IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Samhit Eppanapally
5/26/24

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection
  - Data wrangling
  - Exploratory Data Analysis with Data Visualization
  - Exploratory Data Analysis with SQL
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive analysis(Classification)

- Summary of all results
  - Exploratory Data Analysis results
  - Interactive maps and dashboard
  - Predictive Results

Introduction

- Project background and context

- 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. This goal of the project is to produce a machine learning pipeline to predict if the First stage will land successfully.

- Problems you want to find answers

- What factors will determine if the rocket landing was successful or not?

- Does the rate of successful landings increase across the years?

Section 1

Methodology

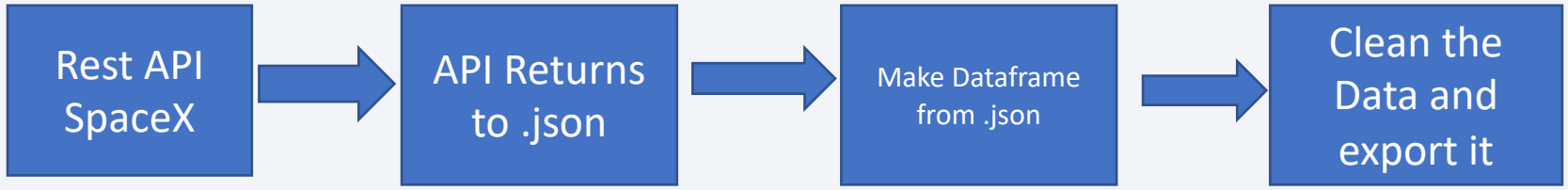
Methodology

Executive Summary

- Data collection methodology:
  - Data was collected by two sources. The two sources are SpaceX API and Web Scraping.
- Perform data wrangling
  - Data will be cleansed, transformed, and then prepared.
- 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

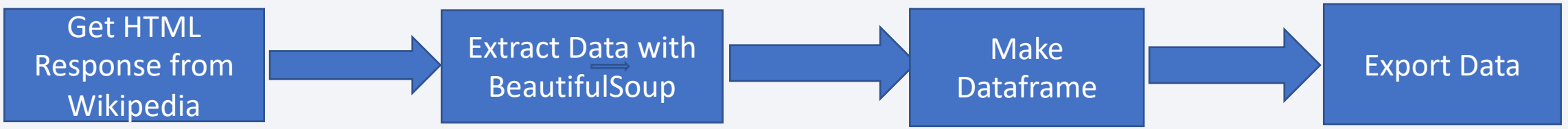
Data Collection

- Datasets are collected from the SpaceX API and web scraping
  - the data that is acquired by the API are rocket, launches, and payload information
- You need to present your data collection process use key phrases and flowcharts

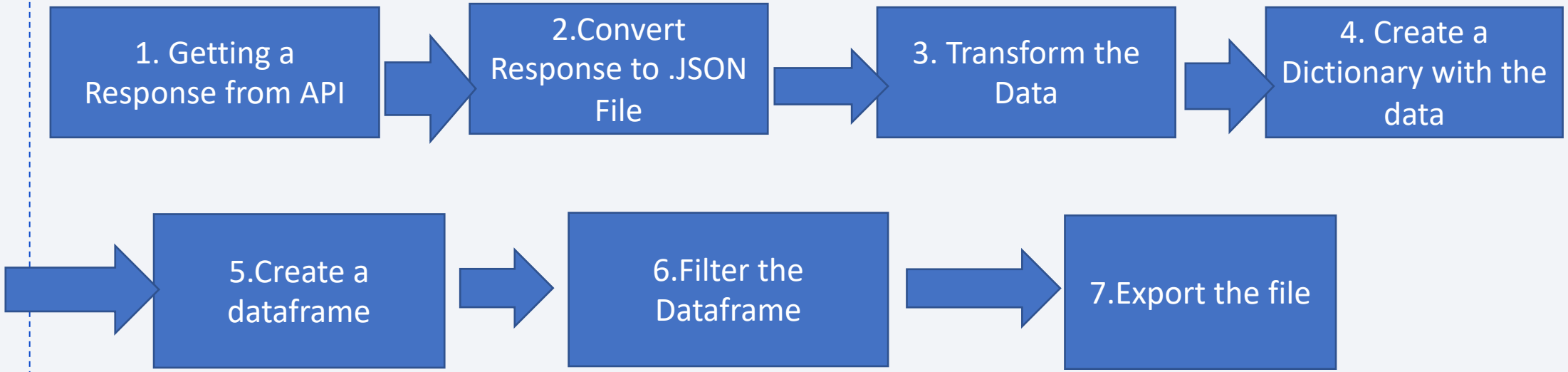


Data Collection Cont.

