Winning Space Race with Data Science

Joel Tan

29 Jan 2022

Outline

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

Executive Summary

Summary of methodologies

1. Data Collection
2. Exploratory Data Analysis
3. Create interactive Maps with Folium
4. Create Dashboard with Plotly
5. Run Machine Learning Models

Summary of all results

1. EDA
2. Maps and Dashboard
3. Result of the different models

Introduction

Project Background

Taken from Week 1 Introduction:
“In this capstone, we will predict if the Falcon 9 first stage will land successfully. 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. 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.”

Section 1

Methodology

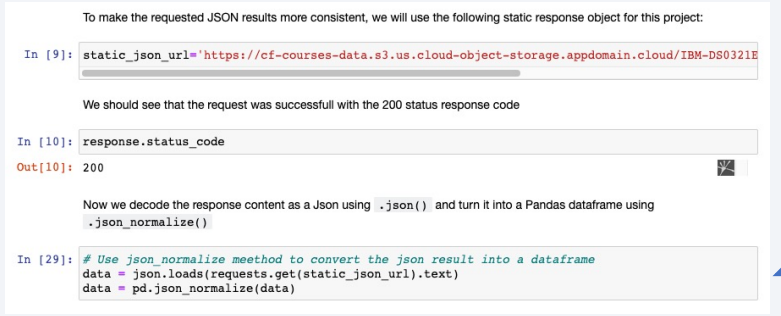
Methodology

Executive Summary

- Data collection methodology:
  - Data was collected from SpaceX API and from web scraping from Wikipedia
- Perform data wrangling
  - Irrelevant columns were dropped, new columns created and renamed, Falcon 9 data was isolated and finally null values were filled.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Standardize the data
  - Split data into training and test sets
  - Find the model that performs the best

Data Collection – SpaceX API

GitHub Link here



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-DS0321E
```

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 [29]: # Use json_normalize meethod to convert the json result into a dataframe
data = json.loads(requests.get(static_json_url).text)
data = pd.json_normalize(data)
```

Request and parse the SpaceX launch data using GET request

```
In [33]: #Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

These functions will apply the outputs globally to the above variables. Let's take a looks at `BoosterVersion` variable. Before we apply `getBoosterVersion` the list is empty:

```
In [34]: BoosterVersion
Out[34]: []
```

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

```
In [35]: # Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

```
In [36]: BoosterVersion[0:5]
Out[36]: ['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

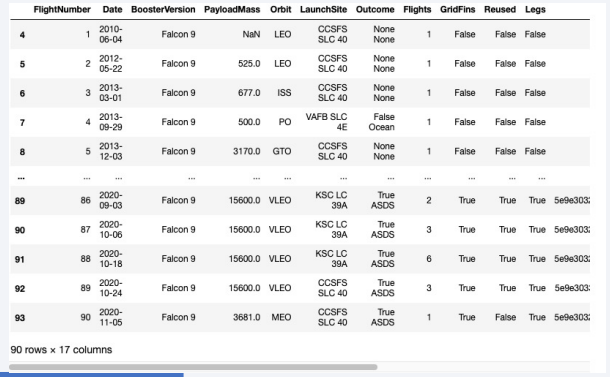
we can apply the rest of the functions here:

```
In [37]: # Call getLaunchSite
getLaunchSite(data)

In [38]: # Call getPayloadData
getPayloadData(data)

In [39]: # Call getCoreData
getCoreData(data)
```

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

```
In [40]: launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
```

Select required data from the response and store them into separate list



	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...
89	86	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e303
90	87	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e303
91	88	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e303
92	89	2020-10-24	Falcon 9	15600.0	VLEO	CCSFS SLC 40	True ASDS	3	True	True	True	5e9e303
93	90	2020-11-05	Falcon 9	3681.0	MEO	CCSFS SLC 40	True ASDS	1	True	False	True	5e9e303

90 rows × 17 columns

Create a Data Frame from the list of data collected

Data Collection - Scraping

GitHub URL here

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.

```
In [77]: # use requests.get() method with the provided static_url
         # assign the response to a object
         r = requests.get(url = static_url)
```

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

```
In [78]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
         html_response = BeautifulSoup(r.text,"html5lib")
```

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

```
In [79]: # Use soup.title attribute
         html_response.title
Out[79]: <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`, please check the external reference link towards the end of this lab

```
In [80]: # Use the find_all function in the BeautifulSoup object, with element type `table`
         # Assign the result to a list called `html_tables`
         html_tables = html_response.find_all(name = 'table')
```

Starting from the third table is our target table contains the actual launch records.

```
In [81]: # Let's print the third table and check its content
         first_launch_table = html_tables[2]
         #print(first_launch_table)
```

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

```
[84]: launch_dict= dict.fromkeys(column_names)

      # Remove an irrelvant column
      del launch_dict['Date and time ( )']

      # Let's initial the launch_dict with each value to be an empty list
      launch_dict['Flight No.'] = []
      launch_dict['Launch site'] = []
      launch_dict['Payload'] = []
      launch_dict['Payload mass'] = []
      launch_dict['Orbit'] = []
      launch_dict['Customer'] = []
      launch_dict['Launch outcome'] = []
      # Added some new columns
      launch_dict['Version Booster']=[]
      launch_dict['Booster landing']=[]
      launch_dict['Date']=[]
      launch_dict['Time']=[]
```

Next, we just need to fill up the `launch_dict` with launch records extracted from table rows.

Usually, HTML tables in Wiki pages are likely to contain unexpected annotations and other types of noises, such as reference links `B0004.1[8]`, missing values `N/A [e]`, inconsistent formatting, etc.

To simplify the parsing process, we have provided an incomplete code snippet below to help you to fill up the `launch_dict`. Please complete the following code snippet with TODOs or you can choose to write your own logic to parse all launch tables:

```
[85]: extracted_row = 0
      #Extract each table
      for table_number,table in enumerate(html_response.find_all('table',"wikitable plainrowheaders c
          # 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')
```

Final Result

```
In [16]: df
```

Out[16]:

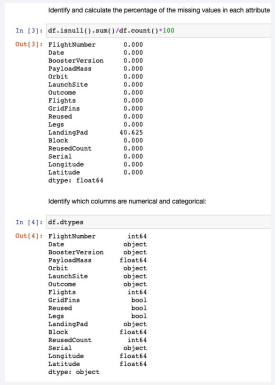
	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10
...	...	...	...	...	...	...	...	...	...	...	...
116	117	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1051.10	Success	9 May 2021	06:42
117	118	KSC	Starlink	~14,000 kg	LEO	SpaceX	Success\n	F9 B5B1058.8	Success	15 May 2021	22:56
118	119	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1063.2	Success	26 May 2021	18:59
119	120	KSC	SpaceX CRS-22	3,328 kg	LEO	NASA	Success\n	F9 B5B1067.1	Success	3 June 2021	17:29
120	121	CCSFS	SXM-8	7,000 kg	GTO	Sirius XM	Success\n	F9 B5	Success	6 June 2021	04:26

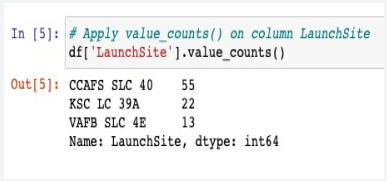
121 rows × 11 columns

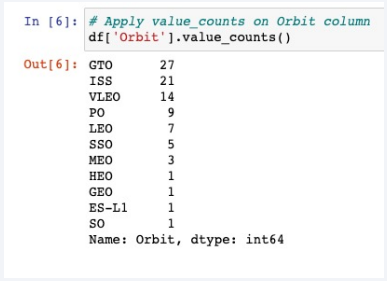
Data Wrangling

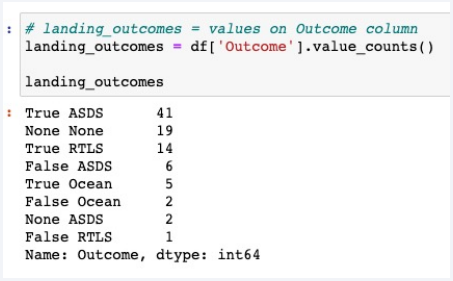
GitHub URL here

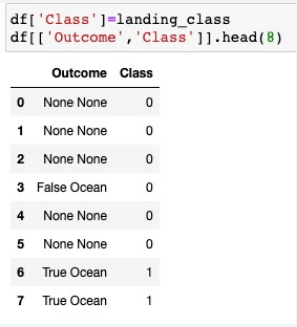
- we will perform some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.
- In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad. True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.
- We will mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.











