Winning Space Race with Data Science

<Name>
<Date>

Outline

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

Executive Summary

- **Summary of methodologies**

The project is based on the space launches of the Space X's Falcon 9 and Falcon Heavy launches. The Falcon design features reusable first – stage boosters which lands either on the ground pad near the launch site or on a drone ship into the sea. This project is based on the prediction whether the boosters will land or not. This project is made with the help of different types of software tools and scripted in the jupyter notebook for the most part. SQL queries are used to extract required data from the tables, the library 'BeautifulSoup' is being used to extract data from the web from the Wikipedia page and the scikit learn library to predict the landing outcomes of the boosters. We have also created a web application with the help of DASH which has interactive visualization consisting of pie charts and scatter plots.

- **Summary of all results**

While analyzing the prediction of the boosters on the basis of data we had different approaches. Initially what we did all the necessary data collection, wrangling and visualizing. The visualization was done with the help of web application created with the help of dash. Then when it comes to predicting the landing outcomes we tried the following machine learning techniques: Logistic regression, support vector, K nearest neighbor and Decision tree. Which ever had the best accuracy score was considered to be the predicting value and the method for the further process.

Introduction

- This Project is based on the falcon rockets launched by Space X. Our mission is to make this project as cost effective as we can.
- For making it cost effective we need to predict whether the boosters will land on the particular site or not. Once landed they can be reused for another mission hence making it cost effective.

Section 1

Methodology

Methodology

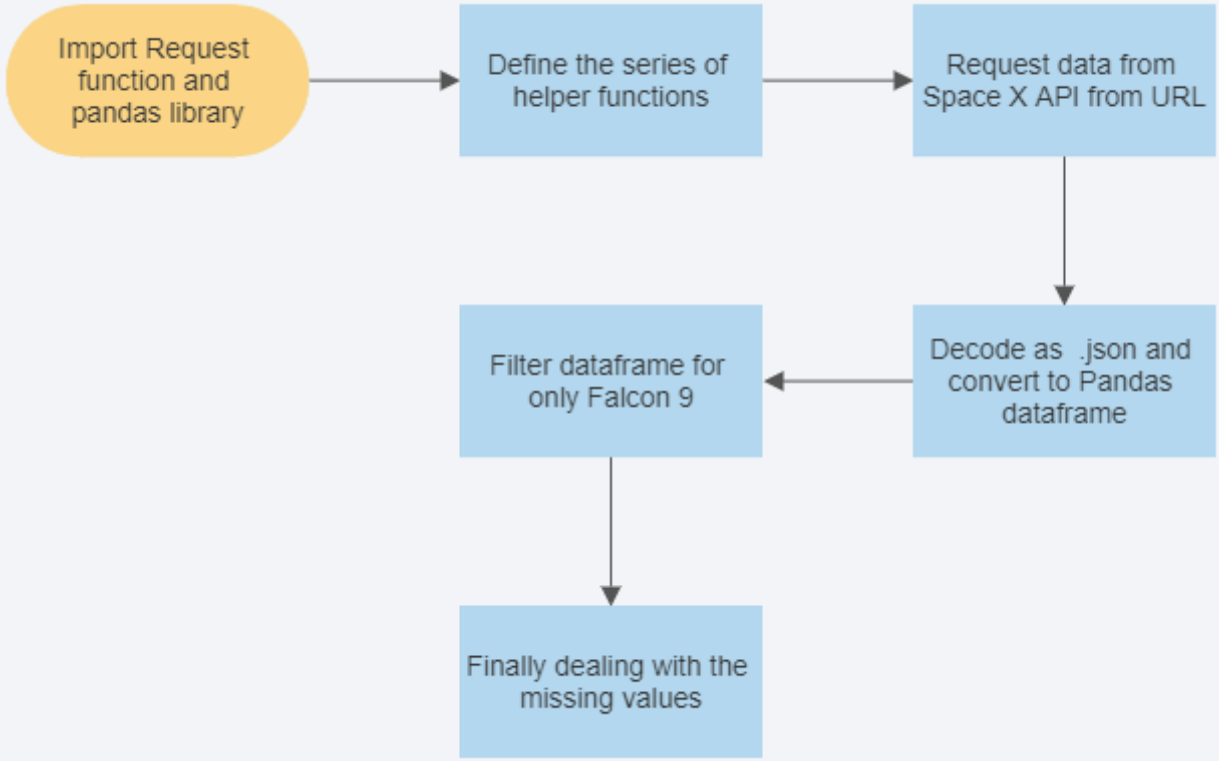
Executive Summary

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

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

- Add the GitHub URL of the completed SpaceX API calls notebook (must include completed code cell and outcome cell), as an external reference and peer-review purpose



Importing required Libraries

In [1]:
```
# Requests allows us to make HTTP requests which we will use to get data from an API
import requests
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
# NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays
import numpy as np
# Datetime is a library that allows us to represent dates
import datetime
```

Define the series of helper functions

In [3]:
```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

In [4]:
```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
        response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
        Longitude.append(response['longitude'])
        Latitude.append(response['latitude'])
        LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

In [5]:
```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

Requesting data from Space X API from URL and decode to .json file, convert to pandas dataframe

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

In [7]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [8]:
```
response = requests.get(spacex_url)
```

Check the content of the response

In [10]:
```
print(response.content)
```

```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images
2.imgbox.com/3c/0e/T8iJcSN3_o.png","large":"https://images2.imgbox.com/40/e3/GypSkayF_o.png"},"reddit":{"campaign":null,"launc
h":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/wa
tch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.space.com/2196-spacex-inaugural-falcon-1-rocket-lost-launc
h.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":"2006-03-17T00:00:00.000Z","static_fire_dat
e_unix":1142553600,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":33,"altitud
```

Filter data frame for Falcon 9

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 [28]:
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data[data.BoosterVersion == 'Falcon 9']
data_falcon9
```

Out[28]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPac
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None
7	11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None
8	12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None

Dealing with the Missing Values

In [31]:
```
# Calculate the mean value of PayloadMass column
Mean_PayloadMass = data_falcon9.PayloadMass.mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, Mean_PayloadMass)
```

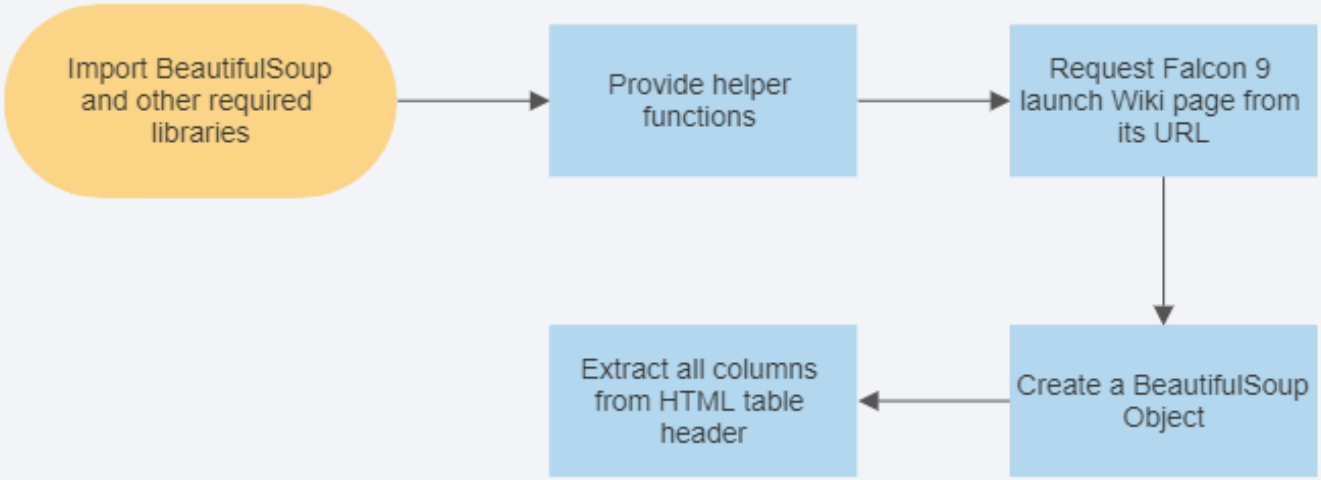
```
/tmp/wsuser/ipykernel_154/1566627384.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, Mean_PayloadMass)
```

In [32]:
```
data_falcon9.isnull().sum()
```

Out[32]:
```
FlightNumber      0
Date              0
```

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts



- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose

Import the Beautifulsoup library

In [2]:
```
import sys

import requests
from bs4 import BeautifulSoup
import re
import unicodedata
import pandas as pd
```

Provide helper functions

In [3]:
```
def date_time(table_cells):
    """
    This function returns the data and time from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    return [data_time.strip() for data_time in list(table_cells.strings)][0:2]

def booster_version(table_cells):
    """
    This function returns the booster version from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=''.join([booster_version for i,booster_version in enumerate( table_cells.strings) if i%2==0][0:-1])
    return out

def landing_status(table_cells):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=[i for i in table_cells.strings][0]
    return out
```

Request the Falcon9 Launch Wiki page from its URL

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 [6]:
```
# use requests.get() method with the provided static_url
response = requests.get(static_url).text
# assign the response to a object
```

