Capstone Project - SpaceX

Shikha Singh

14th November, 2022

OUTLINE

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

EXECUTIVE 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

  Space X's Falcon 9 rocket launches have a cost of 62 million dollars; the other providers cost over 165 million dollars each, the savings usually happen because SpaceX can reuse the first stage. Therefore, we have to determine the price of each launch,we have to determine if the first stage will land, we can determine the cost of a launch. We will also determine if SpaceX will reuse the first stage.. 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

  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 SpaceX for a rocket launch. In this lab, you will collect and make sure the data is in the correct format from an API. The following is an example of a successful and launch.

- What operating conditions needs to be in place to ensure a successful landing program.

METHODOLOGY

- Collect Data using SPACE X API and web scraping from wikipedia.
- Deploy Data Wrangling
- Applied One-hot encoding to categorical features for converting to dummy variables (continuous)
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
- Build, tune, evaluate classification models

DATA COLLECTION

- Data was collected baed 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									[False, False, False, False					[167.7431292, 167.7431292, 167.7431292, 167.7431292	[9.0477206, 9.0477206, 9.0477206, 9.0477206

[None None, None None,

False, False,

-80.577366, -120.610829,

28.5618571, 34.632093,

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,
```

EXPLORATORY DATA ANALYSIS – SQL

- Performed Exploratory Data Analysis using SQL. All the queries were generated post
- The notebook has been shared Github link

Task 3

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

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

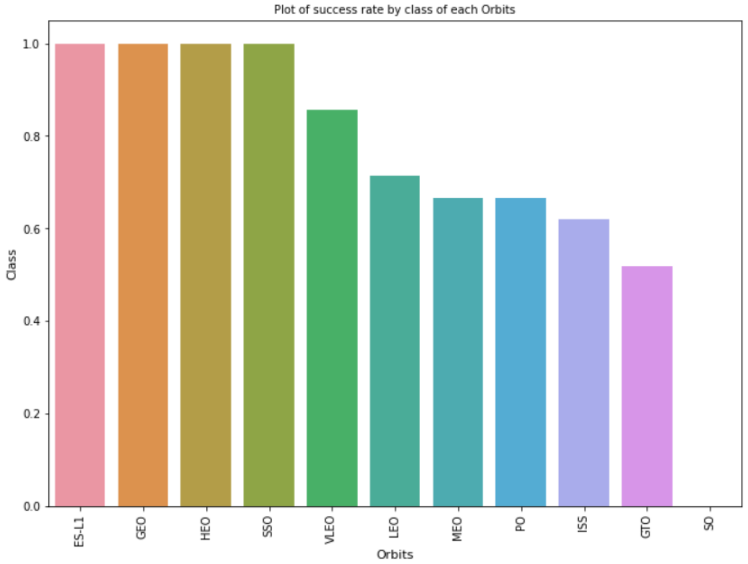
Task 5

List the date when the first successful landing outcome in ground pad was acheived.

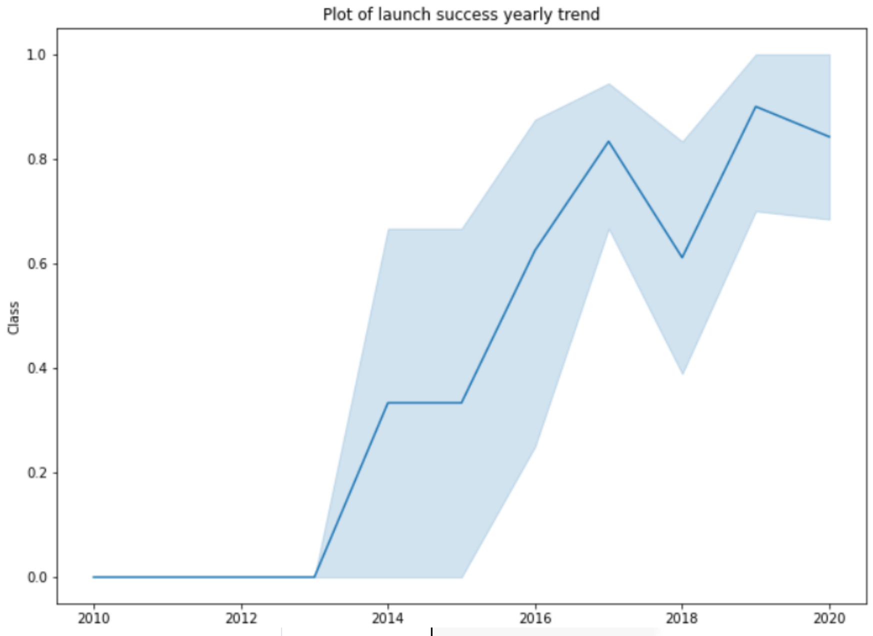
Hint:Use min function

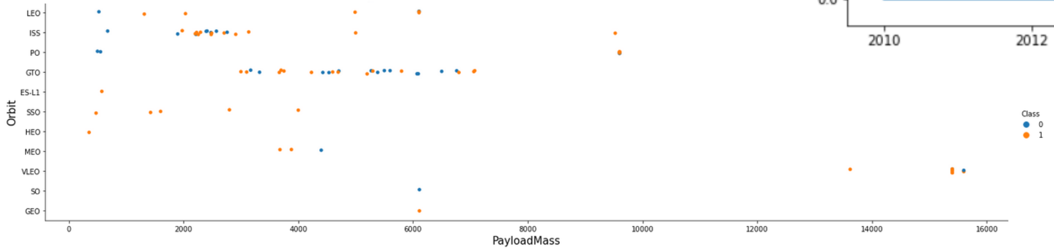
EXPLORATORY DATA ANALYSIS – Visulaization

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



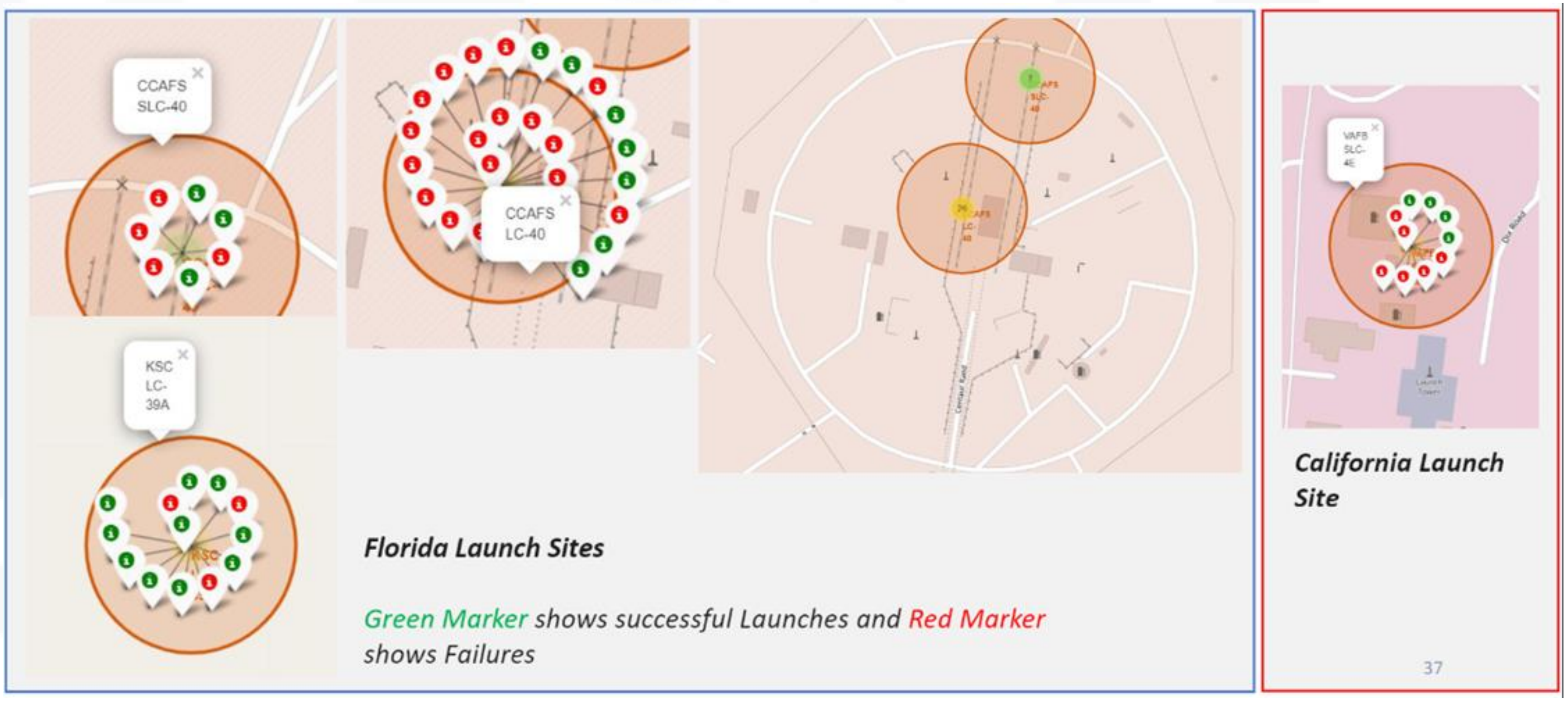
Analyze the ploted bar chart try to find which orbits have high sucess rate.





INTERACTIVE MAP - FOLIUM

- Generated Interactive Map with Folium
- The notebook has been shared Github link



DASHBOARD WITH PLOTLY

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

PREDICTIVE ANALYIS

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

TASK 5

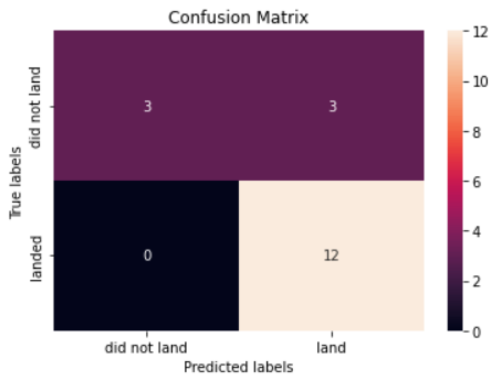
Calculate the accuracy on the test data using the method `score` :

In [25]:
```
print('The test data acurracy has been determined as: {:.3f}'.format(logreg_cv.score(X_test, Y_test)))
```