Perform simple EDA: look for null values, and identify data type

Calculate the number of launches on each site

Calculate the number and occurence of mission outcome per orbit type

Calculate the number and occurrence of each orbit

Create a landing outcome label from Outcome column

EDA with Data Visualization

GitHub URL here

Charts Used

1. Scatter plot relationship between Flight Number and Launch Site
2. Scatter plot relationship between Payload and Launch Site
3. Bar plot relationship between success rate of each orbit type
4. Scatter plot relationship between Flight Number and Orbit type
5. Scatter plot relationship between Payload and Orbit type
6. Line plot launch success yearly trend

EDA with SQL

GitHub URL here

SQL queries performed

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string 'CCA’
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome in ground pad was achieved.
6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. List the total number of successful and failure mission outcomes
8. List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
9. List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
10. Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Build an Interactive Map with Folium

GitHub URL here

- Objects created and added to folium map

1. Mark all launch sites on the map
2. Draw an Equator line
3. Mark the success/failed launches for each site on the map
4. Mark and trace the nearest: city, railroad, highway and coastline to a launch site.

- Explanation

Find some geographical patterns about launch sites because:

“The launch success rate may depend on many factors such as payload mass, orbit type, and so on. It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories. Finding an optimal location for building a launch site certainly involves many factors and hopefully we could discover some of the factors by analyzing the existing launch site locations.”

Build a Dashboard with Plotly Dash

GitHub URL here

- Plots/graphs and interactions added to dashboard

1. Add a Launch Site Drop-down Input Component
2. Created a Pie Chart to show rate of successful launches
3. Added a Range Slider to filter Payload Range
4. Created a Scatter plot to show different Payload success or failure

- Explanation

This would give insights into SpaceX successful launch data and answer questions such as:

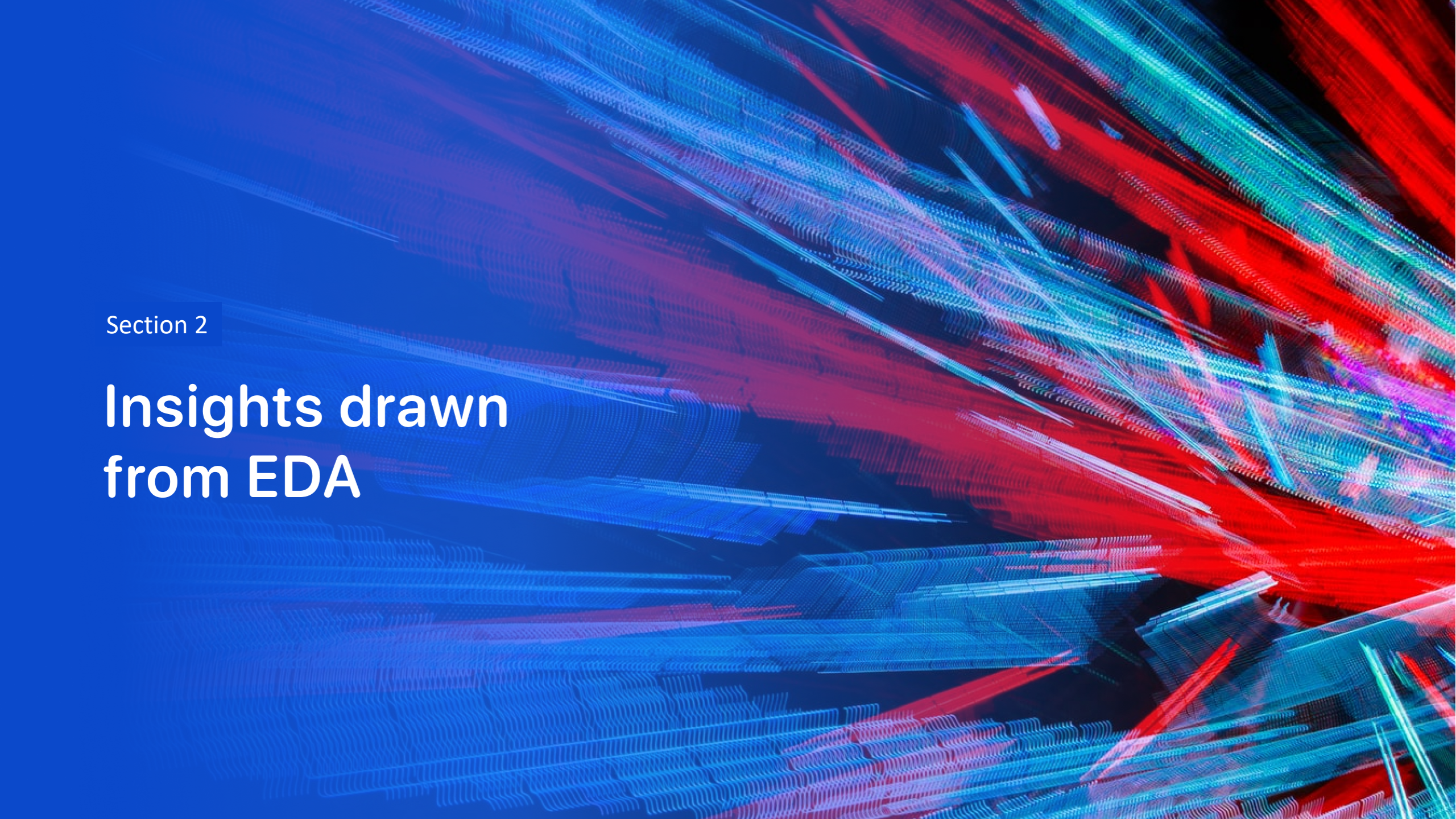
- Which site has the largest successful launches?
- Which site has the highest launch success rate?
- Which payload range(s) has the highest launch success rate?
- Which payload range(s) has the lowest launch success rate?
- Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest

Predictive Analysis (Classification)