Creating BeautifulSoup object

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

In [7]:
```
# 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

In [8]:
```
# Use soup.title attribute
print(soup.title)
```

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

Extract all columns from the HTML table header

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

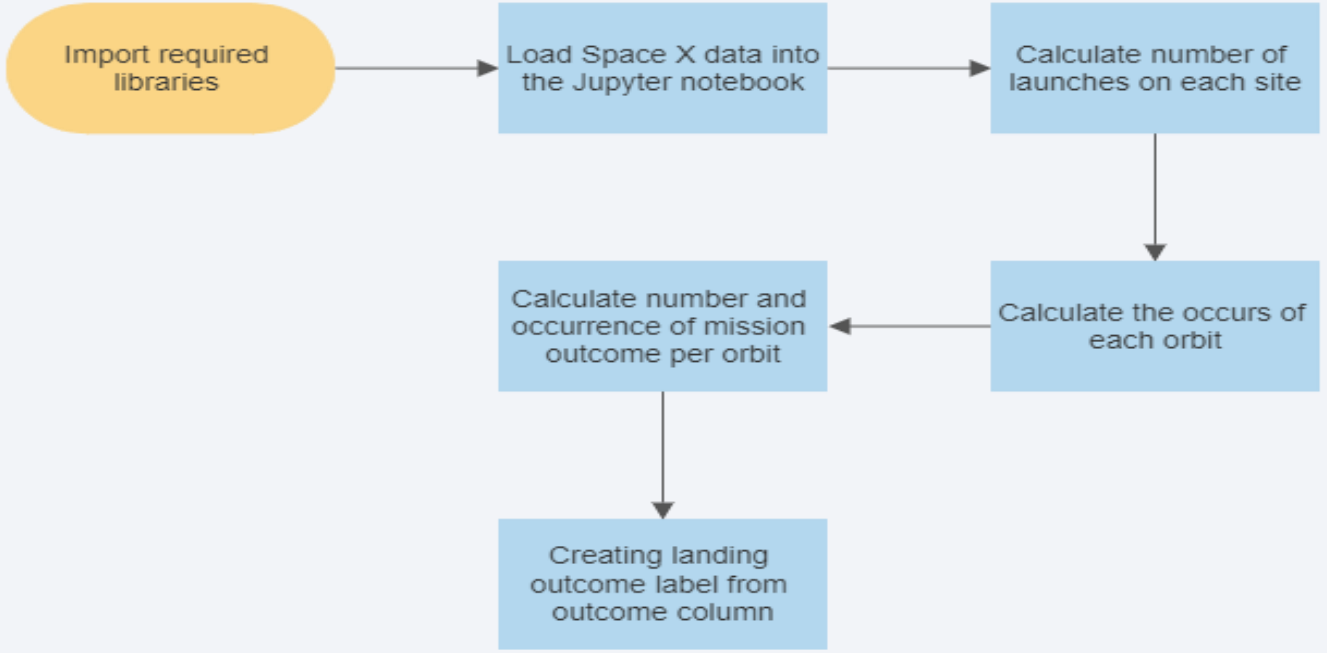
Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

In [10]:
```
# 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)
```

