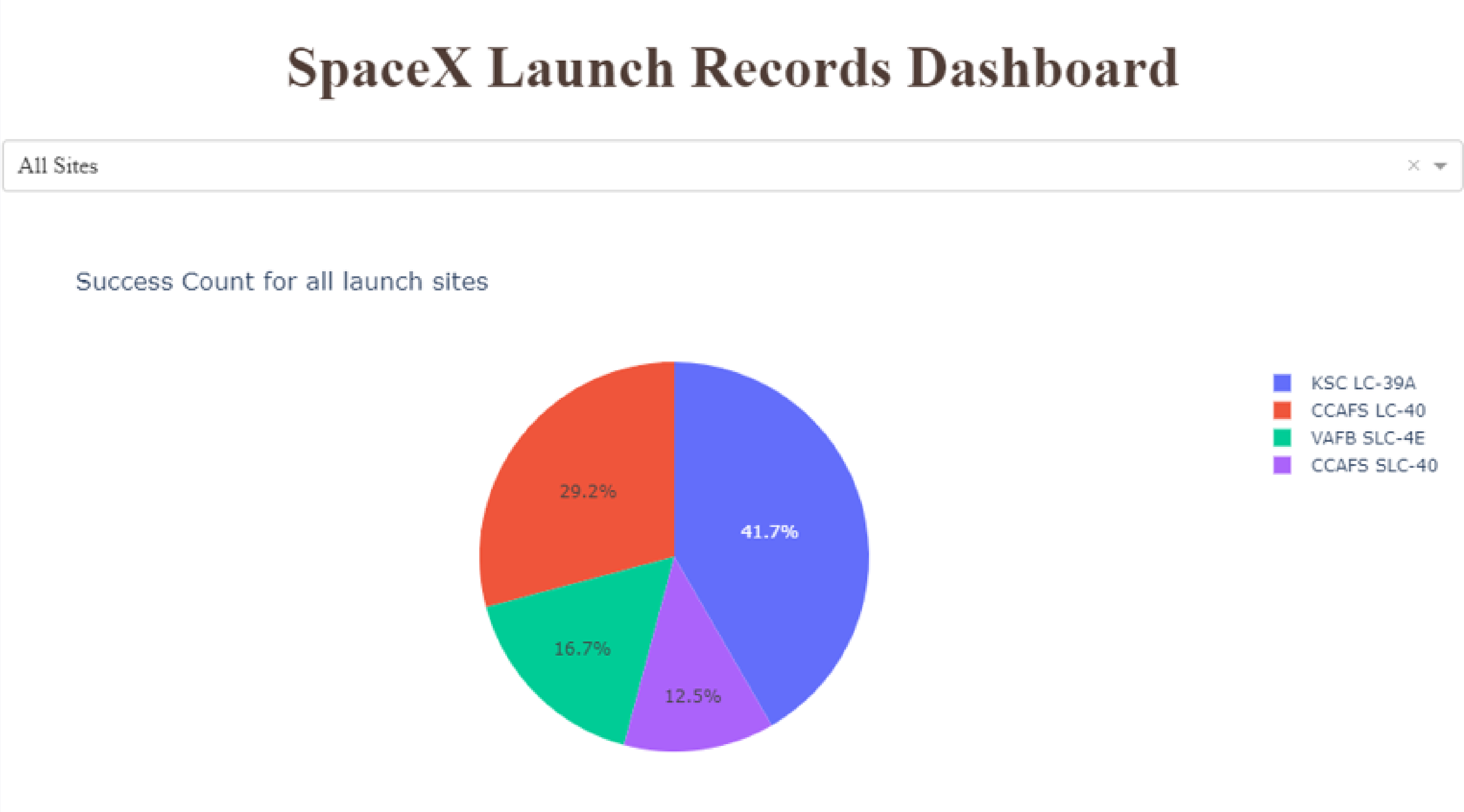


Section 4

Build a Dashboard with Plotly Dash

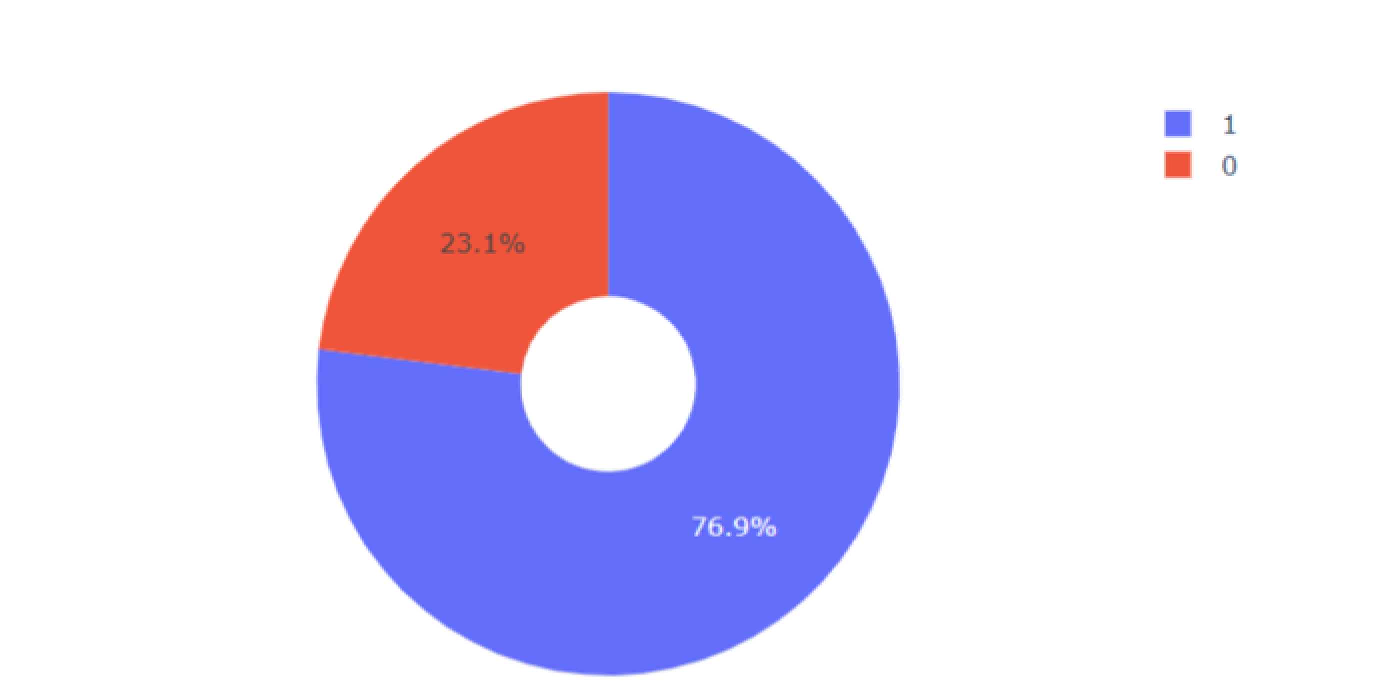
Pie Chart of Success Percentage

- We can see that KSC LC 39A