GitHub URL here

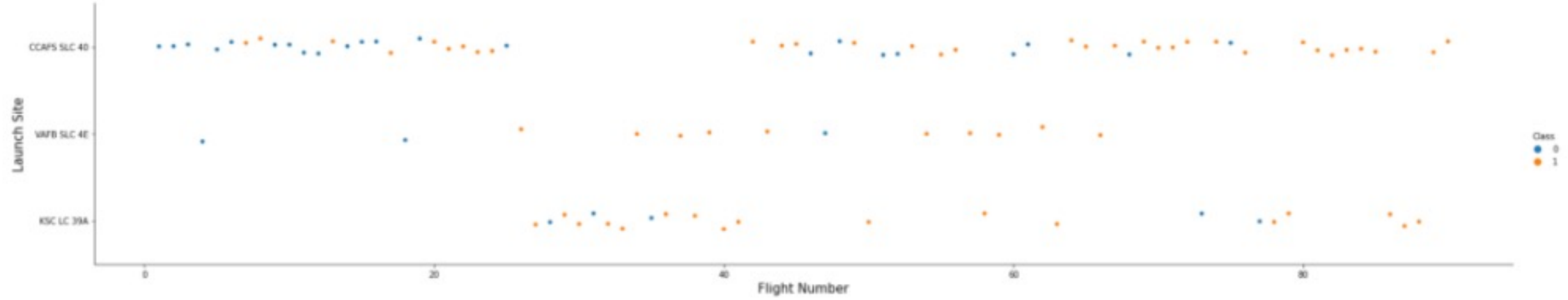
- Perform exploratory Data Analysis to determine Training Labels
  - create a column for the class
  - Standardize the data
  - Split into training data and test data
- Find best Hyperparameter for SVM, Classification Trees, K Nearest Neighbour and Logistic Regression
- Find and compare the method that performs best using test data

	Accuracy	Test Score
Logistic Regression	0.846429	0.833333
Support Vector Machine	0.848214	0.833333
Decision Tree	0.887500	0.777778
k nearest neighbors	0.848214	0.833333


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

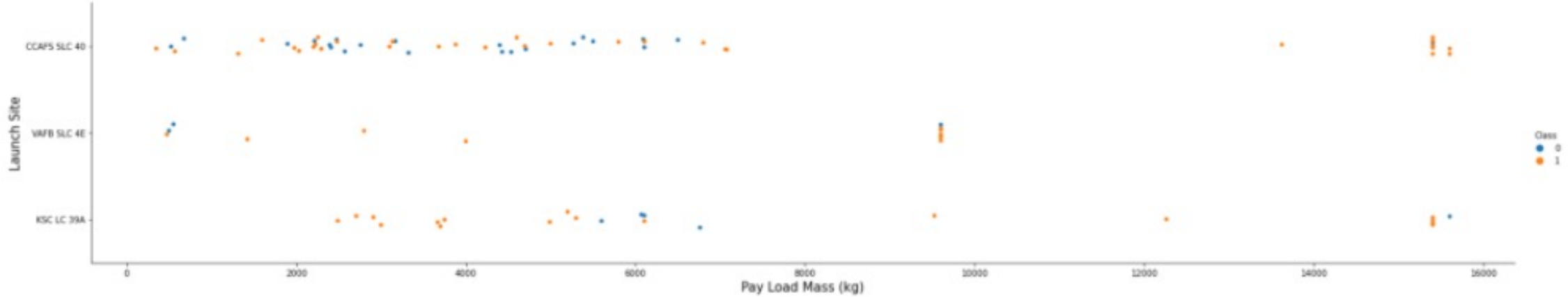
In [4]:
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class
sns.catplot(y='LaunchSite',x='FlightNumber',hue ='Class',data=df, aspect = 5)
plt.xlabel('Flight Number',fontsize=15)
plt.ylabel('Launch Site',fontsize=15)
plt.show()
```



The outcome improves the more flights there are

Payload vs. Launch Site

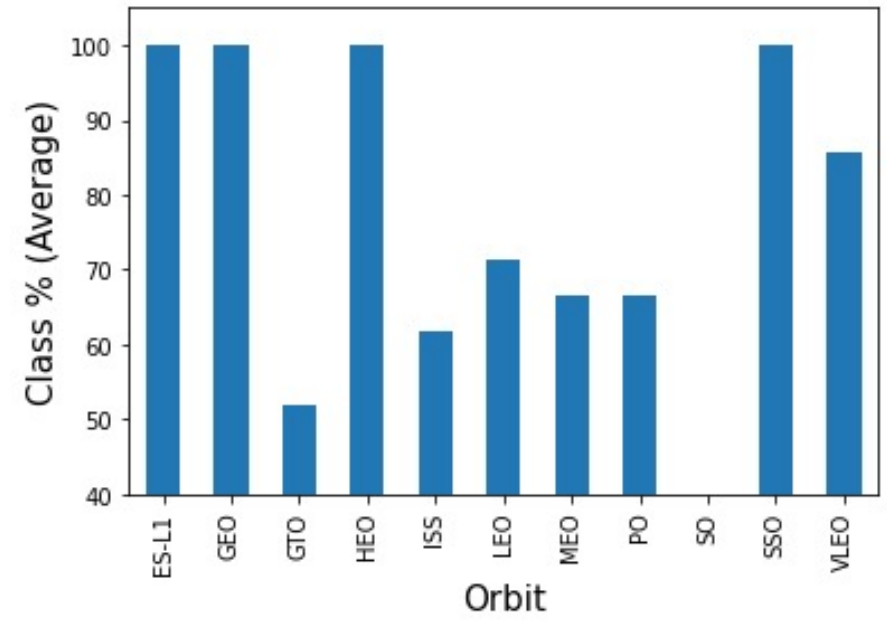
In [5]:
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the c
sns.catplot(y='LaunchSite',x='PayloadMass',hue ='Class',data=df, aspect =5)
plt.xlabel('Pay Load Mass (kg)',fontsize=15)
plt.ylabel('Launch Site',fontsize=15)
plt.show()
```



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

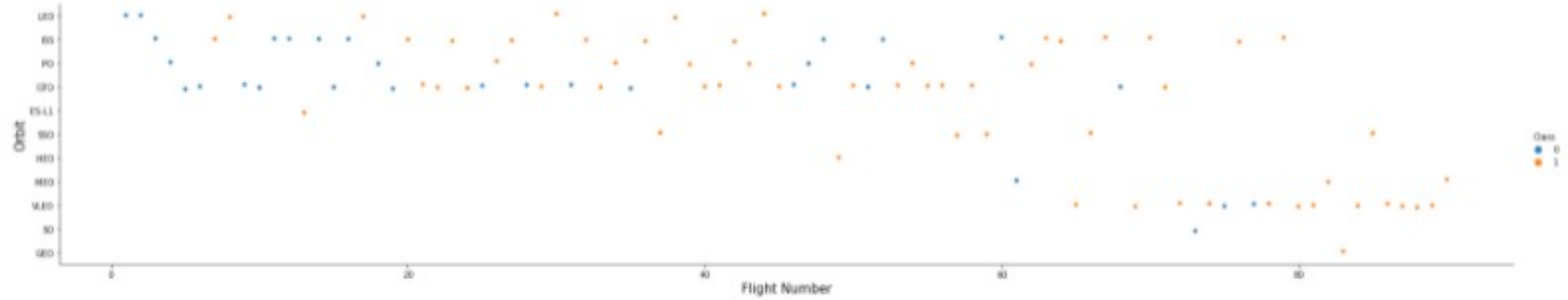
```
bar_df = pd.DataFrame(df.groupby(df['Orbit'])['Class'].mean()*100)
bar_df.plot(kind ='bar',legend = False)
plt.ylim(40,105)
plt.xlabel("Orbit",fontsize=15)
plt.ylabel("Class % (Average)",fontsize=15)
plt.show()
```