```
[<table class="multicol" role="presentation" style="border-collapse: collapse; padding: 0; border: 0; background:transparent; w
idth:100%;">
<tbody><tr>
<td style="text-align: left; vertical-align: top;">
<h3><span class="mw-headline" id="Rocket_configurations">Rocket configurations</span></h3>
<div class="chart noresize" style="margin-top:1em;max-width:420px;">
<div style="position:relative;min-height:320px;min-width:420px;max-width:420px;">
<div style="float:right;position:relative;min-height:240px;min-width:320px;max-width:320px;border-left:1px black solid;border-b
ottom:1px black solid;">
<div style="position:absolute;left:3px;top:224px;height:15px;min-width:18px;max-width:18px;background-color:LightSteelBlue;-web
kit-print-color-adjust:exact;border:1px solid LightSteelBlue;border-bottom:none;overflow:hidden;" title="[[Falcon 9 v1.0]]: 2">
</div>
<div style="position:absolute;left:55px;top:224px;height:15px;min-width:18px;max-width:18px;background-color:LightSteelBlue;-we
bkit-print-color-adjust:exact;border:1px solid LightSteelBlue;border-bottom:none;overflow:hidden;" title="[[Falcon 9 v1.0]]:
2"></div>
<div style="position:absolute;left:81px;top:232px;height:7px;min-width:18px;max-width:18px;background-color:LightSteelBlue;-web
kit-print-color-adjust:exact;border:1px solid LightSteelBlue;border-bottom:none;overflow:hidden;" title="[[Falcon 9 v1.0]]: 1">
</div>
```

Data Wrangling

The data used for the analyzing was extracted from the web page of Wikipedia. Here the data is cleaned or also can be said as wrangling of data. The data was first been loaded into the Jupyter notebook. Then the number of launches on each site, number and the occurrence of the mission outcome per orbit was calculated. At last we created a landing outcome label from the Outcome column. The flowchart for the above process is presented for better understanding.



Calculate the number of launches on each site

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

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

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

In [6]:
```
# Apply value_counts() on column LaunchSite
df.LaunchSite.value_counts()
```

Out[6]:
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

Calculate the number and occurrence of each orbit

In [7]:
```
# Apply value_counts on Orbit column
df.Orbit.value_counts()
```

Out[7]:
```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
GEO       1
ES-L1     1
SO        1
HEO       1
Name: Orbit, dtype: int64
```

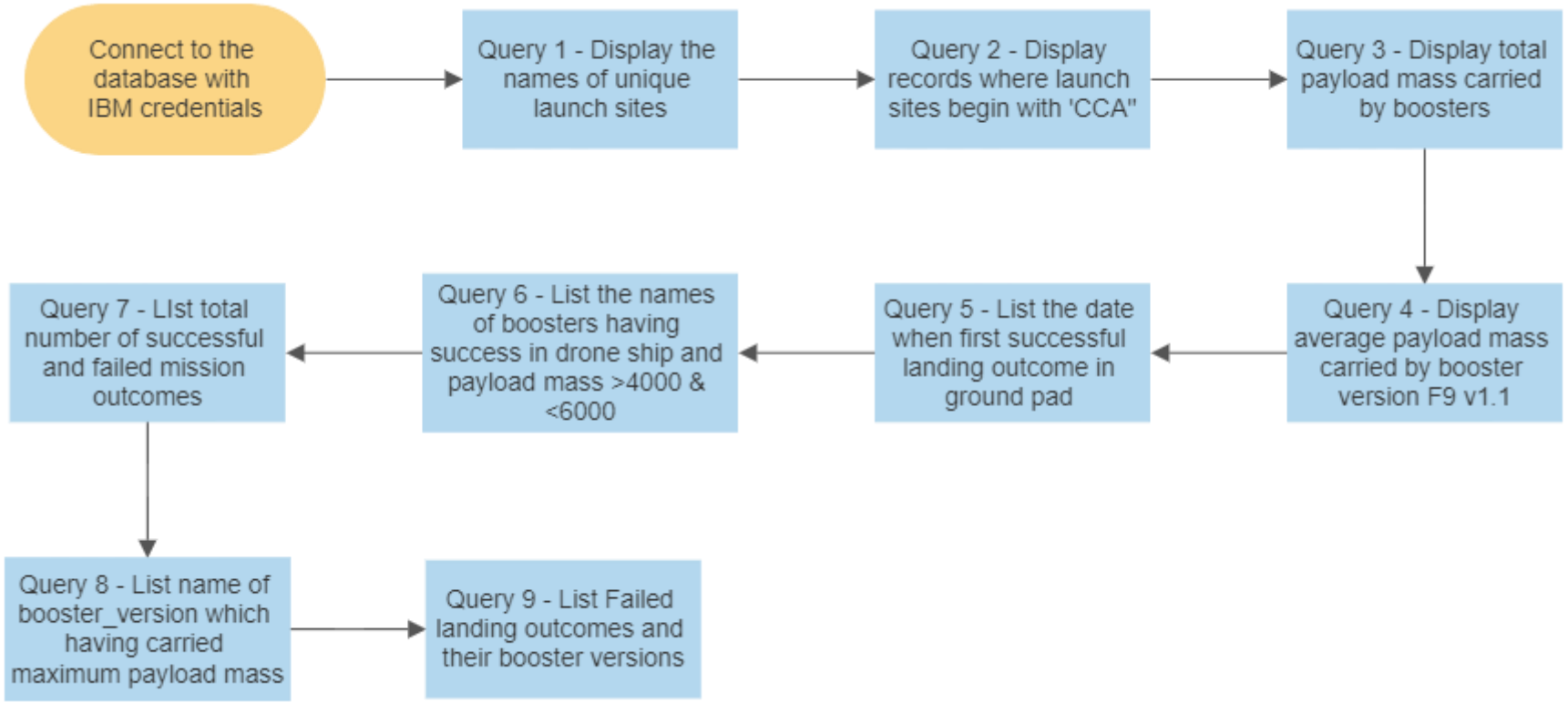
EDA with Data Visualization

- The exploratory data analysis here with the visualization is done with the help of python libraries like matplotlib, seaborn. We have created scatter point chart, bar plot and a line chart.
- Next one is also a scatter point chart with y axis as launch site and x axis as payload mass. After observing this chart you will find that for the VAFB-SLC launch site there are no rockets launched for heavy payload.
- Here I have created a scatter point chart where x axis is Flight number while y axis is launch site. And the hue is set to the class parameter
- I have created a bar plot with an x axis as orbit and y axis as class. This bar plot helps us analyze which orbits have high success rates.
- There another scatter point chart where x axis is payload mass and y axis is orbit which helps us analyze With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- And lastly we have a line chart where x axis is the date and y axis is the success rate which helps us see that the success rate after 2013 kept on increasing.
- The Github link to all of my projects is - https://github.com/Abhi6803/IBM-Data-Science-Professional.git

EDA with SQL

- The SQL queries were used to extract necessary data from the table. The table was stored in the db2 server of IBM and was connected in the Jupyter Notebook with the help of credentials.