- The data that is used for web scraping from Wikipedia includes data on launches, landings, and payload information.



Data Collection – SpaceX API



https://github.com/seppanapally167/Applied-Capstone-Final-Project-2/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection – SpaceX API Cont.

1. Getting a Response from API

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

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

2. Convert Response to .JSON File

```
[38]: # Use json_normalize meethod to convert the json result into a dataframe
      data = pd.json_normalize(response.json())
```

3. Transform the Data

```
[72]: getLaunchSite(data)
      getPayloadData(data)
      getCoreData(data)
      getBoosterVersion(data)
```

Data Collection – SpaceX API Cont.

4. Create a Dictionary with the data

```
[73]: launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

5. Create a dataframe

```
[74]: # Create a data from launch_dict
data = pd.DataFrame({key:pd.Series(value) for key, value in launch_dict.items()})
```

Data Collection – SpaceX API Cont.

6. Filter the Dataframe

```
[76]: # Hint data['BoosterVersion']!='Falcon 1'
      data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
```

7. Export the file

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection – Scraping

1. Get a response from HTML

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.

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

2. Create a BeautifulSoup Object

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

```
[144]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
       soup = BeautifulSoup(html_data,"html.parser")
```

https://github.com/seppanapally167/Applied-Capstone-Final-Project-2/blob/main/jupyter-labs-webscraping.ipynb

Data Collection – Scraping Cont.

3. Find all the tables

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.

[143]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url).text
```

4. Get Column Names

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

[148]:
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
tc = first_launch_table.find_all('th')
for th in tc:
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
            column_names.append(name)
```

Data Collection – Scraping Cont.

5. Create a dictionary

```
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']=[]
```

6. Add the data to the keys

```
[151]: extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # 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')
        #if it is number save cells in a dictonary
```

Data Collection – Scraping Cont.

7. Create a dataframe from dictionary

```
df=pd.DataFrame(launch_dict)
```

8. Export to File

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

- In the data set, there are several different cases where the booster did not land successfully.
- Mission was successful:True Ocean, True RTLS, True ASDS
- Mission was not successful: False Ocrean, False RTLS, False ASDS

- The main objective of this lab is to convert these outcomes into training labels with 1 being booster landed successfully, and 0 means the booster did not land successfully.

Data Wrangling Cont.

1. Calculate the LaunchSite

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

```
[50]: # Apply value_counts() on column LaunchSite
      df['LaunchSite'].value_counts()
```

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

2. Calculate the number and occurrence of each orbit

```
[51]: # Apply value_counts on Orbit column
      df['Orbit'].value_counts()
```

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

Data Wrangling Cont.

3. Calculate the number and occurrence of mission outcomes of the orbits

```
[52]: # landing_outcomes = values on Outcome column
      landing_outcomes = df['Outcome'].value_counts()
      landing_outcomes
```

```
[52]: True ASDS      41
      None None      19
      True RTLS      14
      False ASDS      6
      True Ocean      5
      False Ocean     2
      None ASDS       2
      False RTLS      1
      Name: Outcome, dtype: int64
```

4. Create a landing outcome label from Outcome Column

```
[55]: # landing_class = 0 if bad_outcome
      # landing_class = 1 otherwise
      landing_class = []
      for key,value in df["Outcome"].items():
          if value in bad_outcomes:
              landing_class.append(0)
          else:
              landing_class.append(1)
```

Data Wrangling Cont.

5. Export to file