ES-L1, GEO, HEO and SSO have 100% success rate

Flight Number vs. Orbit Type

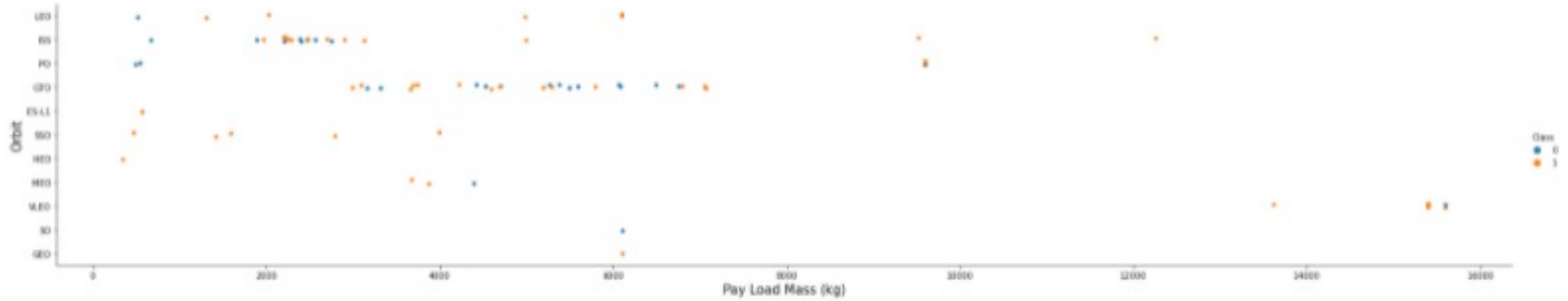
In [6]:
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue
sns.catplot(y='Orbit',x='FlightNumber',hue ='Class',data=df,aspect=5)
plt.xlabel('Flight Number',fontsize=15)
plt.ylabel('Orbit',fontsize=15)
plt.show()
```



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

In [7]:
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to b
sns.catplot(y='Orbit',x='PayloadMass',hue ='Class',data=df,aspect =5)
plt.xlabel('Pay Load Mass (kg)',fontsize=15)
plt.ylabel('Orbit',fontsize=15)
plt.show()
```

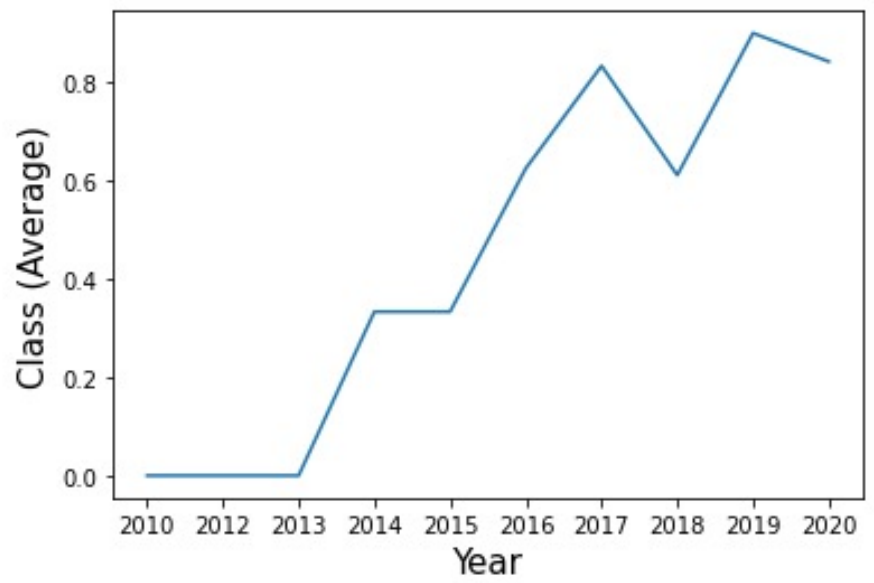


With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

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

```
df['Year'] = Extract_year(df['Date'])
df_year = pd.DataFrame(df.groupby('Year')['Class'].mean())

sns.lineplot(y="Class", x="Year", data=df_year)
plt.xlabel("Year",fontsize=15)
plt.ylabel("Class (Average)",fontsize=15)
plt.show()
```



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

All Launch Site Names

```
In [12]: %sql SELECT unique(launch_site) FROM SPACEXDATASET;
 * ibm_db_sa://fgg88173:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348
es.appdomain.cloud:32286/bludb
Done.
```

Out[12]:

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

There are 4 unique launch sites

Launch Site Names Begin with 'CCA'

```
%%sql

SELECT * FROM SPACEXDATASET
WHERE launch_site LIKE 'CCA%'
LIMIT 5;
```

```
 * ibm_db_sa://fgg88173:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:322
6/bludb
Done.
```

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

5 records where launch sites begin with `CCA`

Total Payload Mass

```
%%sql

SELECT SUM(payload_mass__kg_) as "Total Payload Mass NASA(CRS)" FROM SPACEXDATASET
WHERE customer  = 'NASA (CRS)';
```

```
 * ibm_db_sa://fgg88173:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.
es.appdomain.cloud:32286/bludb
Done.
```

Total Payload Mass NASA(CRS)
45596

The total payload carried by boosters from NAS 45596

Average Payload Mass by F9 v1.1

```
%%sql

SELECT AVG(payload_mass__kg_) as "Avg Payload Mass" FROM SPACEXDATASET
WHERE booster_version LIKE 'F9 v1.1%'
;
```

```
 * ibm_db_sa://fgg88173:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98
es.appdomain.cloud:32286/bludb
Done.
```

Avg Payload Mass
2534

The average payload mass carried by booster version F9 v1: 2534

First Successful Ground Landing Date

```
%%sql
SELECT MIN(DATE) as "Date when the first successful landing outcome in ground pad was achieved"
FROM SPACEXDATASET
WHERE landing__outcome = 'Success (ground pad)'
;
```

```
 * ibm_db_sa://fgg88173:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

Date when the first successful landing outcome in ground pad was achieved
2015-12-22

The date of the first successful landing outcome on ground pad: 22 Dec 2015

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql

SELECT UNIQUE(booster_version)
FROM SPACEXDATASET
WHERE landing__outcome = 'Success (drone ship)'
AND payload_mass__kg_ > 4000
AND payload_mass__kg_ < 6000;
```

```
 * ibm_db_sa://fgg88173:***@1bbf73c5-d84a-4bb0-85b9-
es.appdomain.cloud:32286/bludb
Done.
```

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

- Names of the 4 boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

```
%%sql

SELECT mission_outcome, COUNT(*) as "total #"
FROM SPACEXDATASET
GROUP BY mission_outcome;
```

```
 * ibm_db_sa://fgg88173:***@1bbf73c5-d84a-4bb0-8
es.appdomain.cloud:32286/bludb
Done.
```

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

The total number of successful and failure mission outcome

Boosters Carried Maximum Payload

```
%%sql

SELECT UNIQUE(booster_version) as "booster_versions which have carried the maximum payload mass
FROM SPACEXDATASET
WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXDATASET);
```

```
 * ibm_db_sa://fgg88173:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databas
es.appdomain.cloud:32286/bludb
Done.
```

booster_versions which have carried the maximum payload mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

The names of the booster which have carried the maximum payload mass

2015 Launch Records

```
%%sql
SELECT landing__outcome, payload, booster_version, launch_site
FROM SPACEXDATASET
WHERE landing__outcome LIKE 'Fail%'
AND YEAR(DATE) = '2015'
;
```

```
 * ibm_db_sa://fgg88173:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a
es.appdomain.cloud:32286/bludb
Done.
```

landing__outcome	payload	booster_version	launch_site
Failure (drone ship)	SpaceX CRS-5	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	SpaceX CRS-6	F9 v1.1 B1015	CCAFS LC-40

Failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

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

```
%%sql

SELECT landing__outcome, count(*) as "Count"
FROM SPACEXDATASET
WHERE DATE >= '2010-06-04'
AND DATE <= '2017-03-20'
GROUP BY landing__outcome
ORDER BY "Count" DESC
;
```

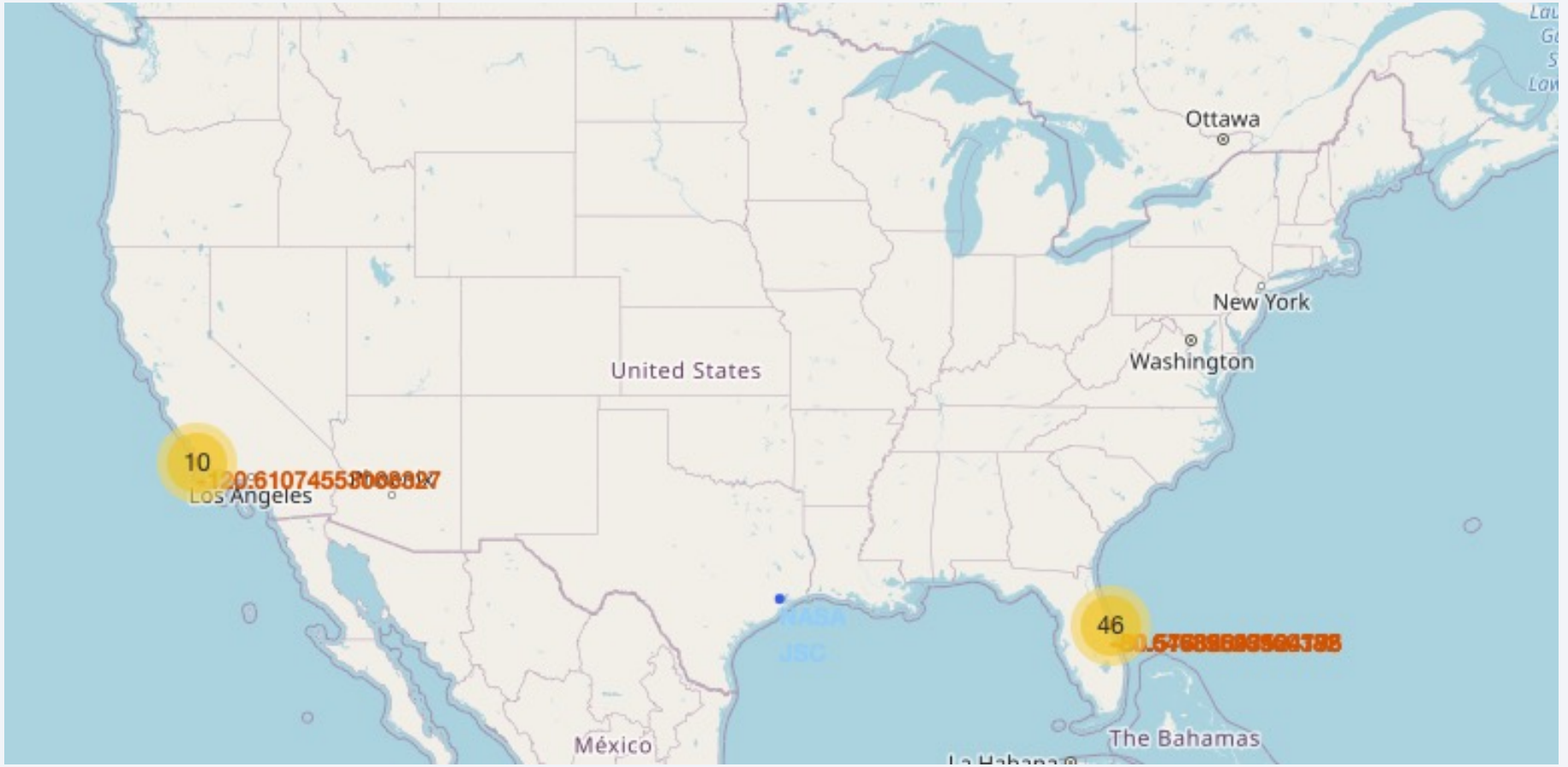
```
 * ibm_db_sa://fgg88173:***@1bbf73c5-d84a-4bb0-
es.appdomain.cloud:32286/bludb
Done.
```

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

- 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

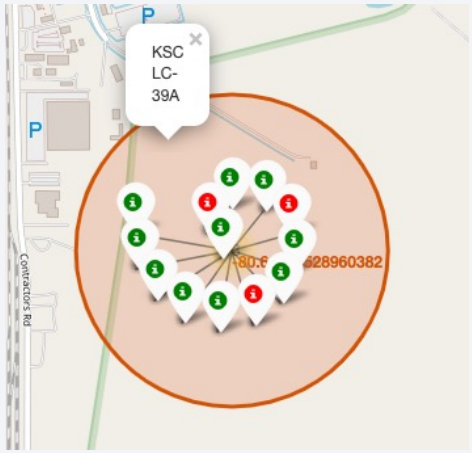

Section 3
Launch Sites
Proximities Analysis

Folium Map Location of the launch Sites



The launch sites are generally grouped together on the lower end of both the east and west coastlines.

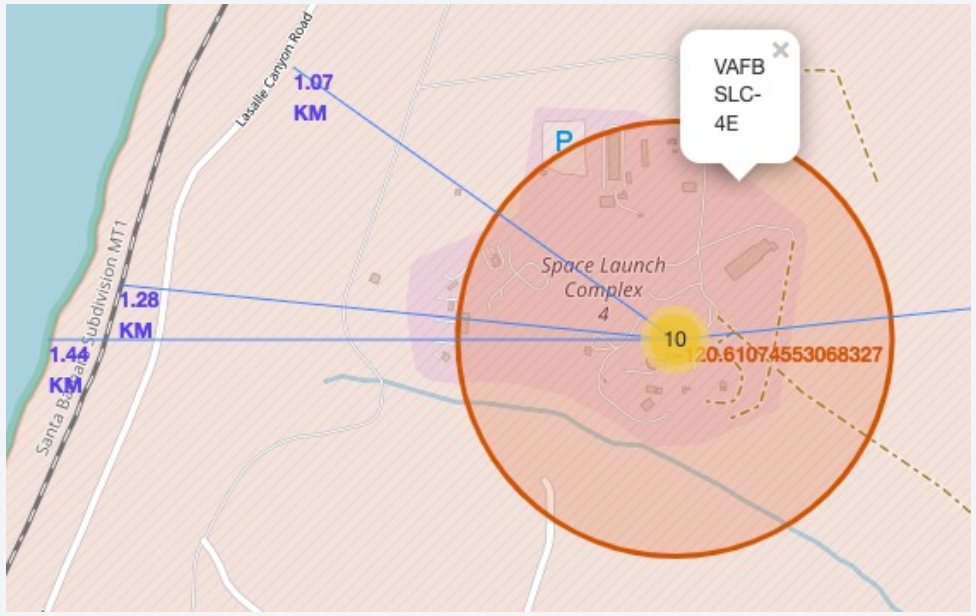
Folium Map Color Labeled Launch Outcomes