- These queries helped us to extract data easy and form another data frame easily.
- The queries consisted like Displaying the names of unique launch sites, displaying the records where launch sites begin with ‘CCA’.
- Display the total payload mass carried by boosters
- Display the average payload mass carried by booster version F9 v1.1.
- Listing the date when the first successful landing outcome in ground pad, boosters having success in drone ship and having payload mass greater than 4000 and lesser than 6000, total number of successful and failed mission outcomes.
- For further info you can check out my Github repository, the link is : https://github.com/Abhi6803/IBM-Data-Science-Professional.git
- The flow chart for the above summary is presented in the next slide.

Flow chart for SQL queries



Build an Interactive Map with Folium

- 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.
- Folium is a library which has built in map functionality. In this you can locate the places all around the globe just with the help of co-ordinates provided. Its an interactive visualization.
- Here we located all the space shuttle launch sites in assessment with NASA.
- We also located the nearby locations around those launch sites like highway, city and the beach closest to the launch site of Kennedy Space center.
- For more information regarding this you can visit my Github repository, the link is : https://github.com/Abhi6803/IBM-Data-Science-Professional.git

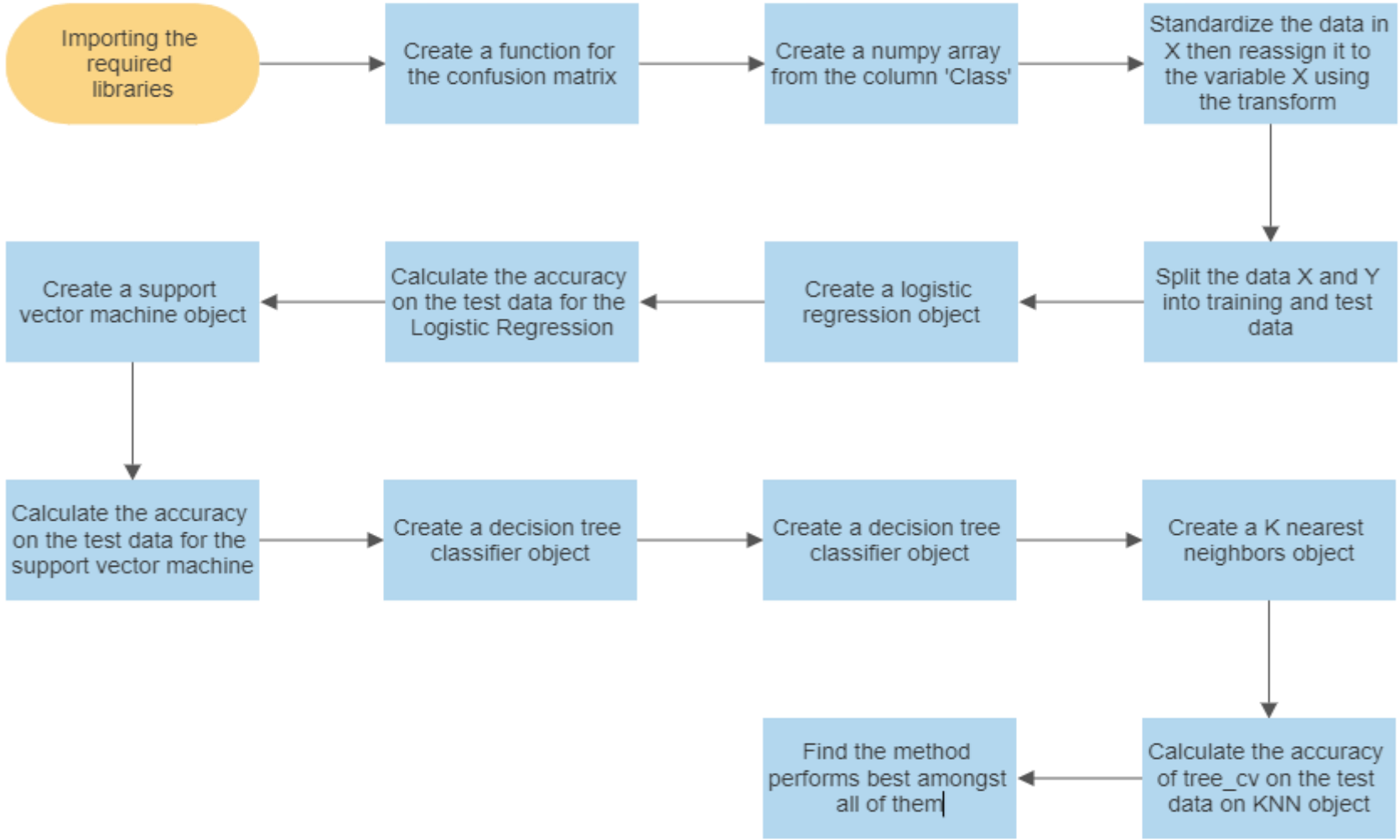
Build a Dashboard with Plotly Dash

- Plotly Dash is a very useful framework, it enables us to create a web application with an interaction support.
- We use the dash library here to create a interactive visualization where we have created a pie chart and a scatter plot.
- We have a dropdown where the name of the launch sites are entered. There is a functionality created where the site which is being selected from the dropdown the pie chart and the scatter plot will be adjusted accordingly.
- The Pie chart show the number of launches from each site and the same goes for the scatter where we get the info about the Class and Payload mass of the booster. Booster is used as a index.
- The slider also included where it defines the payload range.
- The range is starting from 0 till 10000.
- The link to the project : https://github.com/Abhi6803/IBM-Data-Science-Professional.git

Predictive Analysis (Classification)

- The Predictive analysis for the Space X project consist of different algorithms and techniques. This predictive analysis is very important because it is what makes the Falcon rocket so much cost effective.
- The cost effectiveness is completely dependent on the first stage of the rocket because it is reused in the next launch. So the main motive here is to predict whether the first stage will land or not.
- This is done with help of data webscraped from the Wikipedia consisting parameters like landing_outcomes, mission accomplishment record, booster version, class, Payload mass, grid fans and the co-ordinates for the launch sites.
- We have predicted with the help of following techniques : Logistic Regression, K nearest neighbors, Decision Tree, Support Vector.
- And after obtaining the values from these we picked which ever had the best accuracy score.
- The better understanding for this procedure can be picked after seeing the flow chart I created on the next slide.
- You can also visit my Github repository and have access to all of these project files. The link is : https://github.com/Abhi6803/IBM-Data-Science-Professional.git

Machine learning for the prediction of Falcon landings



Results

Results for the EDA