had most successful launches than other sites



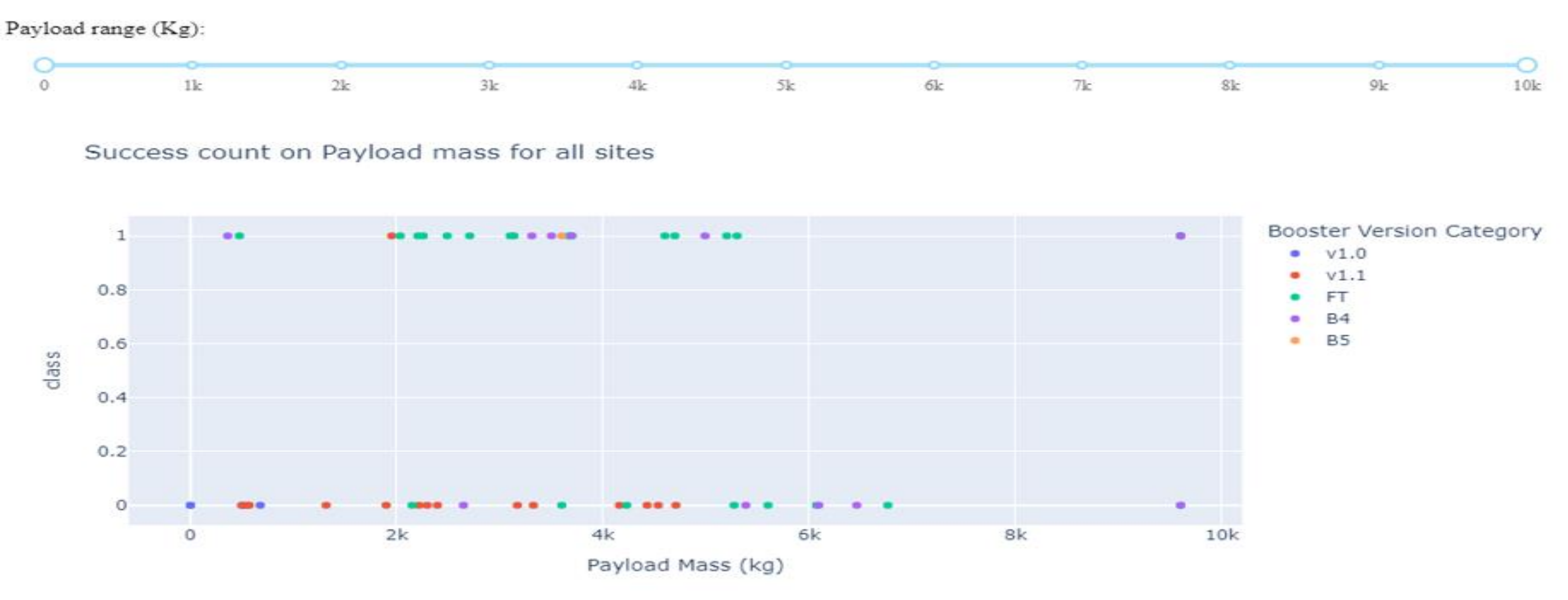
Pie Chart of Success Ratio

- KSC LC-39A has 76.9% success rate and 23.1% failure rate



Payload vs Launch outcome for all sites

- For Launch site CCAFS LC-40 the booster version FT has the largest success rate from a payload mass of >2000kg



Section 5

Predictive Analysis (Classification)