```
df.to_csv("dataset_part_2.csv", index=False)
```

https://github.com/seppanapally167/Applied-Capstone-Final-Project-2/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- There are 3 graphs that this lab uses and those three are Scatter Graphs, Bar Graph, and Line Graph.
- The Scatter Graph is used for observing the relationship between variables
- For this lab the Scatter Graph is used 6 times .
- The 6 Scatter Graphs are Flight Number vs. Payload Mass, Flight Number vs. Launch Site,Payload vs. Launch site, Orbit vs. Flight Number, Payload vs. Orbit Type, Orbit vs. Payload Mass
- The 2nd graph that is used is Bar Graph and the bar graph is used to compare differences between groups or follow changes over time.

EDA with Data Visualization

- The Bar Graph is only used one time for this lab and it compares the association between Success Rate and Orbit.
- The 3rd and final graph is the Line Graph
- The Line Graph is used to track changes over long and also short period of times and make predictions based on these changes
- The Line Graph is used only one time for this lab and it is used for the Success rate and the Year.

https://github.com/seppanapally167/Applied-Capstone-Final-Project-2/blob/main/edadataviz.ipynb

EDA with SQL

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string ‘CCA’
- Display the total payload mass carried by boosters launched by NASA(CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the boosters_versions which have carried the maximum payload mass. Use a subquery
- 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.