- The exploratory data analysis is a very crucial process which helped us to remove all the unnecessary data and make them visualize for better understanding of the results.
- In this exploratory data analysis we have imported data into the pandas data frame.
- After the import we have cleaned data in various techniques like firstly removing the null values. Then extracting the necessary data and creating the columns we need for further procedure.
- After cleaning the data then the time was to visualize it for the better understanding the data where with a glance we can understand the whole composition of the data. Here we created scatter plots, bar plots and line chart as per the requirements.
- All of these procedure is extremely important because after this step we will be creating the web applications and the geometrically locating the space launch centers.

Results

Results on interactive analytics

- Here I created a geographical visualization where we have located the launch sites as per the data.
- This locations were marked with the help of folium library. This library has features to mark any location around the globe in various formats. For locating the launch sites as well as the nearby premises like highway, beach and city we needed the co-ordinates to those sites which were already being provided in the data.
- Next I created a jupyter notebook consisting different tasks for visualizations. I have made different types of plots and charts in it like line chart, bar chart and scatter point chart.
- All of these plots were made with the help of libraries like matplotlib and seaborn.
- Another one which we created was interactive visualization which was web based application. It consisted of dropdown, slider and a scatter point chart.
- The dropdown menus included the names of launch sites and the slider was for indicating the payload where as the scatter plot was for the indicating the class and the payload mass of the boosters.

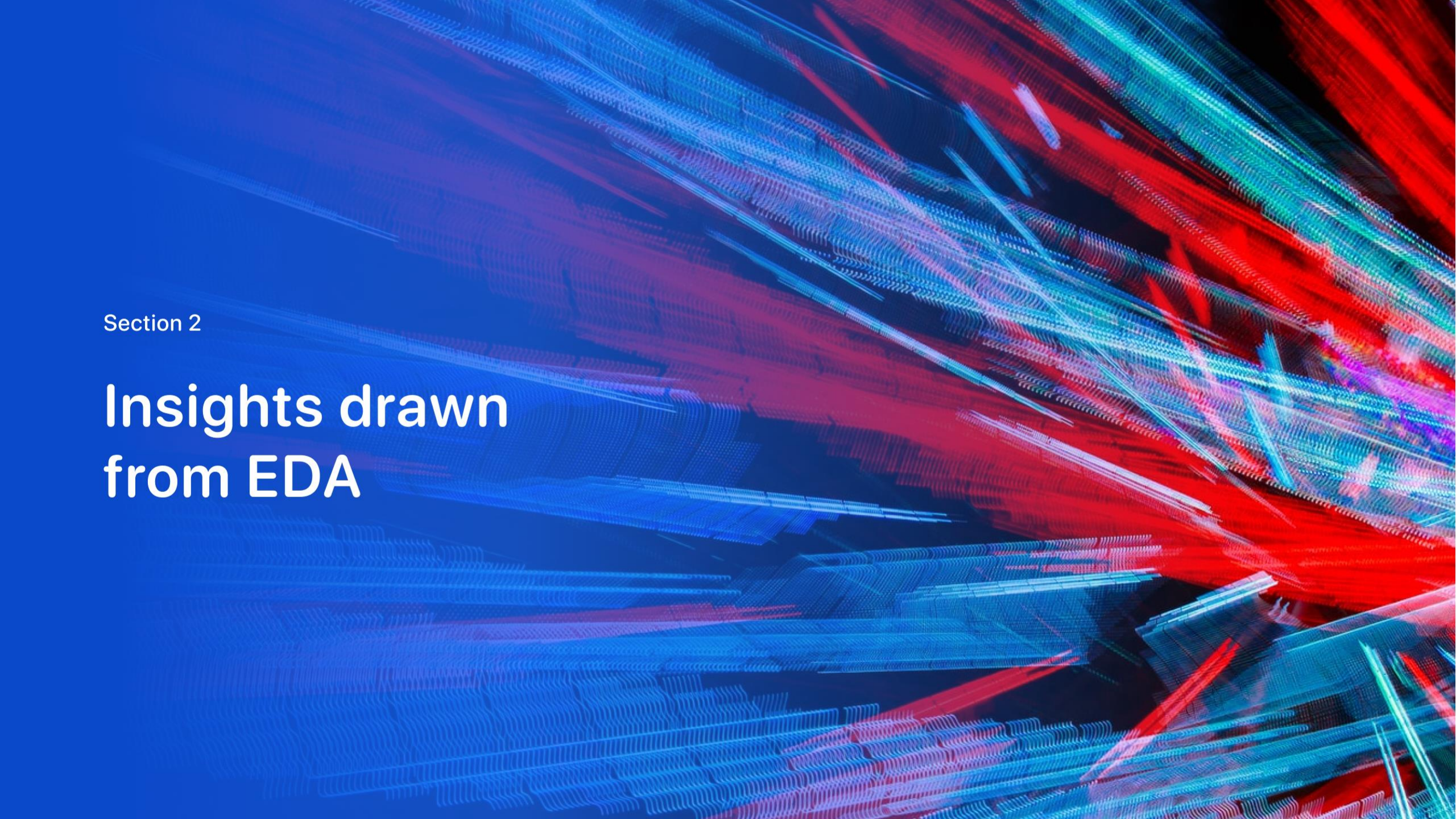
Results

Results on predictive analytics result

- After all the visualization and the analysis of the data, it was the next crucial step of predicting whether the boosters will land or not. Here I have split the data into testing set and the training set.
- The techniques used for prediction are K nearest neighbor, logistic regression, support vector and decision tree.
- The obtained accuracy score for them are 0.848, 0.83, 0.83 and 0.77 respectively.
- All the provided columns/ features were used for this procedure.
- We used the scikit learn library which had the built modules for each of these techniques.
- Data was split into test_size of 0.2 with the random state of 2

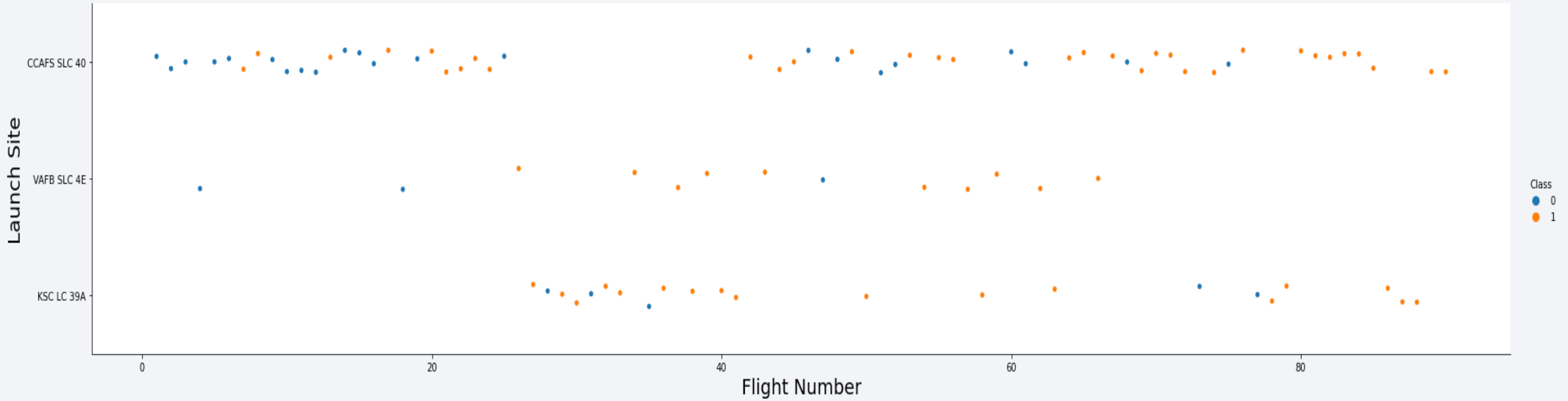
Section 2

Insights drawn from EDA



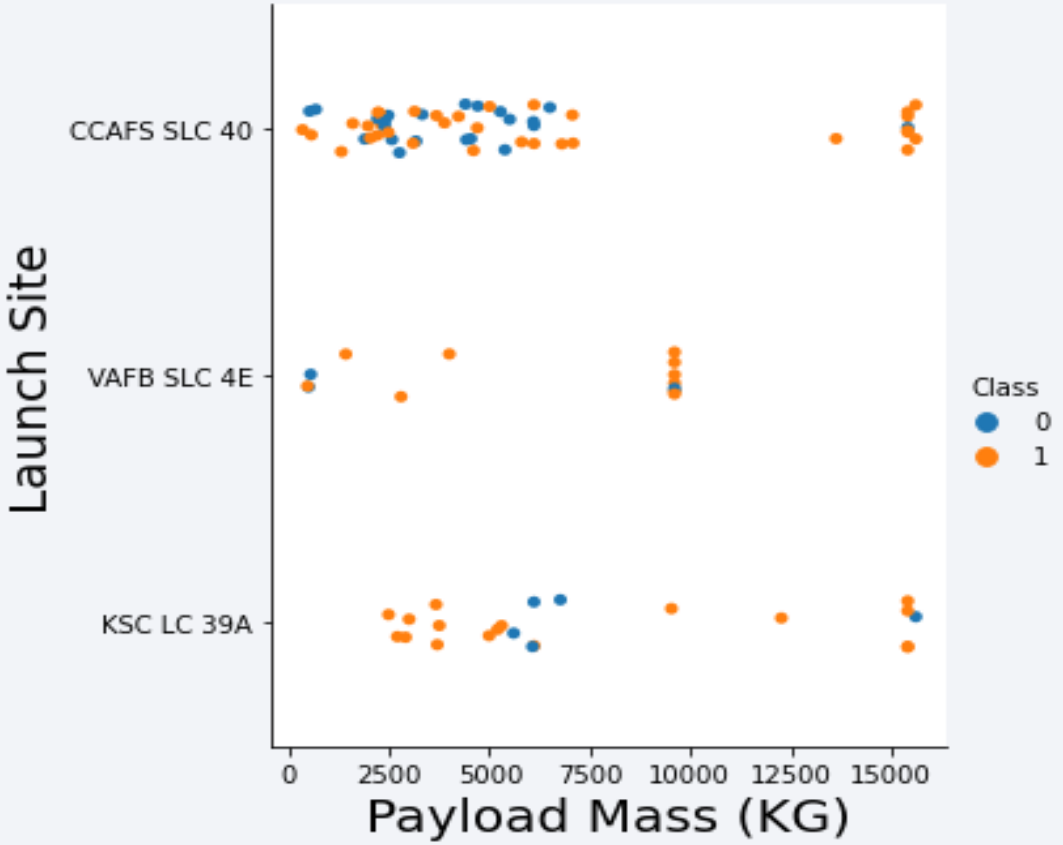
Flight Number vs. Launch Site

- Here in this we can see that CCAFS LC-40 has the most number of launches while VAFB SLC 4E has the least number of launches.



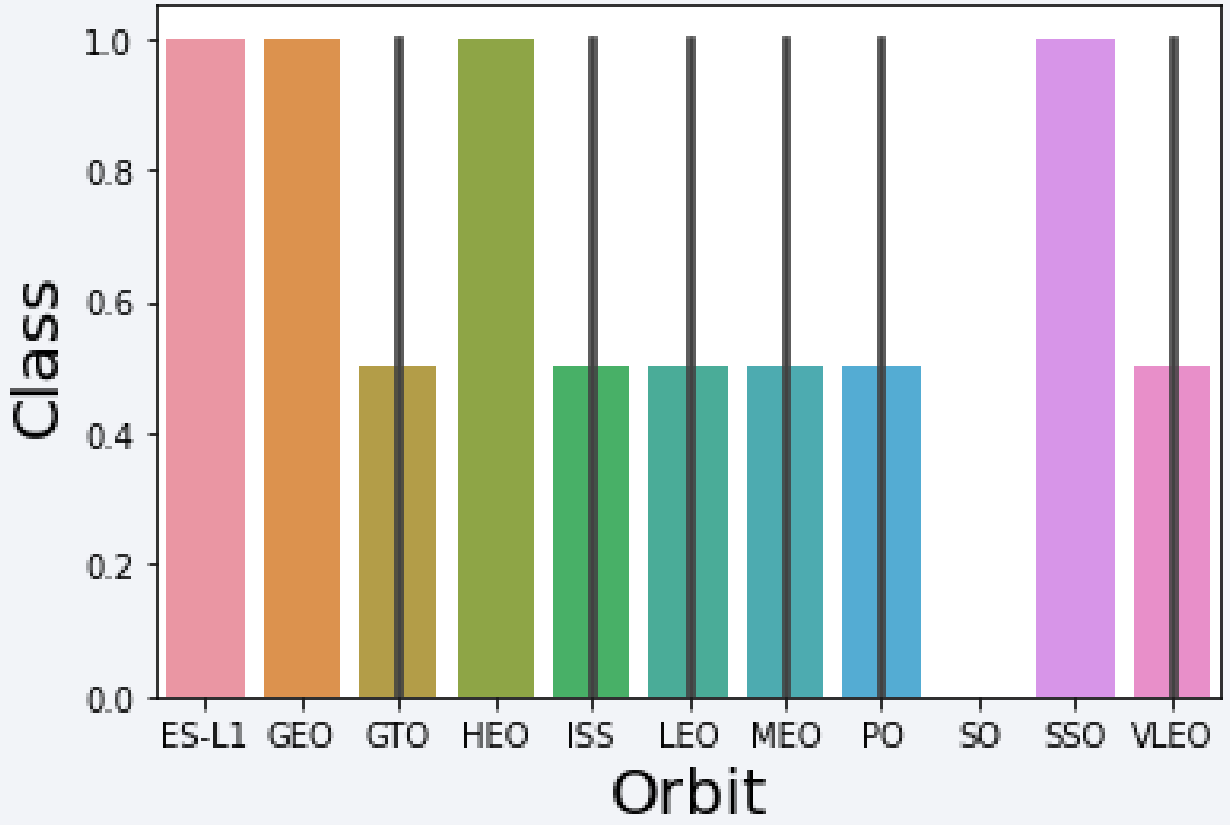
Payload vs. Launch Site

- The VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000)



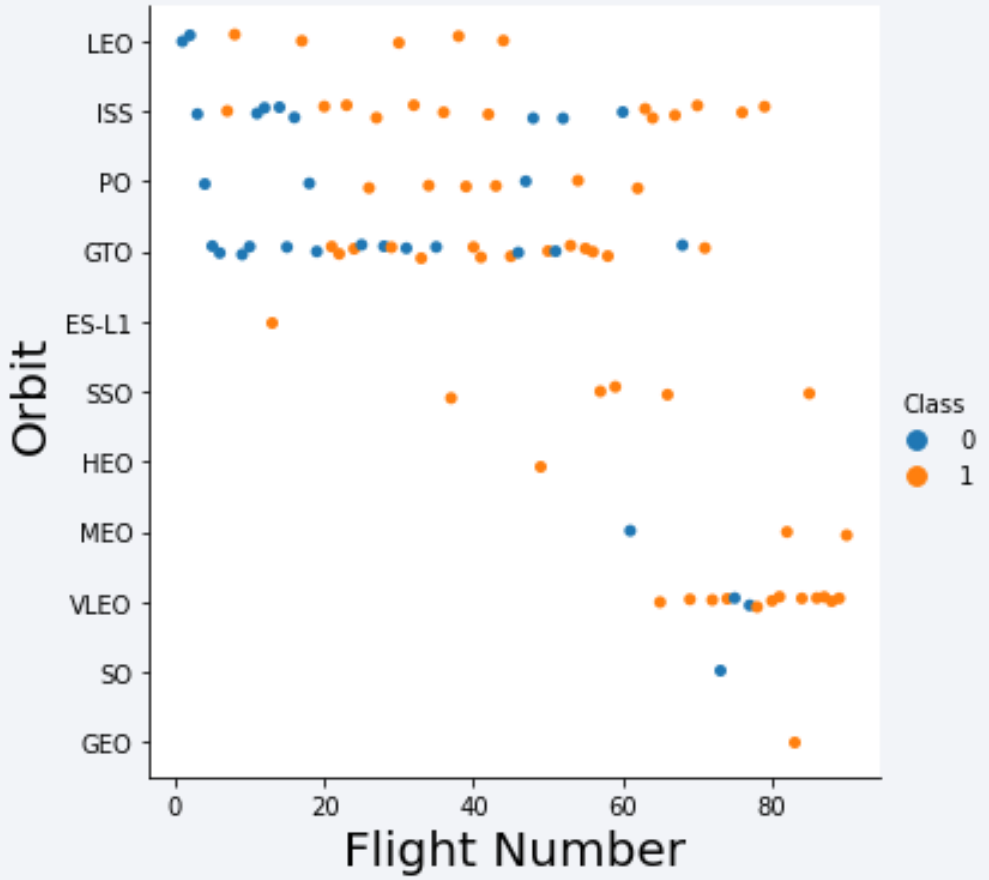
Success Rate vs. Orbit Type

- The ES-L1, GEO, HEO, SSO are the orbit having the highest number of success rate.



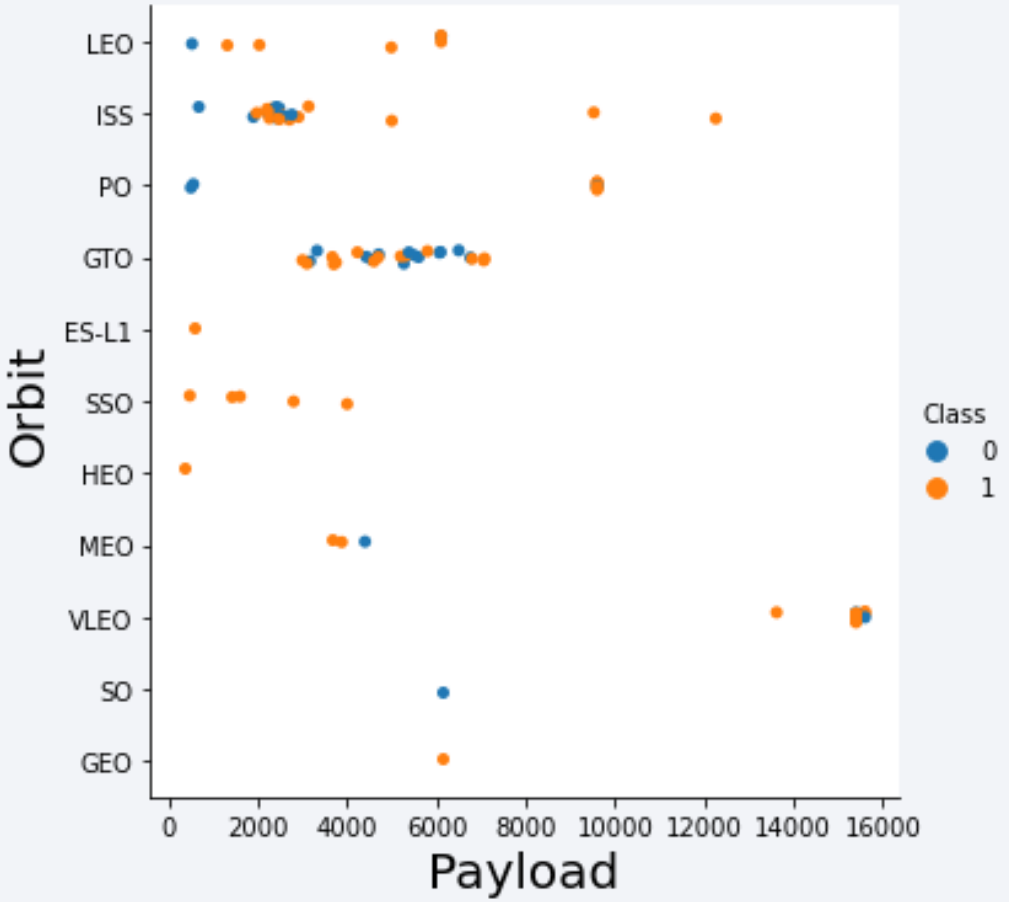
Flight Number vs. Orbit Type

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



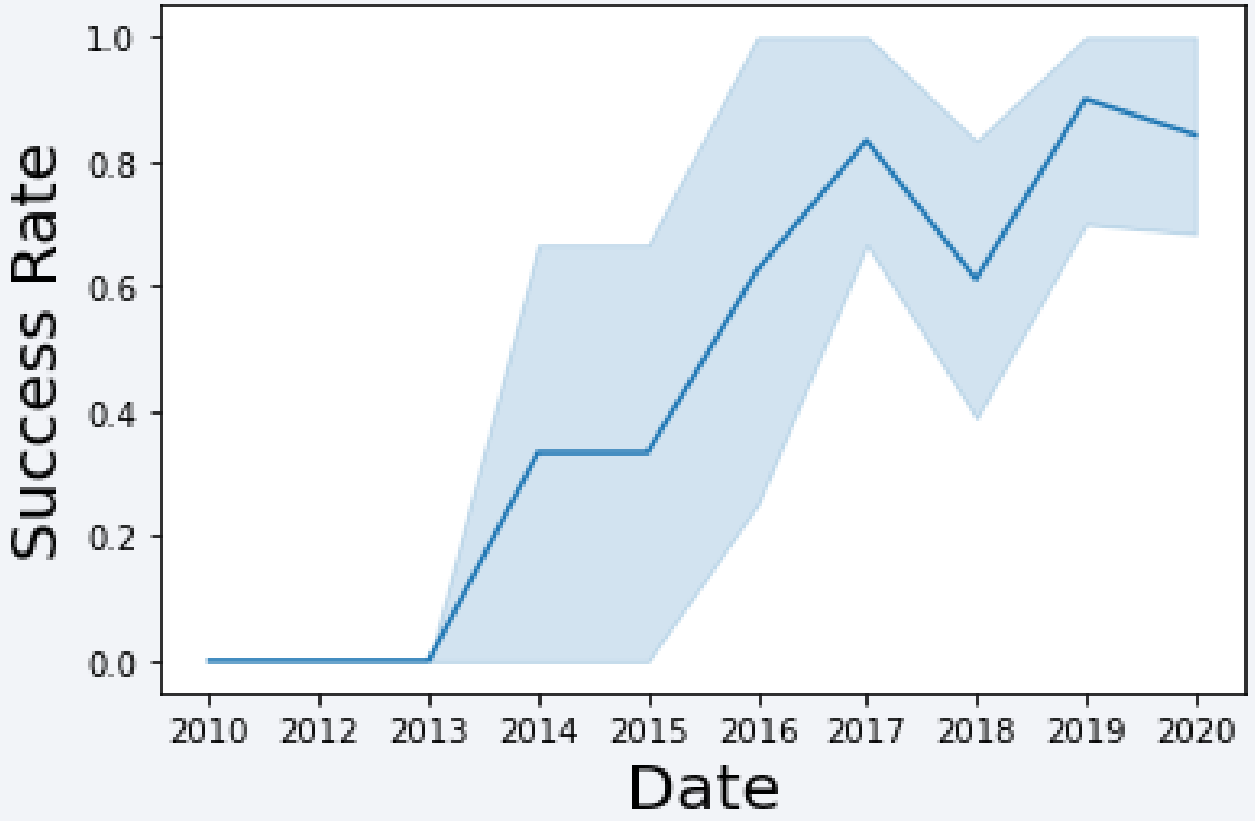
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.



Launch Success Yearly Trend

- the success rate since 2013 kept increasing till 2020
- But it had significant drip in the year 2018 but increased in the year 2019.



All Launch Site Names

- The distinct functions gives the output of distinctive name of the launch sites from the SPACEXTBL table from DB2 server

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

In [9]:
```
%%sql
SELECT DISTINCT Launch_Site FROM SPACEXTBL
```