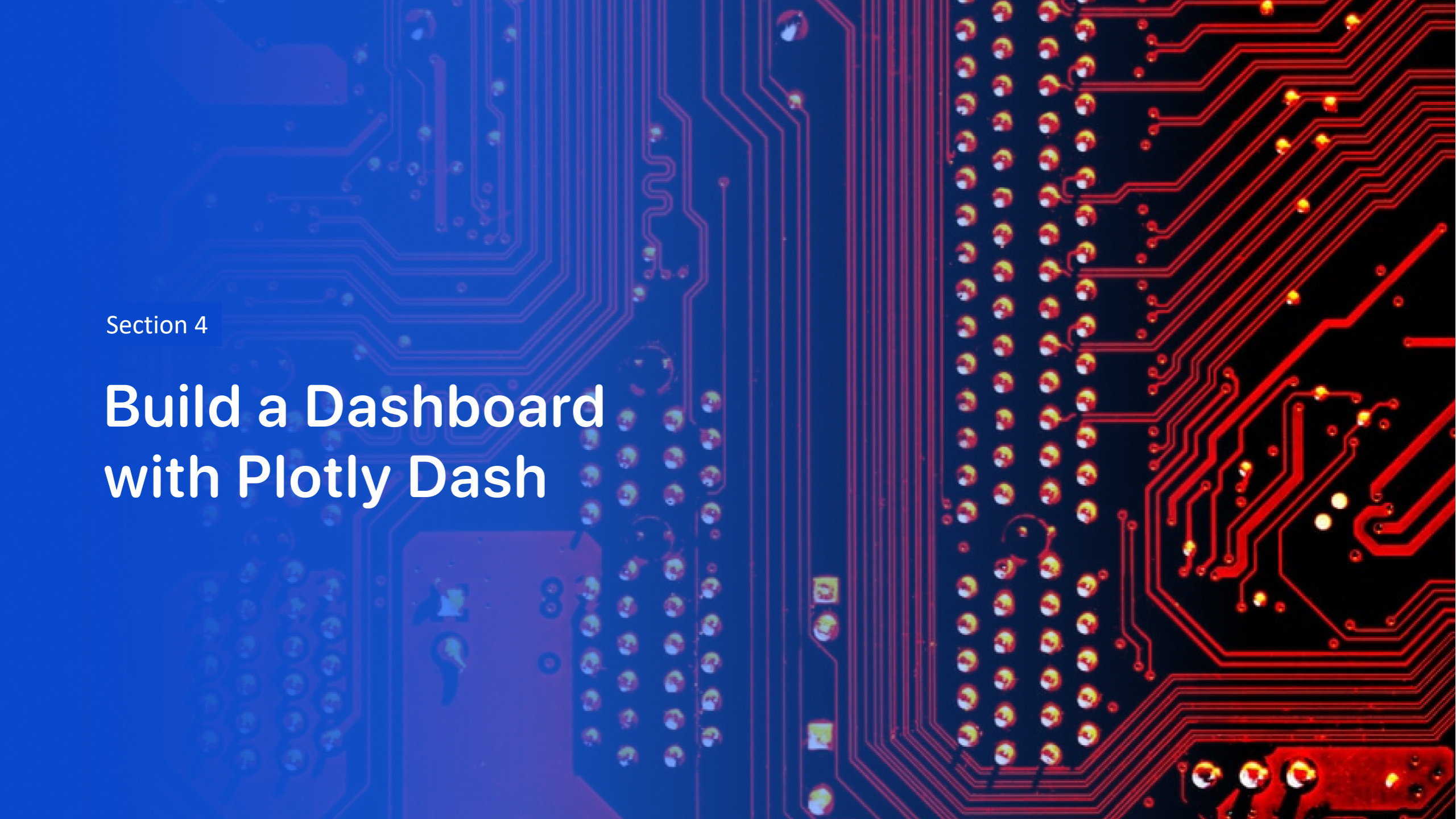
These are the color labeled launch outcomes for the different launch sites

CCAFS SLC-40 and CCAFS LC-40 are so closed together that they overlap, and it is not easy to different between the two

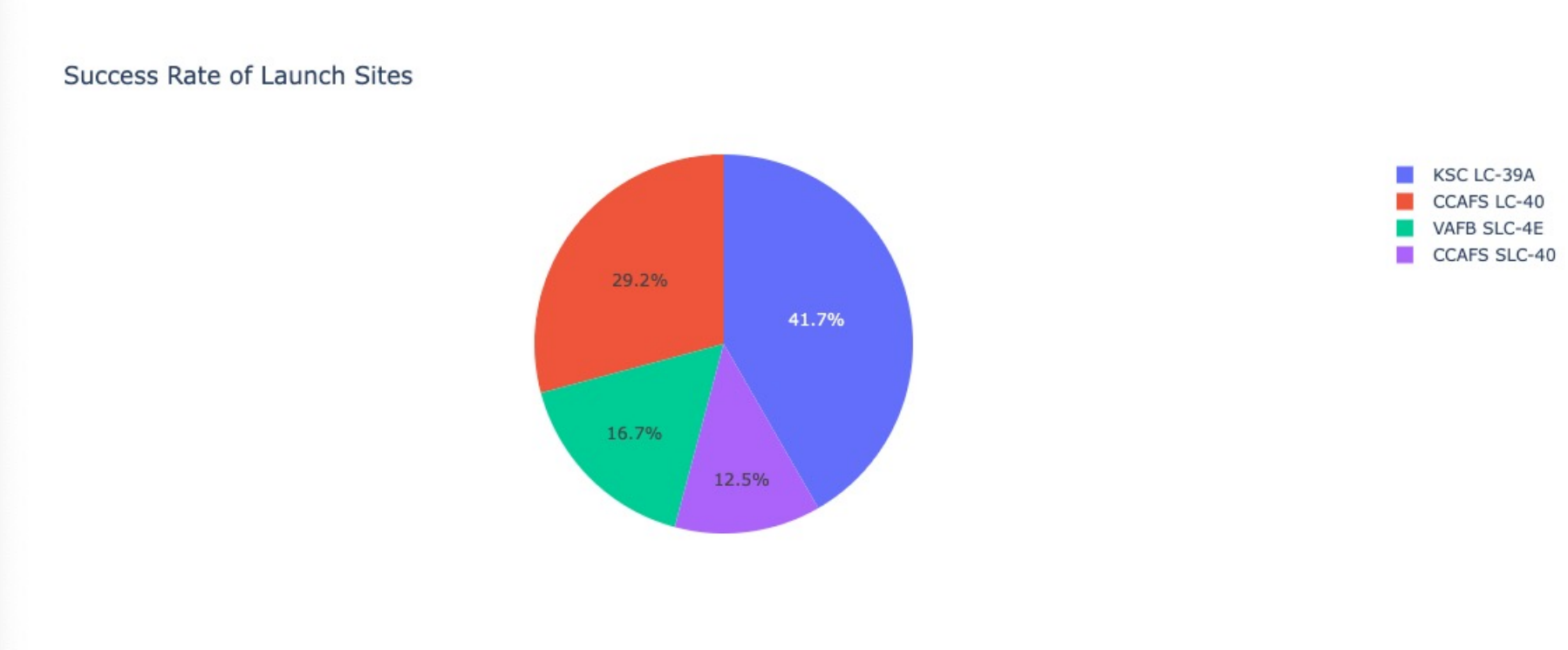
Folium Map VAFB SLC-4E launch site proximity to infrastructures



The launch site is very close to railroads, highways and the coastline.
But it is quite a distance from the nearest city, Lompoc.