- 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.

https://github.com/seppanapally167/Applied-Capstone-Final-Project-2/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(4).ipynb

Build an Interactive Map with Folium

- There are three tasks required for this lab

1. Mark all launch sites on a map
2. Mark the success/failed launches for each site on the map
3. Calculate the distances between a launch site to its proximities

1. Mark all launch sites on a map

- For this task we marked all the launch sites, added map objects like markers, circles, and lines to success/failure of launches for each site in the folium map.

2. Mark the success/failed launches for each site on the map

- For this task we gave the feature launch outcomes (failure or success) a 0 or 1 . 0 means failure and 1 means success
- We also identified which launch sites have the highest success rate using color-labeled marker clusters

3. Calculate the distances between a launch site to its proximities

- This task main goal was to find the distance the launch site and its proximities
- For example, how close is the launch site to railways, highways and coastlines?
- Also, why does the launch site keep some distance away from the cities?
- These were some questions we had to answer for this task

https://github.com/seppanapally167/Applied-Capstone-Final-Project-2/blob/main/lab_jupyter_launch_site_location%20(2).ipynb

Build a Dashboard with Plotly Dash

- In this lab, we had to build a interactive dashboard with Plotly Dash
- We created a pie chart to show the total launches for specific sites
- We created a scatter graph showing the relationship between Outcome and Payload Mass for the different booster version

https://github.com/seppanapally167/Applied-Capstone-Final-Project-2/blob/main/spacex_dash_app%20(1).py

Predictive Analysis (Classification)

1. Building the Model
- We need to load the dataset into the NumPy and Pandas
- Then transform the data and split it into the training and test datasets
- Make sure you create parameters and algorithms to GridSearchCV and fit it in a dataset

2. Evaluating the Model
- Check the accuracy for each model
- Get tuned hyperparameters for each kind of algorithms.
- Lastly, plot the confusion matrix

3. Improving the Model
- Use the Feature Engineering and Algorithm Tuning to improve the model

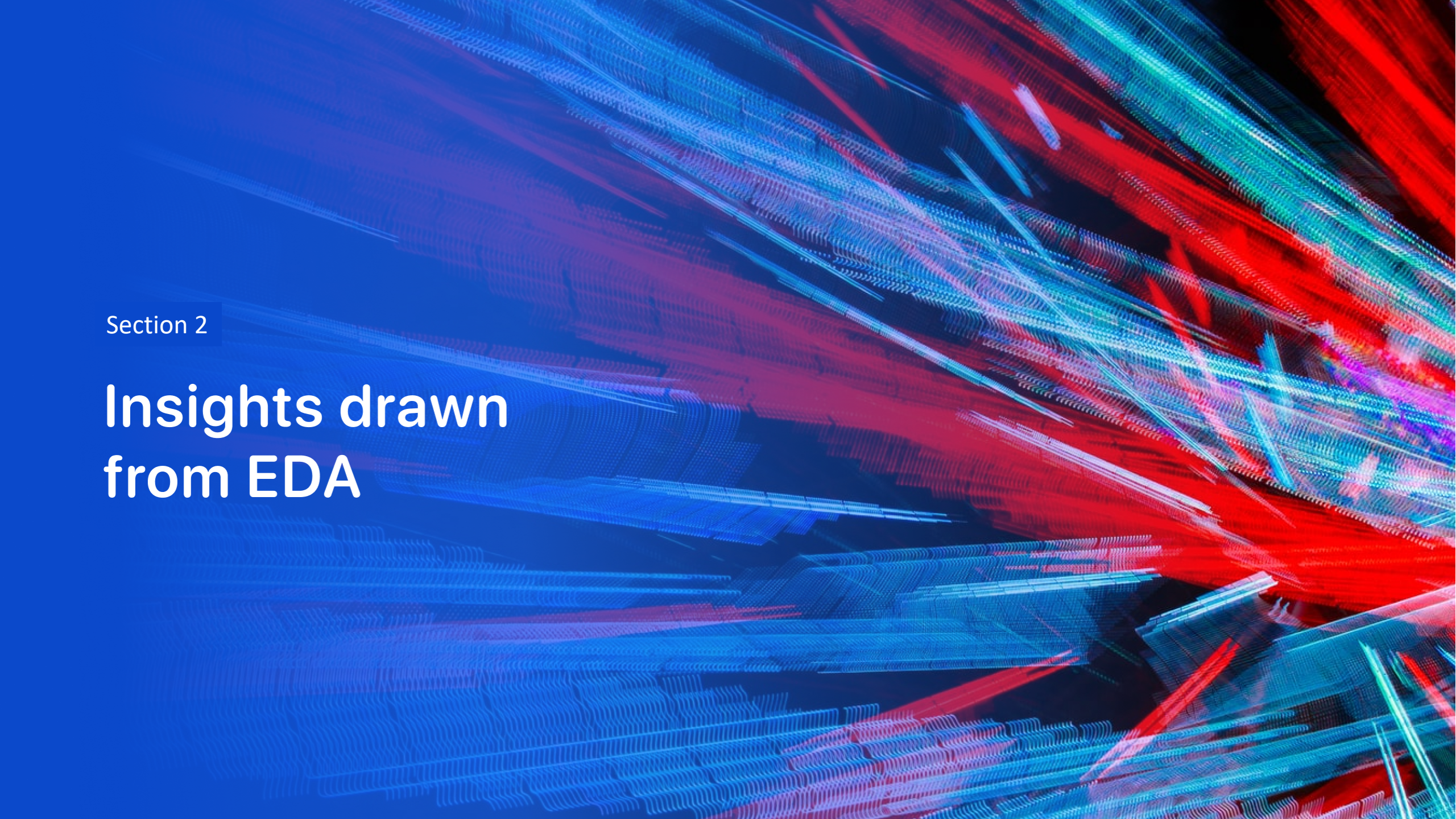
4. Find the Best Model
- The last step is to make sure you find the model with best accuracy score which will lead to the best performing model.

Predictive Analysis (Classification) GitHub Link

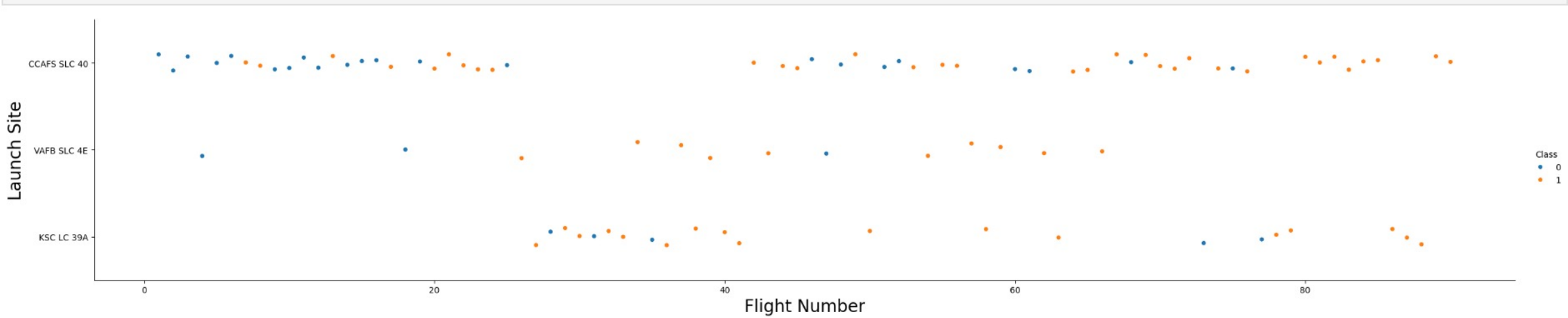
- https://github.com/seppanapally167/Applied-Capstone-Final-Project-2/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5%20(2).ipynb

Results

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

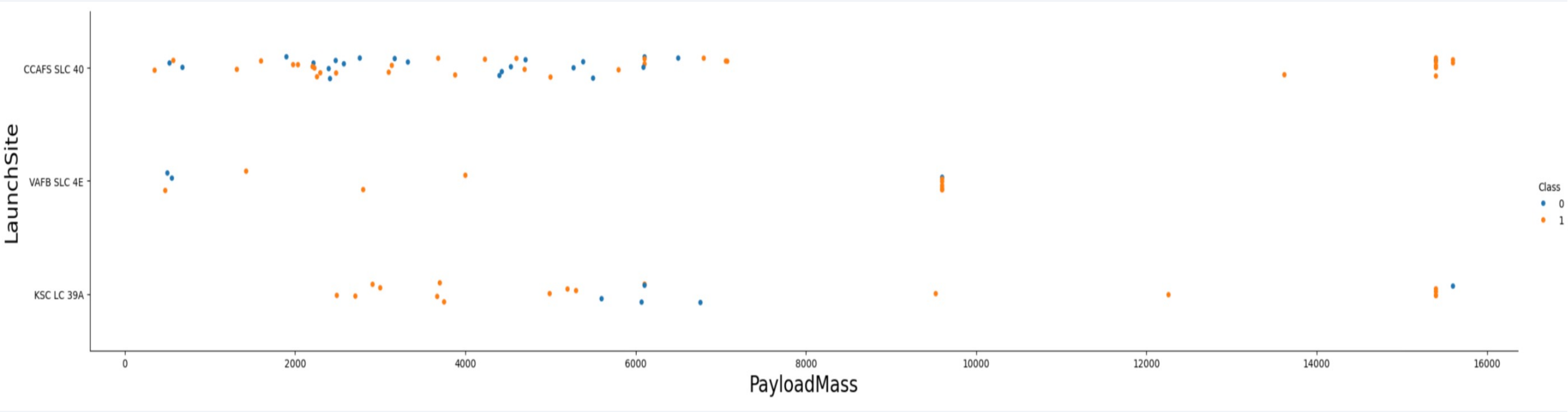

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



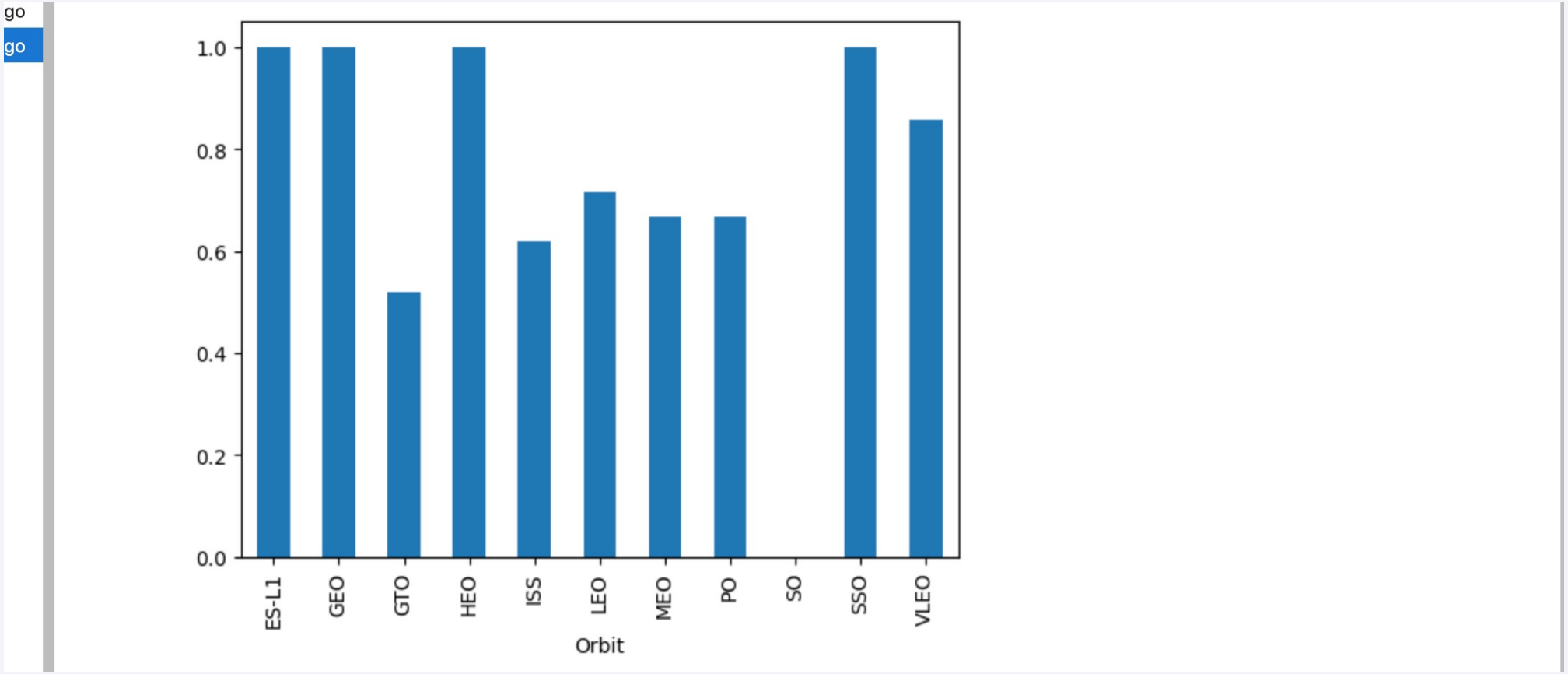
- Based on this Graph, we can see that in each 3 Launch Sites the success rate continues to increase.

Payload vs. Launch Site



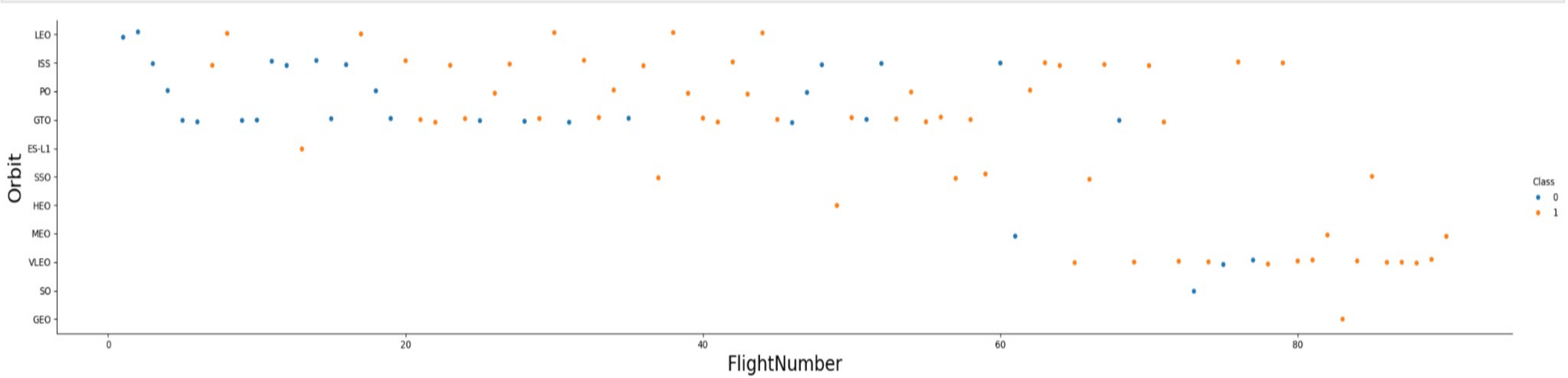