```
The test data acurracy has been determined as: 0.833
```

Lets look at the confusion matrix:

In [26]:
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the major problem is false positives.

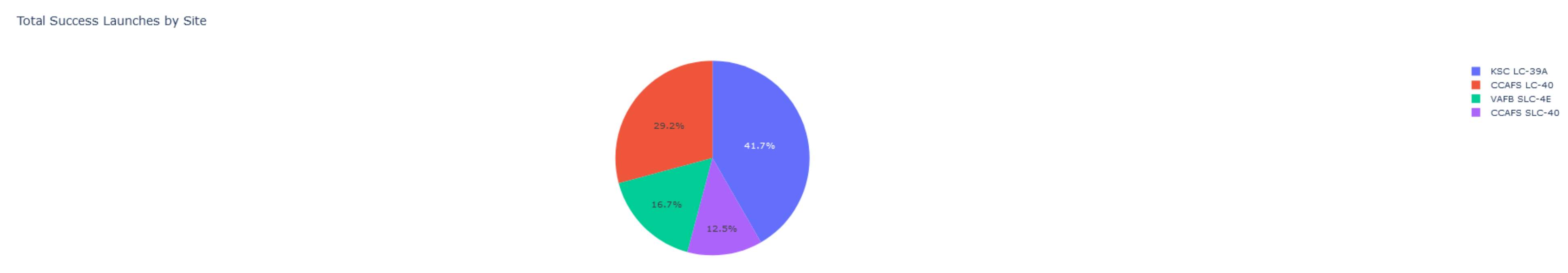
RESULTS

- 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

DASHBOARD