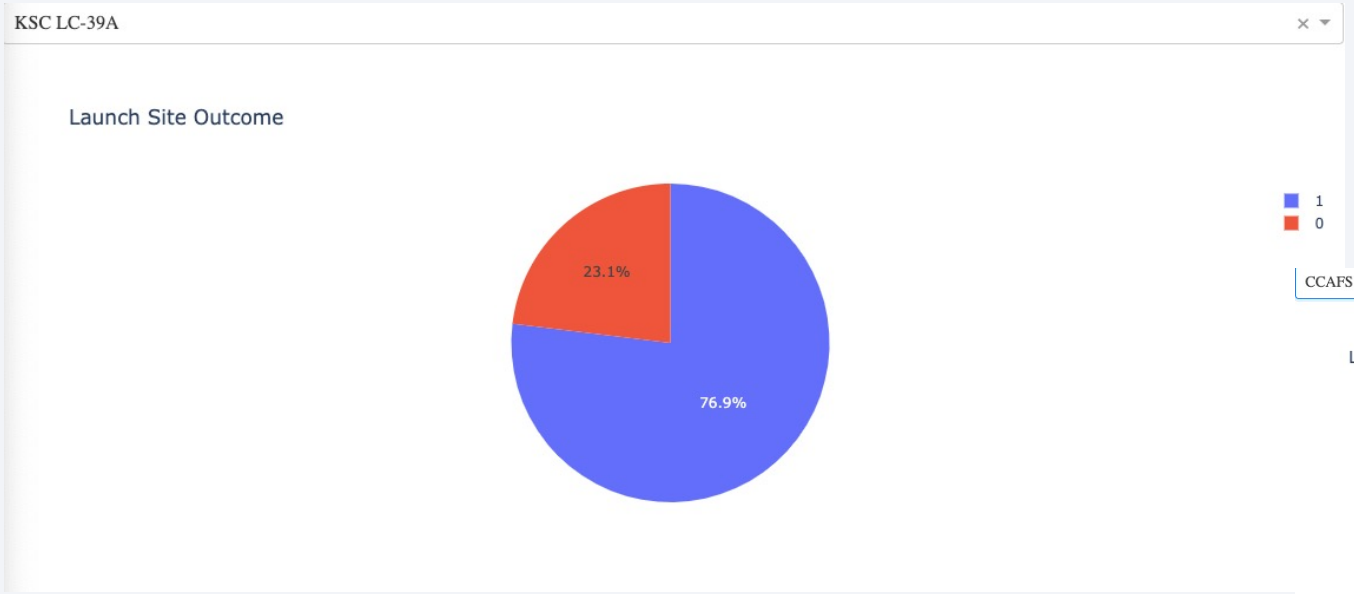

Section 4
Build a Dashboard with Plotly Dash

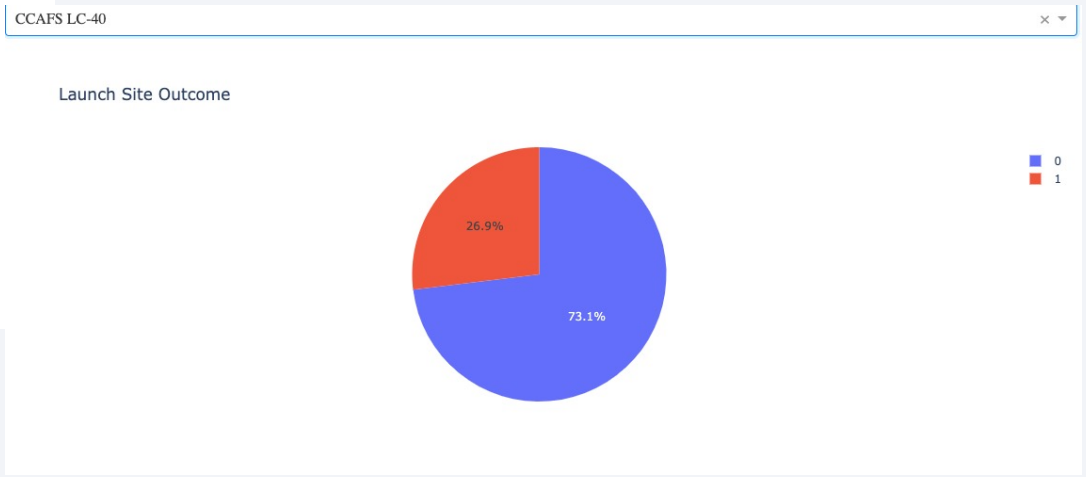
Dashboard Pie Chart of Share of Success



From this we can see that KSC LC-39A has the lion share of success at 41.7% and CCAFS SLC-40 is struggling at 12.5%

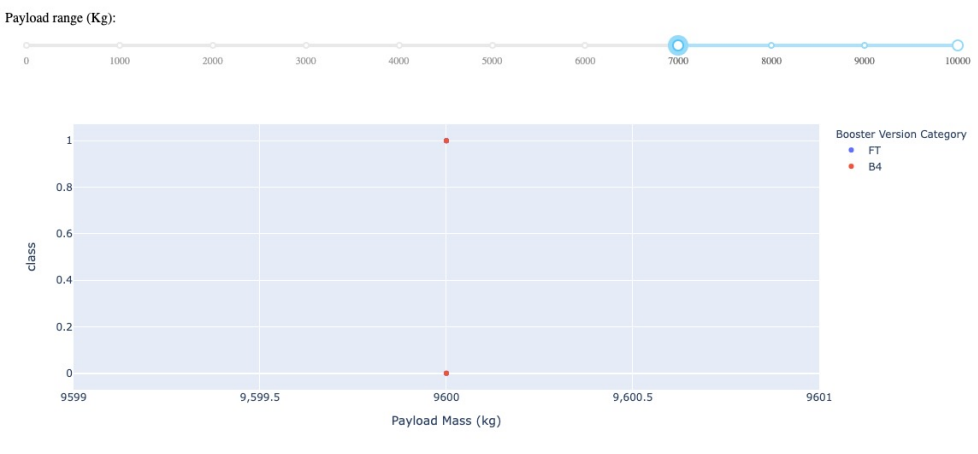
Dashboard KSC LC-39A: Highest Launch Success Ratio