```
 * ibm_db_sa://gwp83471:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

Out[9]:

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

Launch Site Names Begin with 'CCA'

- The following is the query for obtaining the launch site name which contaim ‘CCA’ and we used a WHERE function for differentiating and ORDER BY for sorting the list in descending order with DESC function

In [15]:
```
%%sql
SELECT LAUNCH_SITE FROM SPACEXTBL
WHERE LAUNCH_SITE LIKE 'CCA%'
ORDER BY LAUNCH_SITE DESC
```

```
 * ibm_db_sa://gwp83471:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

Out[15]:

launch_site
CCAFS SLC-40
CCAFS SLC-40
CCAFS SLC-40
CCAFS SLC-40
CCAFS SLC-40
CCAFS SLC-40

Total Payload Mass

- The query is for getting the total payload mass for the boosters with the help of SUM function applied to PAYLOAD_MASS__KG_ column.

In [18]:
```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS
FROM SPACEXTBL
WHERE CUSTOMER = 'NASA (CRS)'
```

```
 * ibm_db_sa://gwp83471:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

Out[18]:

total_payload_mass
45596

Average Payload Mass by F9 v1.1

- Average payload mass is displayed with the AVERAGE function and the WHERE function for particularly obtaining the result for F9 v1.1

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

In [21]:
```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) AS AVERAGE_PAYLOAD_MASS
FROM SPACEXTBL
WHERE BOOSTER_VERSION = 'F9 v1.1'
```

```
 * ibm_db_sa://gwp83471:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

Out[21]:

average_payload_mass
2928

First Successful Ground Landing Date

- SELECT function for the selection of element which we want to obtain and the ORDER BY for sorting the values and LIMIT function for limiting the results to 1.

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

Hint:Use min function

In [48]:
```
%%sql
SELECT Date, LANDING__OUTCOME
FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Success (drone ship)'
ORDER BY Date ASC
LIMIT 1
```

```
 * ibm_db_sa://gwp83471:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

Out[48]:

DATE	landing__outcome
2016-04-08	Success (drone ship)

Successful Drone Ship Landing with Payload between 4000 and 6000

- The landing outcome is supposed to be produced where payload is between 4000 and 6000 with the help of AND function used next to the WHERE function.