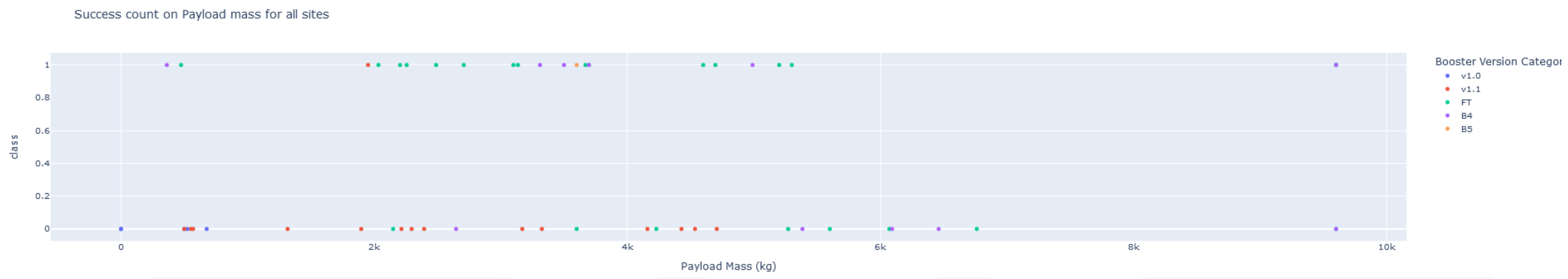
permanent link of the dashboard

DASHBOARD TAB 1

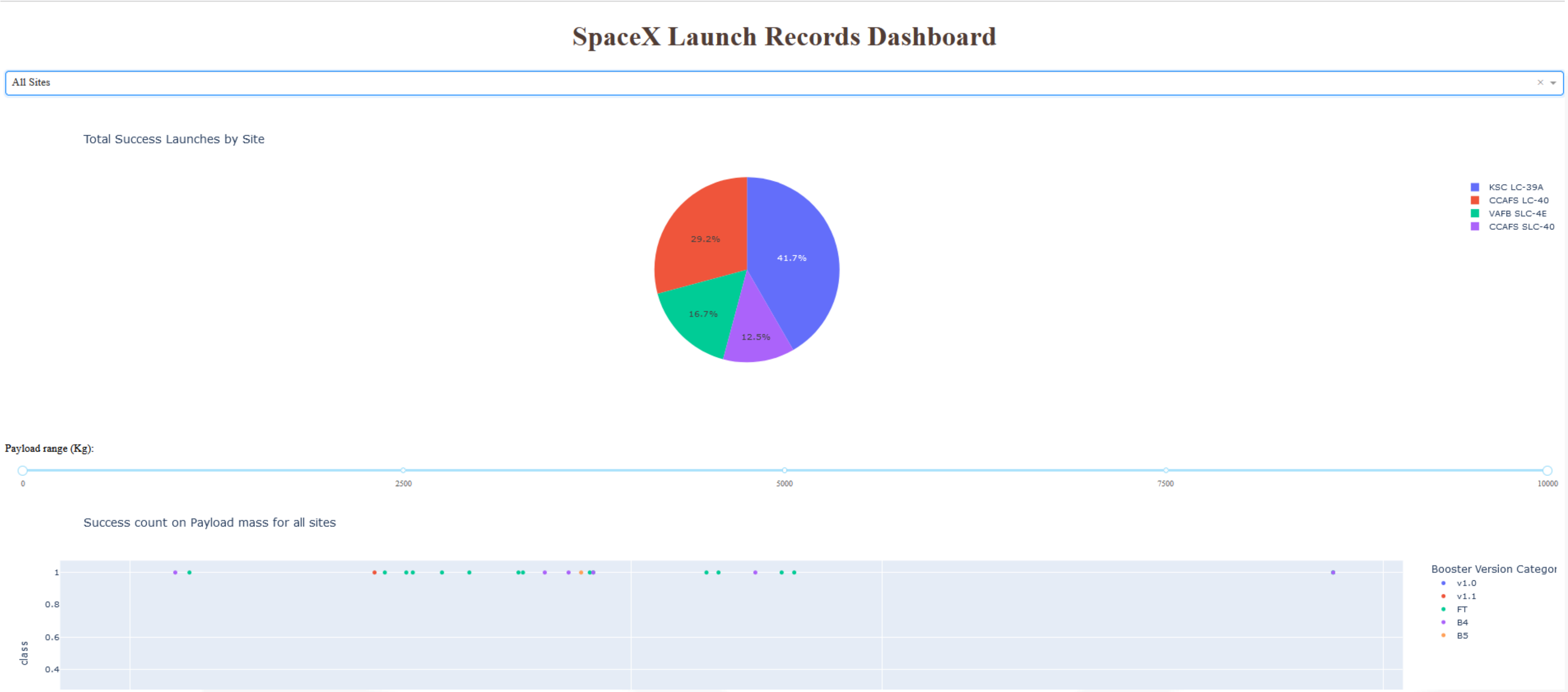


Payload range (Kg):

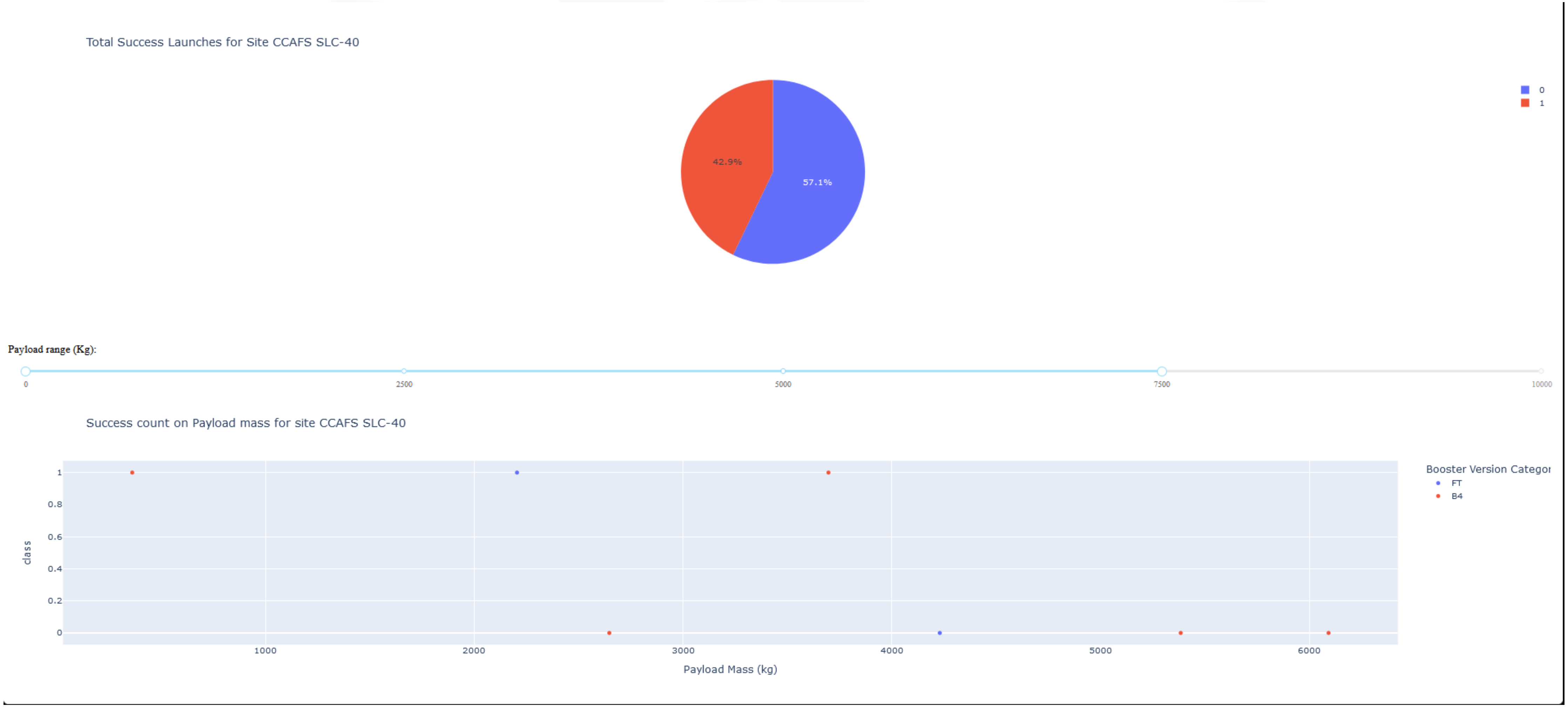
0 2500 5000 7500 10000



DASHBOARD TAB 2



DASHBOARD TAB 3



DISCUSSION

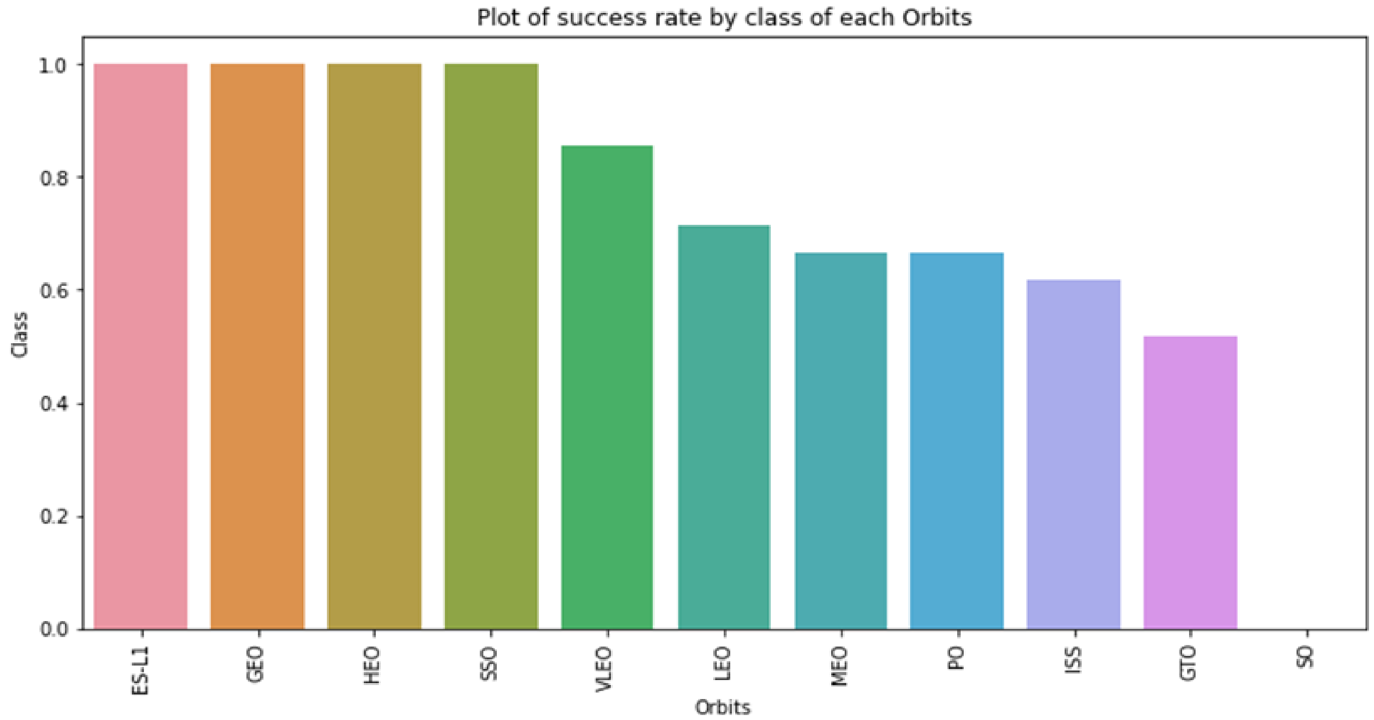