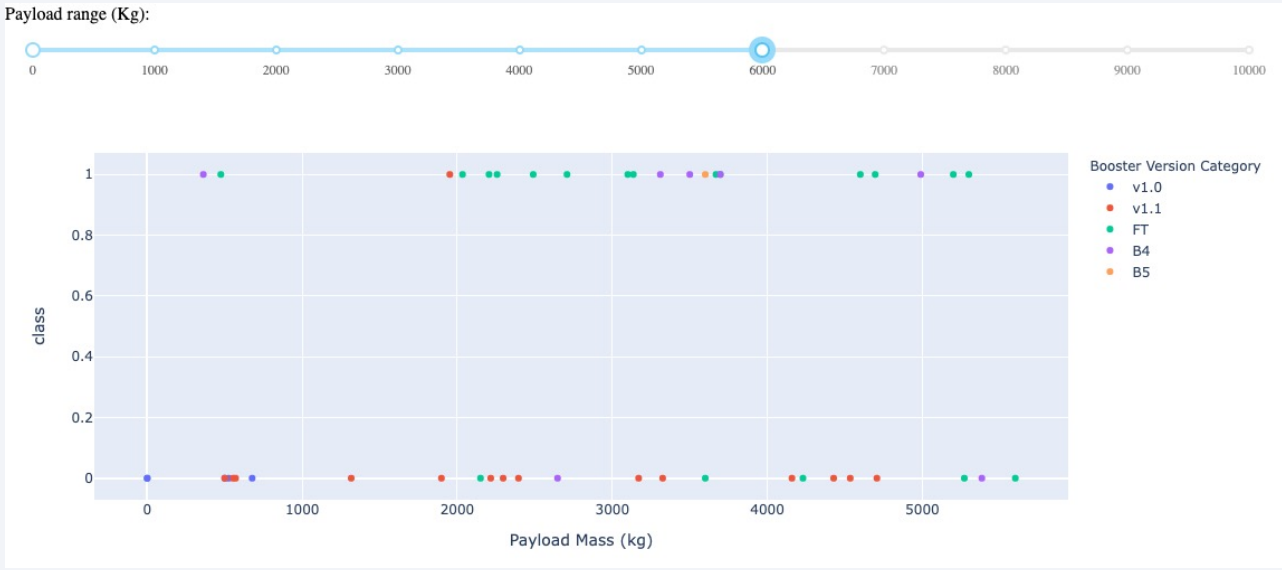


At 76.9% KSC LC-39A has the high launch success rate but CCAFS LC-40 is not that far behind with 73.1%

Dashboard Comparing Payload vs Launch Outcomes



Not many payloads above 7,000kg

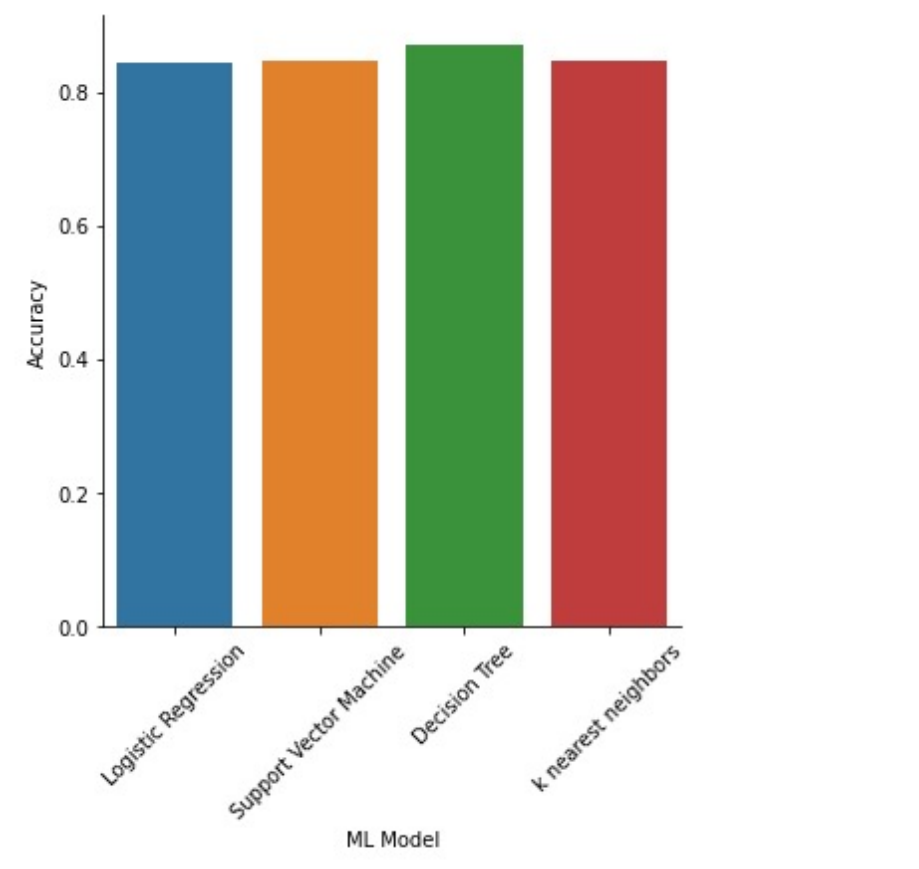


Results are fairly mixed for those 6,000kg and below

Section 5

Predictive Analysis (Classification)

Classification Accuracy

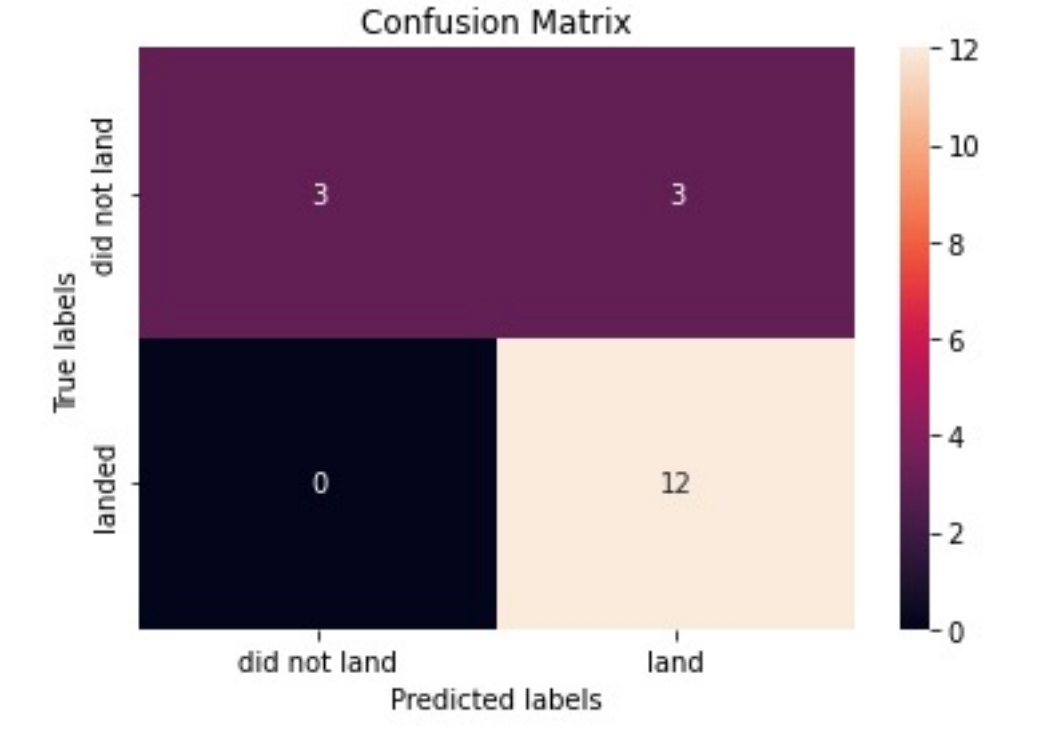


The models have all very close accuracy alittle above 84%.

However, the Decision Tree Model has a slight edge at 87%.

	Accuracy
Logistic Regression	0.846429
Support Vector Machine	0.848214
Decision Tree	0.873214
k nearest neighbors	0.848214

Confusion Matrix for Decision Tree



This is the confusion matrix on the test data results in comparison to the actual result.
For land, the classifier had a False Positive for 3 of the data.
The rest it has managed to predict accurately.

Conclusions

- The Decision Tree was the best out of the 4 classifiers in terms of accuracy, however its edge is only 2% odd. In terms of the confusion matrix, it actually performed similarly to the other models.
- The success of a mission drastically increase as the team has more launches under their belt.
- Launch Sites seems to favor areas far from the city, but near to highways, railroads and the coastlines.
- KSC LC-39A could be considered the most successful of the launch sites perhaps more studies could be done on this particular site to better improve the success rate of the others.

Appendix

- NA

Thank you!