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

In [51]:
```
%%sql
SELECT BOOSTER_VERSION, LANDING__OUTCOME
FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Success (ground pad)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ <6000
```

```
 * ibm_db_sa://gwp83471:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

Out[51]:

booster_version	landing__outcome
F9 FT B1032.1	Success (ground pad)
F9 B4 B1040.1	Success (ground pad)
F9 B4 B1043.1	Success (ground pad)

Total Number of Successful and Failure Mission Outcomes

- From the result we can see that 2 were failed while 99 were accomplished. We used the GROUP BY function on the MISSION_OUTCOME table.

List the total number of successful and failure mission outcomes

In [112]:
```
%%sql
SELECT count(MISSION_OUTCOME) as missionoutcomes
from SPACEXTBL
GROUP BY MISSION_OUTCOME
```

```
 * ibm_db_sa://gwp83471:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

Out[112]:

missionoutcomes
1
99
1

Boosters Carried Maximum Payload

- For getting the list of booster version which have carried the maximum number of payload I used the Sub-query feature for the PAYLOAD_MASS__KG_ table with the MAX function.

In [100]:
```
%%sql
SELECT BOOSTER_VERSION AS Booster_Version
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

```
 * ibm_db_sa://gwp83471:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

Out[100]:

booster_version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4

2015 Launch Records

- The list for the failed missions where drone ship was used for landing in the year 2015. I used a LIKE function as well as the AND for mentioning the year column too.

In [102]:
```
%%sql
SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE, YEAR(DATE)
FROM SPACEXTBL
WHERE LANDING__OUTCOME LIKE '%Failure%' AND YEAR(DATE) = '2015'
```

```
 * ibm_db_sa://gwp83471:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

Out[102]:

landing__outcome	booster_version	launch_site	4
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015

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

- With help of BETWEEN, AND, LIKE functions we are able to obtain the following result from the queries.

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

In [104]:
```
%%sql
SELECT DATE, LANDING__OUTCOME
FROM SPACEXTBL
WHERE (DATE BETWEEN '2010-06-04' AND '2017-03-20') AND (LANDING__OUTCOME LIKE 'Failure (drone ship)' OR LANDING__OUTCOME LIKE 'S
uccess (ground pad)')
```

```
 * ibm_db_sa://gwp83471:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

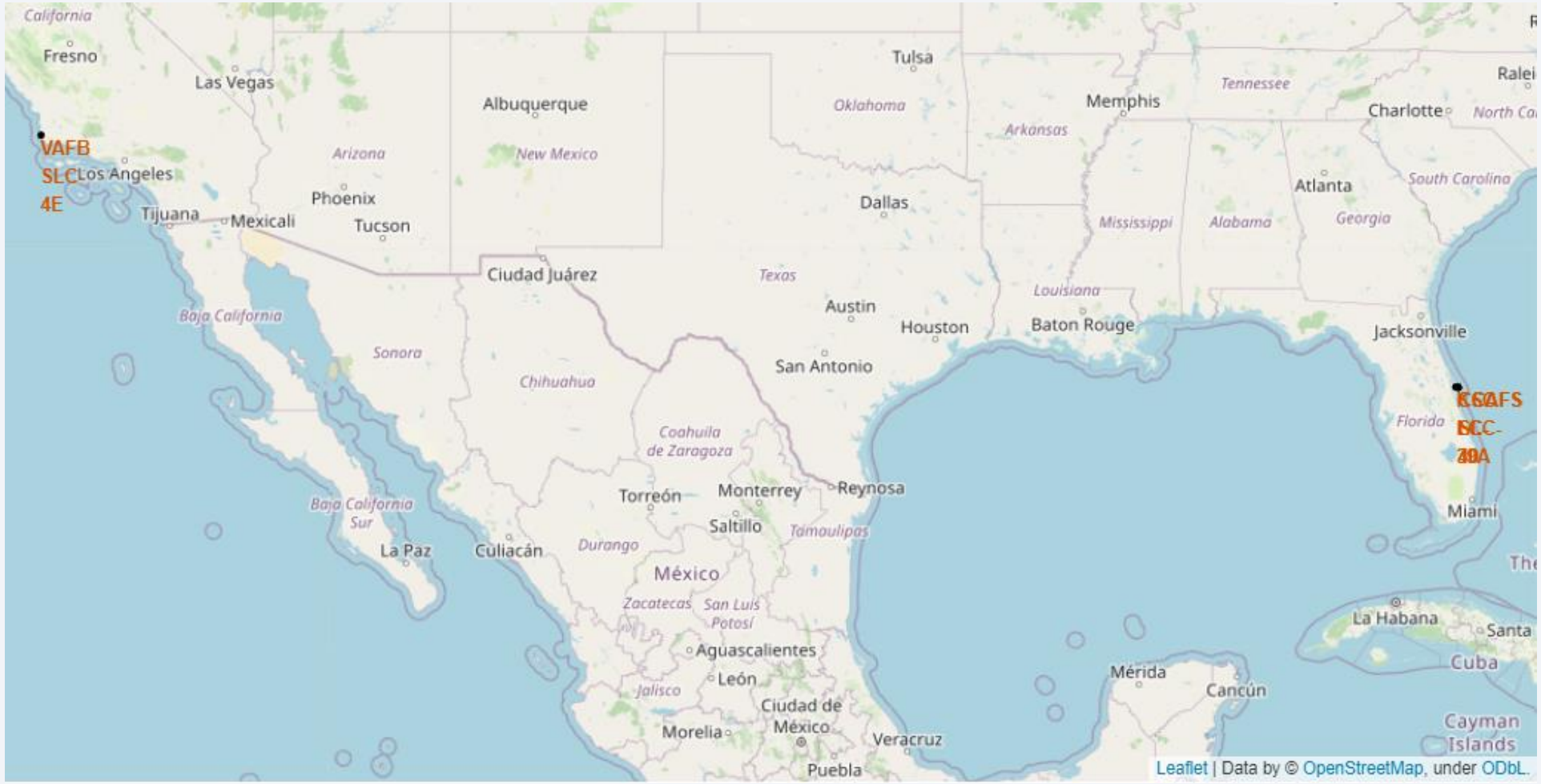
Out[104]:

DATE	landing__outcome
2015-01-10	Failure (drone ship)
2015-04-14	Failure (drone ship)
2015-12-22	Success (ground pad)