- It was found that :
  - Best Performing method is DecisionTree with score 0.8732142857142856
  - Best parameters are : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
- The accuracy for landing prediction seems to be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

CONCLUSION

- Success Rate vs Orbit Type
- Flight Number vs Orbit Type
- Success in landing with heavy payloads is more for Polar, LEO and ISS orbits
- Flight number increase results in number of successful landings

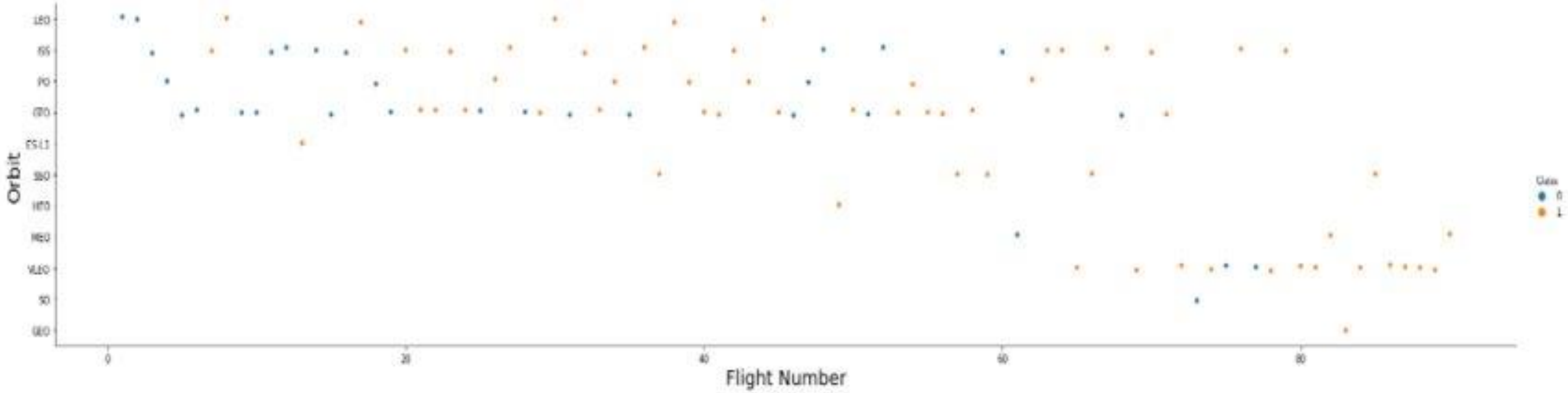
Success Rate vs. Orbit Type

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



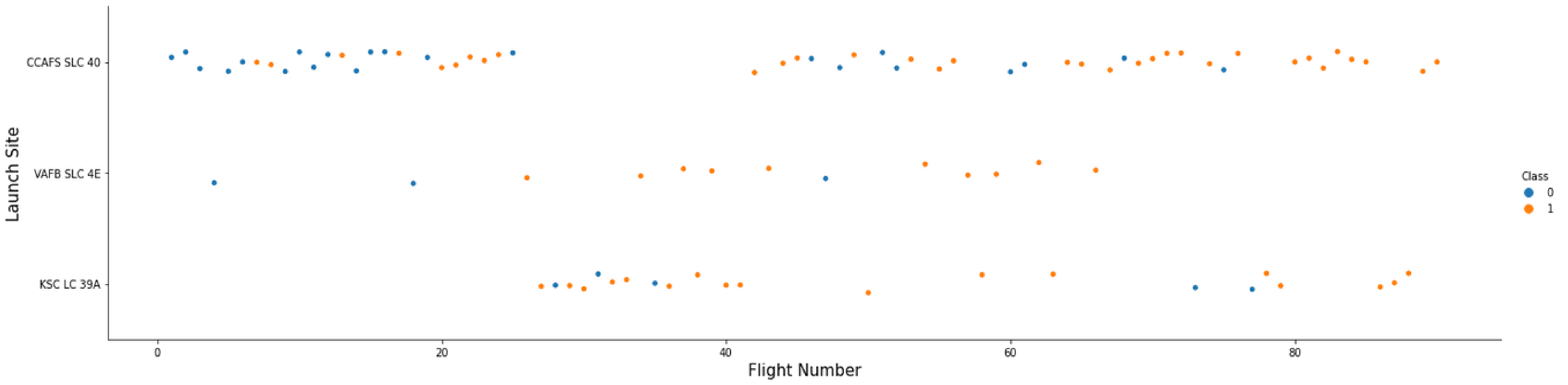