- Based on this graph, the conclusion we can make is depending on the launch site a heavier payload could lead to a successful landing. But, if the payload is way too heavy then the landing will not be successful.

Success Rate vs. Orbit Type



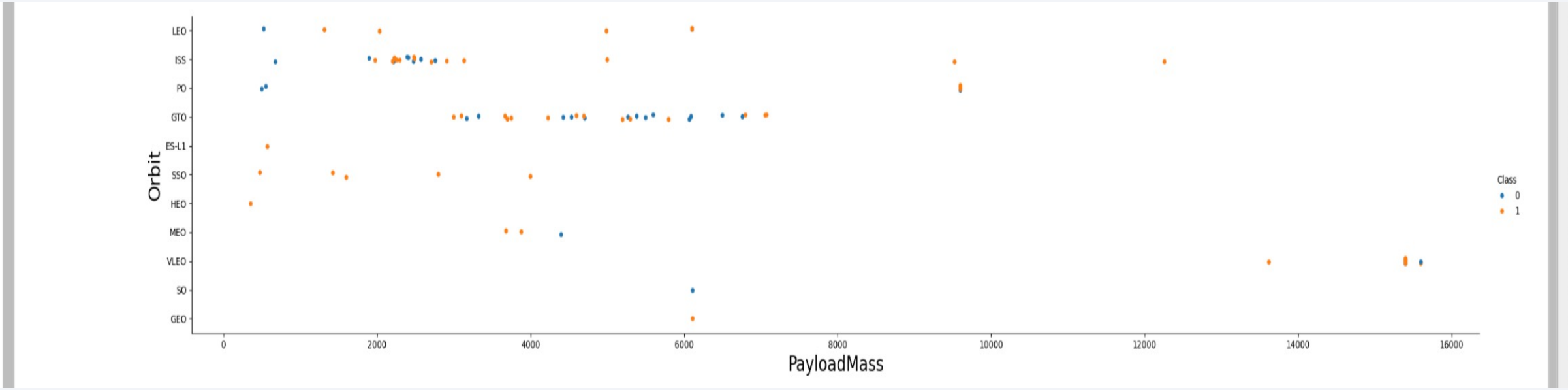
- Based on this Plot, there are multiple Orbits which have a success rate. There are 4 orbits which have the highest success rate.
- Those 4 are ES-L1, GEO, HEO, and SSO.

Flight Number vs. Orbit Type



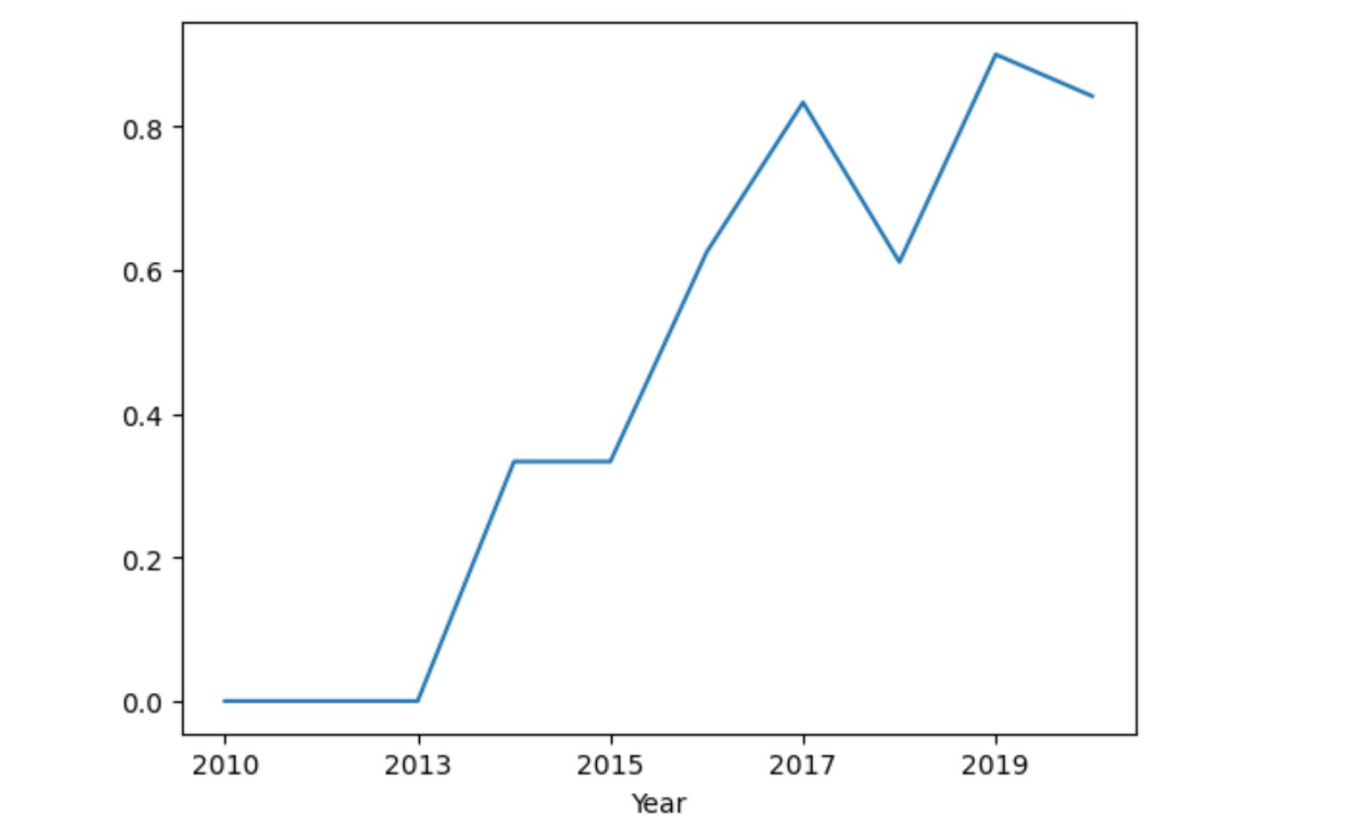
This graph shows that the success rate increases with the number of flights for the LEO orbit. But for other orbits like GTO, there is no similarity between the success rate and the number of flights. However, because other orbits have a high success rate such as SSO and HEO, we can make the conclusion that the insights these orbits have gathered from previous launches has led to the high success rate.

Payload vs. Orbit Type



- Based on this graph the conclusion we can make is that for certain orbits the weight of payloads can have a positive impact on the success rate. An example of this statement is the GEO Orbit. The success rate has increased for GEO because of the heavy payload. However, in some instances decreasing the payload can lead to a higher success rate. For example, decreasing the GTO orbit payload leads to a higher success rate.

Launch Success Yearly Trend



- We can see in this graph that the Success Rate of SpaceX had increased every year

All Launch Site Names

- What these results illustrate is the use of DISTINCT in the query leads to Launch_Site duplicate to be removed.

Task 1

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