Classification Accuracy

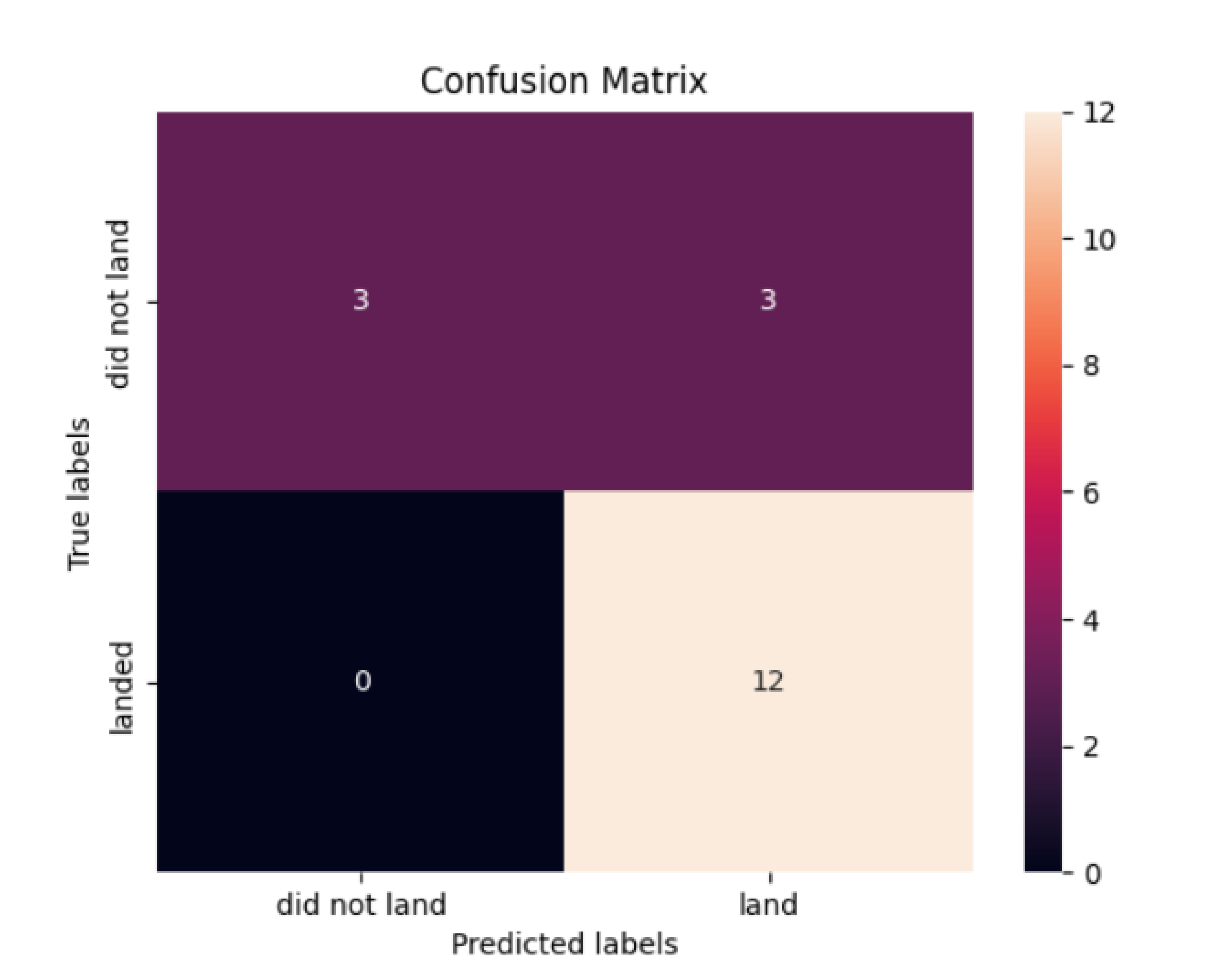
- Decision tree has highest classification accuracy with 'Best score' parameter . All models have similar score with 'Score' parameter.

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

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)

print('Score for Logistic_Reg :', logreg_cv.score(X_test, Y_test))
print('Score for SVM :', svm_cv.score(X_test, Y_test))
print('Score for Decision Tree :', tree_cv.score(X_test, Y_test))
print('Score for KNN :', knn_cv.score(X_test, Y_test))

print('Score for all algoritham is simillar')
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
Score for Logistic_Reg : 0.8333333333333334
Score for SVM : 0.8333333333333334
Score for Decision Tree : 0.8333333333333334
Score for KNN : 0.8333333333333334
Score for all algoritham is simillar
```

Confusion Matrix

- Confusion matrix shows that major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

- Different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.
- Flight number increases in each of the 3 launch sites, so does the success rate.
- Orbits ES-L1, GEO, HEO & SSO have the highest success rates at 100%, with SO orbit having the lowest success rate at ~50%. Orbit SO has 0% success rate.
- Launch success rate started to increase in 2013 till 2020.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!