Flight Number vs. Orbit Type

- 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.



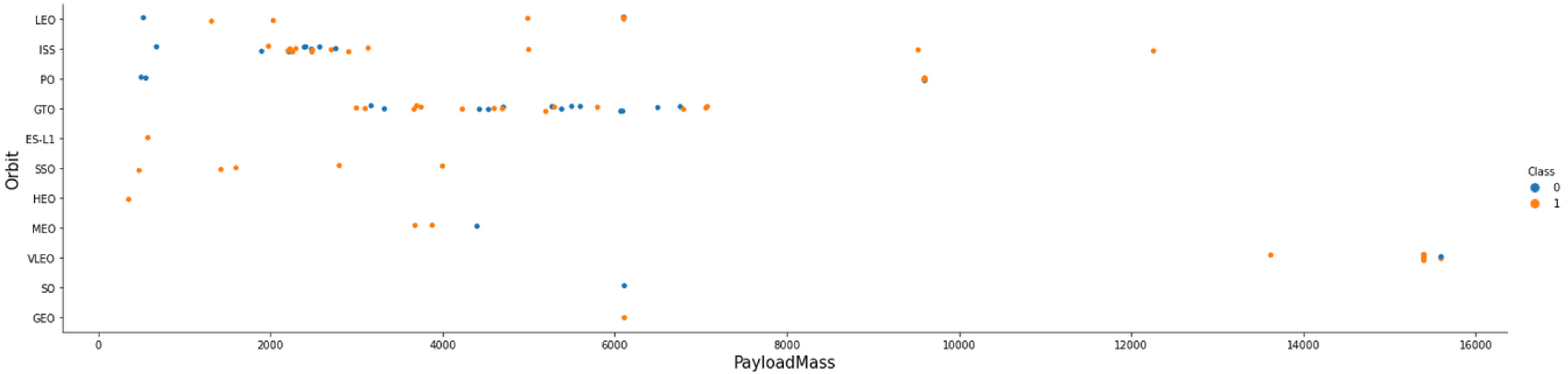
Payload vs. Launch Site

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



Payload vs Orbit Type

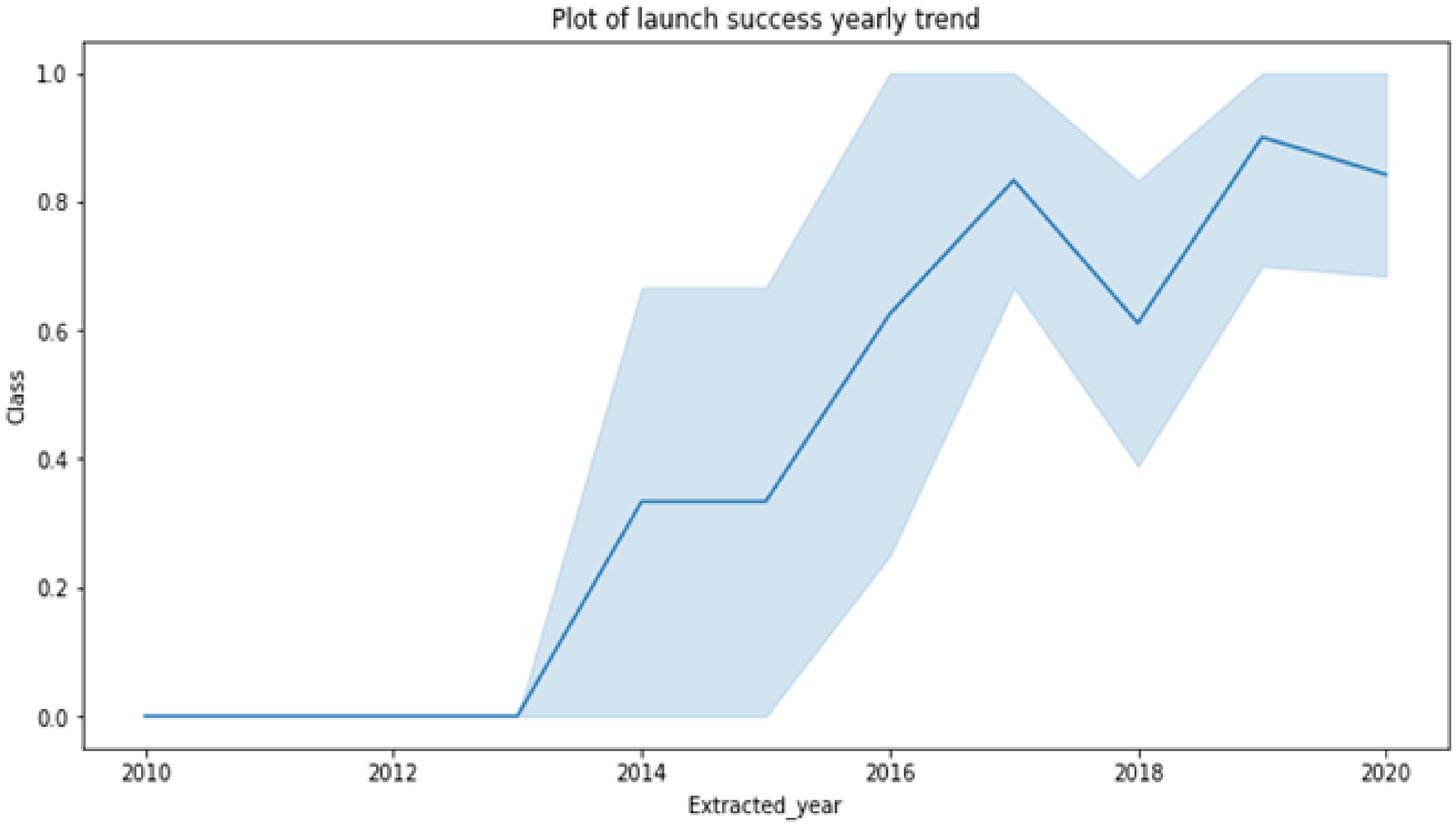