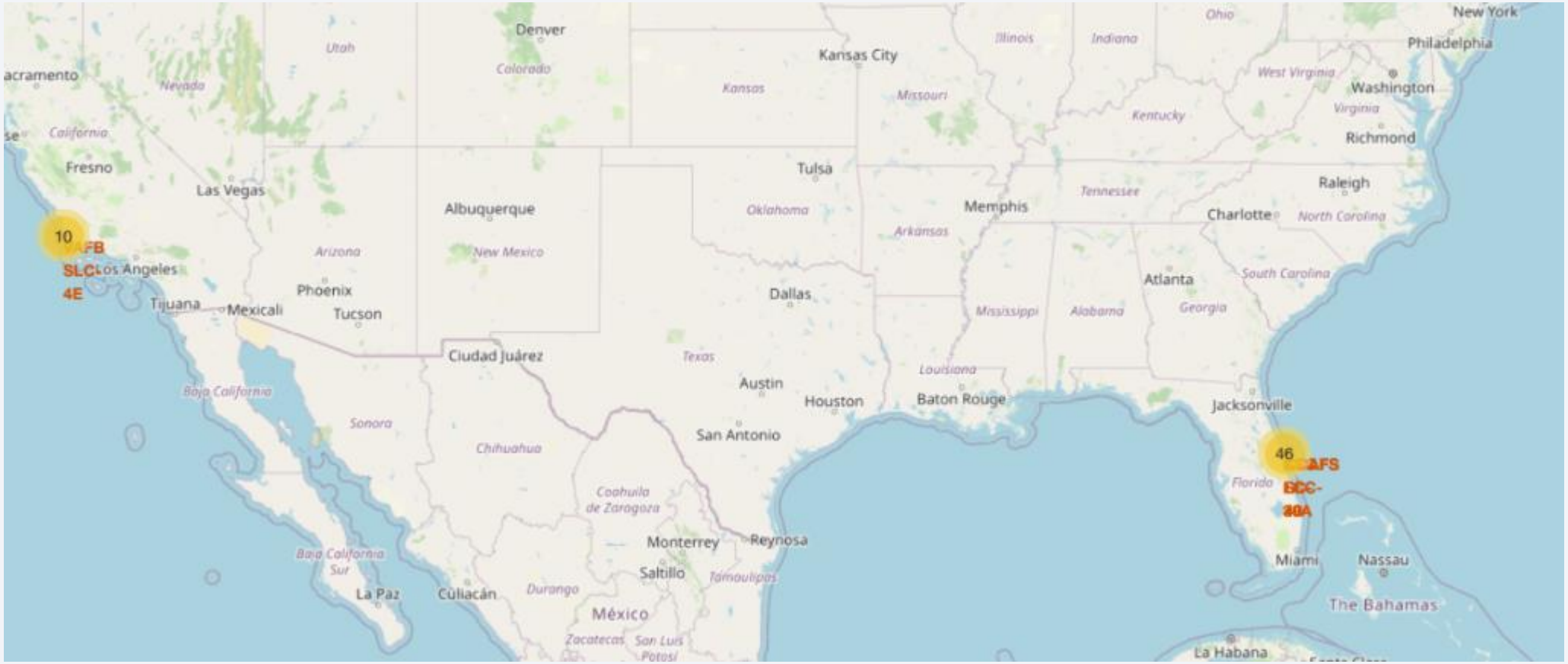
Section 4

Launch Sites Proximities Analysis

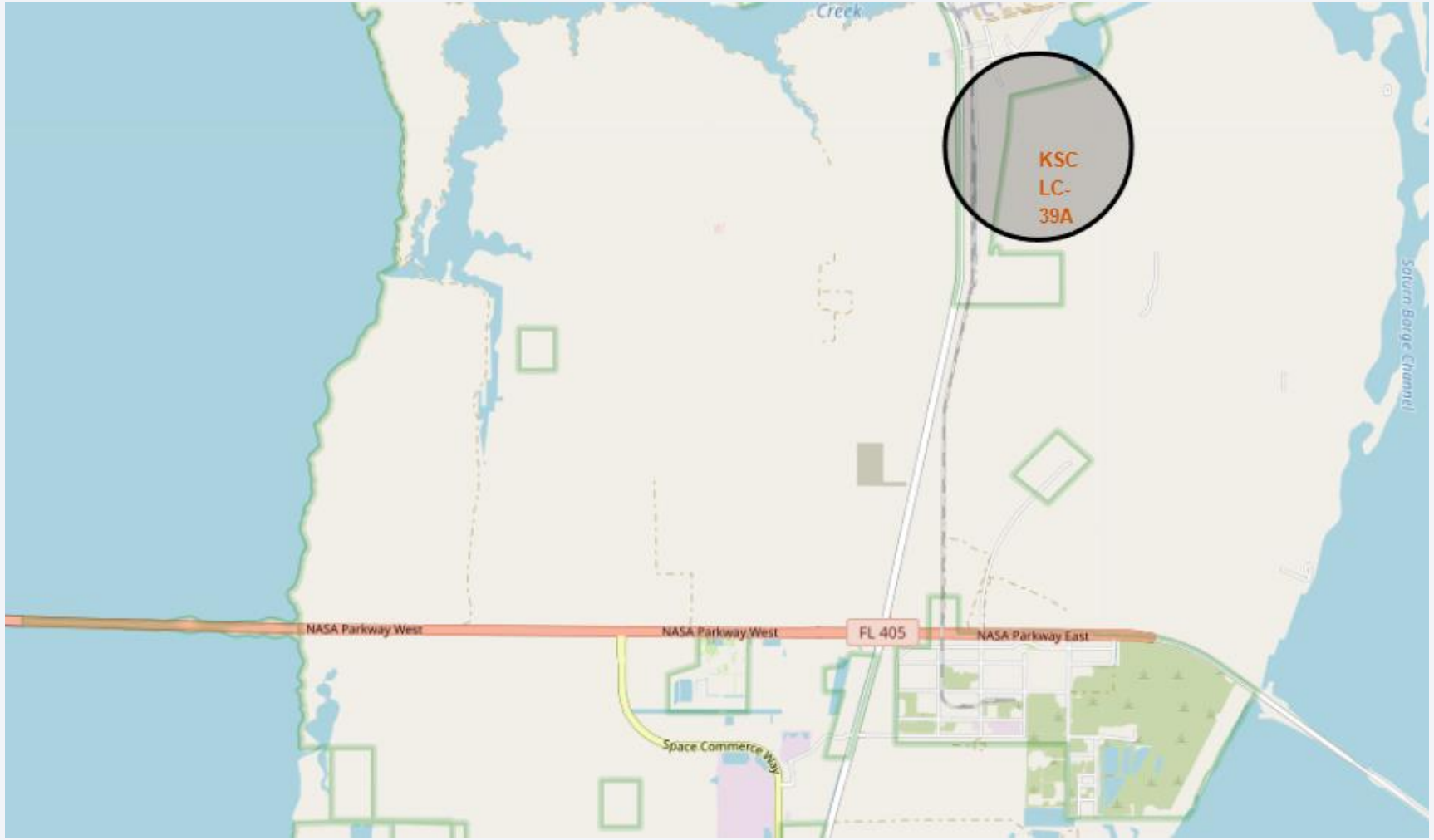
Location of all Launch Sites



Colored labels launch outcomes on map

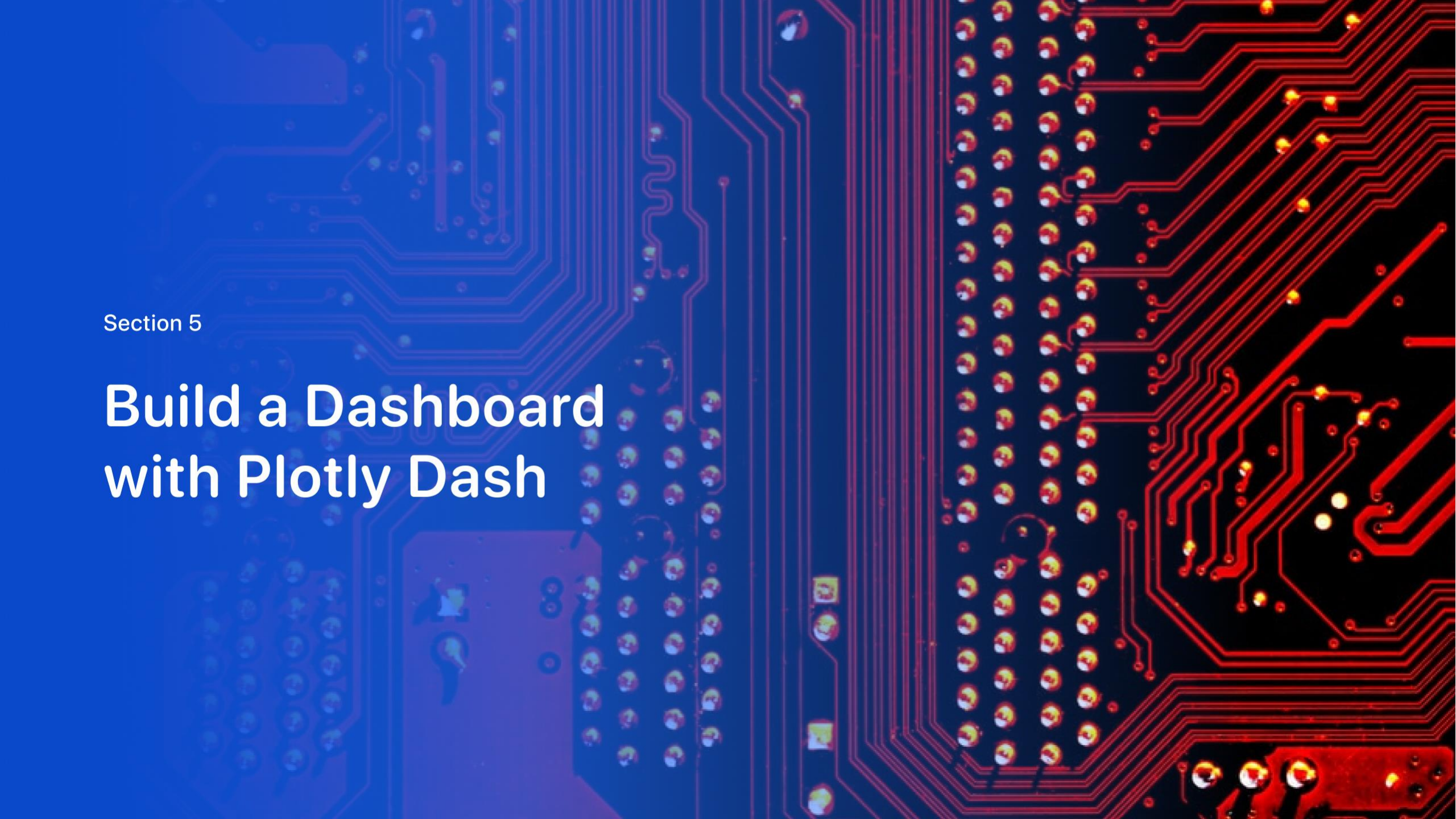


Selected launch site to its proximities

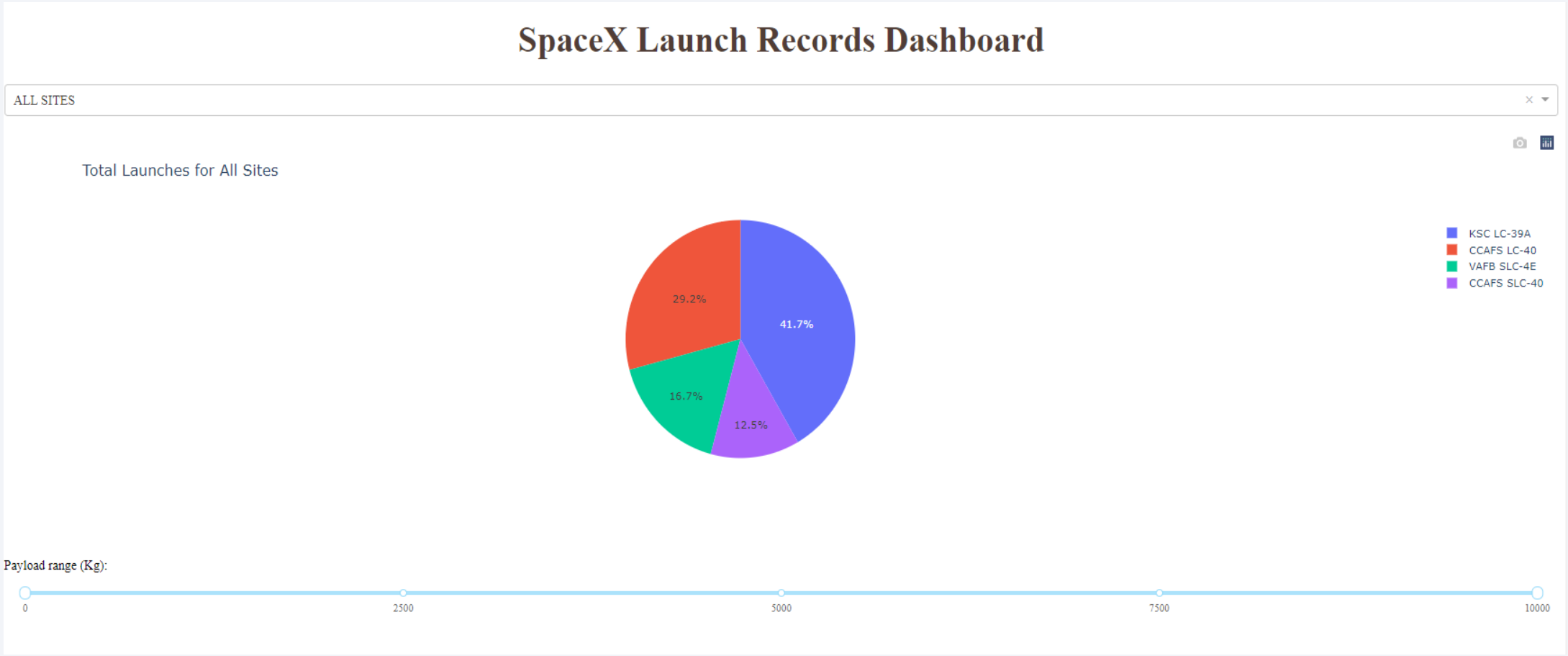


Section 5

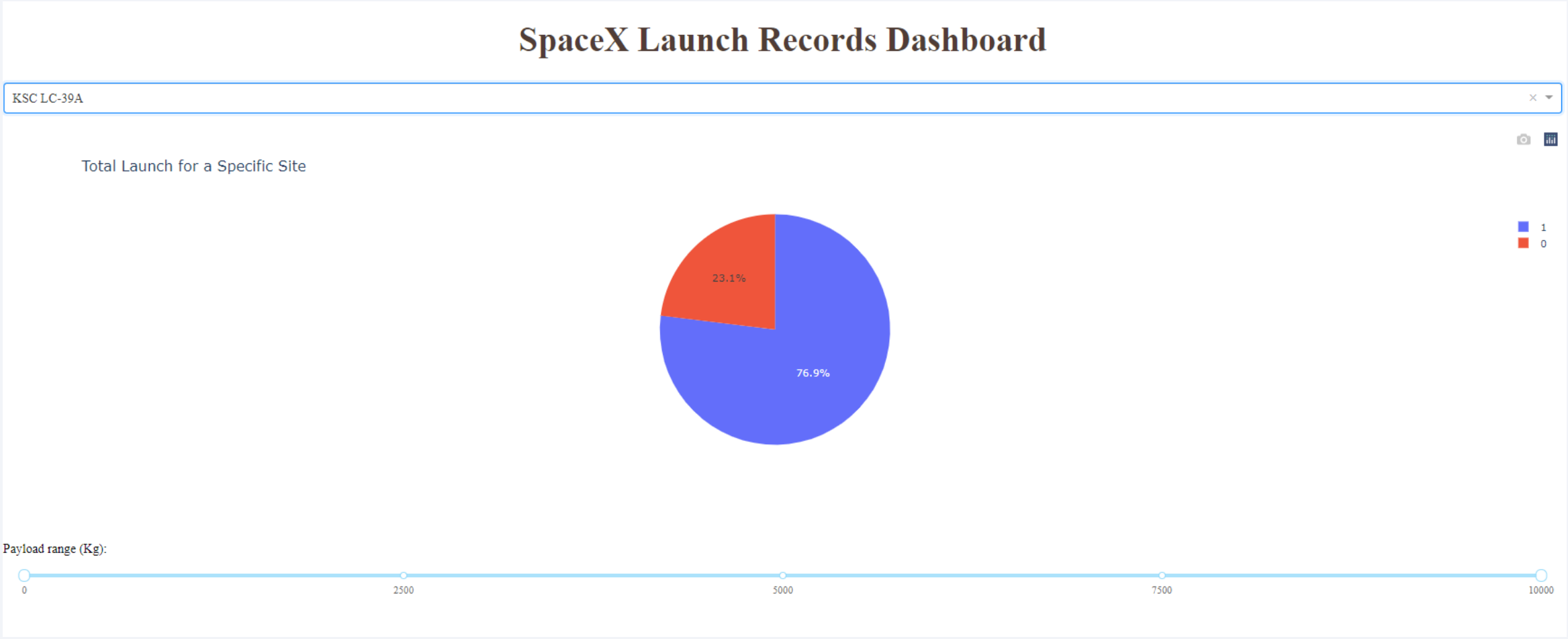
Build a Dashboard with Plotly Dash



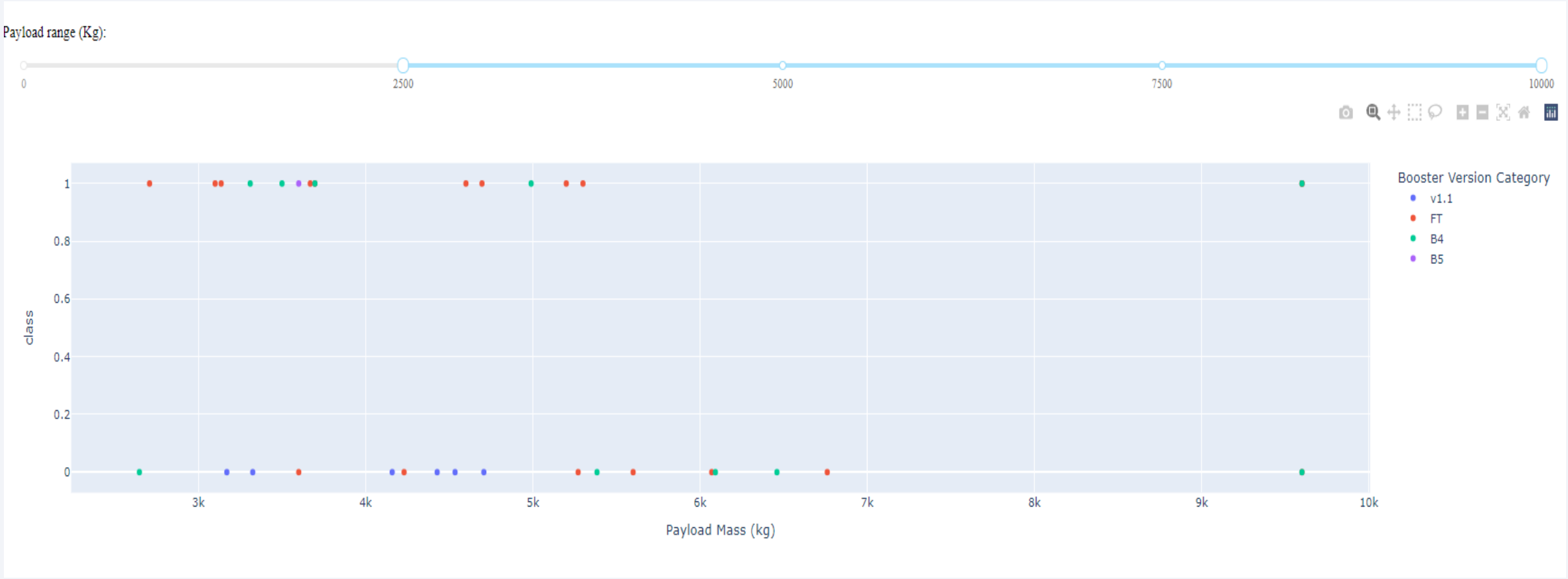
Launch Success for all sites in Piechart



Launch site with highest launch success ratio



Payload vs Launch Outcome with different Payload

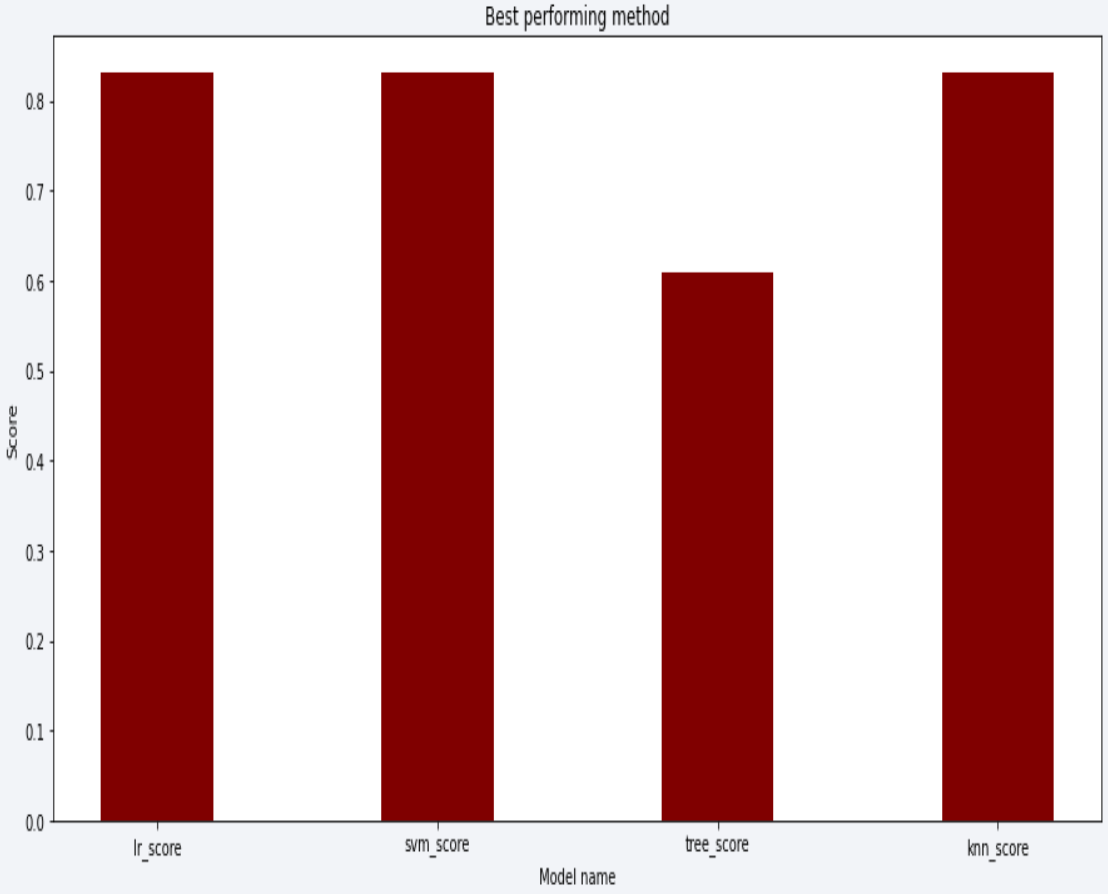


Section 6

Predictive Analysis (Classification)

Classification Accuracy

- The Logistic regression, support vector and the K nearest neighbor perform the best with a score of 0.83
- And the decision tree scores the least with 0.61 score.



Confusion Matrix

- Here the Logistic regression is the best performing method here with a cross-validation score of 0.833.

Conclusions

- Firstly we had collected the data from the Wikipedia page with help of Beautifulsoup library and converted it into the pandas data frame.
- We cleaned the data from null values and extracted the necessary data we required for prediction procedure.
- In the visualization segment we visualized different parameters with respect to each other with the help of line chart, scatter point chart, bar chart, web interactive application and the geometrically locating the launch sites on the map.
- At last for prediction we performed various methods logistic regression, K nearest neighbor, Decision tree and support vector machine. The logistic regression, K nearest neighbor and the Support vector machine turned out to be most accurate.

Thank you!