```
[9]: sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1
 * sqlite:///my_data1.db
Done.
```

[9]:

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

Launch Site Names Begin with 'CCA'

- Based on these results, we can conclude that the WHERE clause is followed by LIKE clause which filters the launch sites which contains the substring CCA. This leads to the limit being 5 and that is why we only have 5 substrings.

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

```
[10]: sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
 * sqlite:///my_data1.db
Done.
```

[10]:

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

Total Payload Mass

Task 3

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

```
[11]: sql SELECT SUM (PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER='NASA (CRS)'
 * sqlite:///my_data1.db
Done.
```

[11]:

SUM (PAYLOAD_MASS__KG_)
45596

- Based on this query result, we can conclude that the query will return the sum of all of the payload masses for all of the customers are NASA(CRS). The result being 45,596.

Average Payload Mass by F9 v1.1

Task 4

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

[16]:
```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version = 'F9 v1.1'
```

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

[16]:

avg(PAYLOAD_MASS__KG_)
2928.4

- Based on the query result, we were able to find the average payload mass carried by Booster version F9 v1.1 . The average payload mass was 2928.4

First Successful Ground Landing Date

- Based on this query result, we found the date for the first successful ground landing. The first success successful landing date is 2015-12-22

Task 5

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

Hint:Use min function

[17]:
```
%%sql
SELECT min(Date)
FROM SPACEXTBL
WHERE Landing_Outcome = 'Success (ground pad)';
```

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

[17]:

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