- Heavy payloads have successful landing with Polar, Leo and ISS



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.

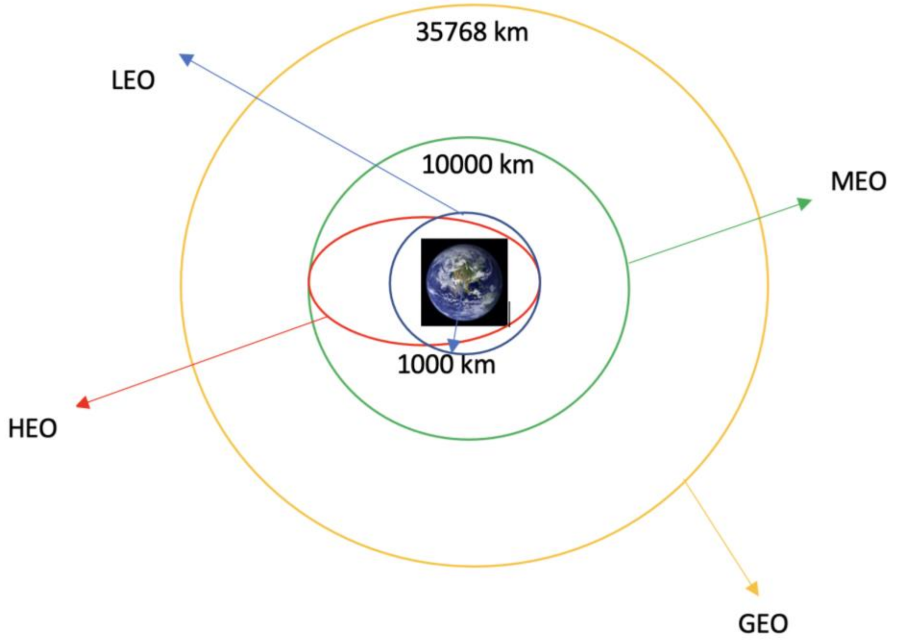
APPENDIX



Plot of launch success yearly trend
1.0
0.8
0.6
0.4
0.2
0.0
Class
2010
2012
2014
2016
2018
2020
Extracted_year

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.