```
[53]: ersion") FROM SPACEXTABLE WHERE "Landing_Outcome" = "Success (drone ship)" AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000
 * sqlite:///my_data1.db
Done.
```

[53]:

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

- This query illustrates the booster version where the landing was successful and the payload is between 4000 and 6000kg. The WHERE and AND clauses fliters main objective is to filter the dataset.

Total Number of Successful and Failure Mission Outcomes

- Based on the first SELECT, the subqueries shows the return results. The first subquery shows the mission failing. The second subquery is the mission being successful.

Task 7

List the total number of successful and failure mission outcomes

[23]:
```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number \
FROM SPACEXTBL \
GROUP BY MISSION_OUTCOME;
```

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

[23]:

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

Boosters Carried Maximum Payload

- Based on this result, we used a subquery to filter data which returns only the heaviest payload mass with the MAX function. The main query uses subquery results and returns booster versions with the heaviest payload mass.

Task 8

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

```
[17]: sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
 * sqlite:///my_data1.db
Done.
```

[17]:

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

2015 Launch Records

Task 9

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, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

[56]:
```
%sql SELECT substr(Date, 6, 2) as "Month", "Landing_Outcome", "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE "L
```

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

[56]:

Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- The query results illustrates that the month, booster version, and launch site and where the landing was unsuccessful and the landing date is 2015. the Substr function processes dates in order to take the month or the year. Substr(Date, 6,2,) shows month. Substr(DATE, 0,5,) shows year.

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

- Based on the results, we can see that the query returns the landing outcomes and their count where the mission is successful and the date between 2010-06-04 and 2017-03-20. The Landing_Outcomes with No attempt has the highest count and the Precluded(drone ship) has the lowest count. The number count goes down for each Landing_outcome.

Task 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.

[57]:
```
%%sql SELECT "LANDING_OUTCOME", COUNT(*) as 'COUNT' FROM SPACEXTBL

WHERE substr(Date,1,4) || substr(Date,6,2) || substr(Date,9,2)

between '20100604' and '20170320' GROUP BY "Landing_Outcome" ORDER BY "COUNT" DESC;
```

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

[57]:

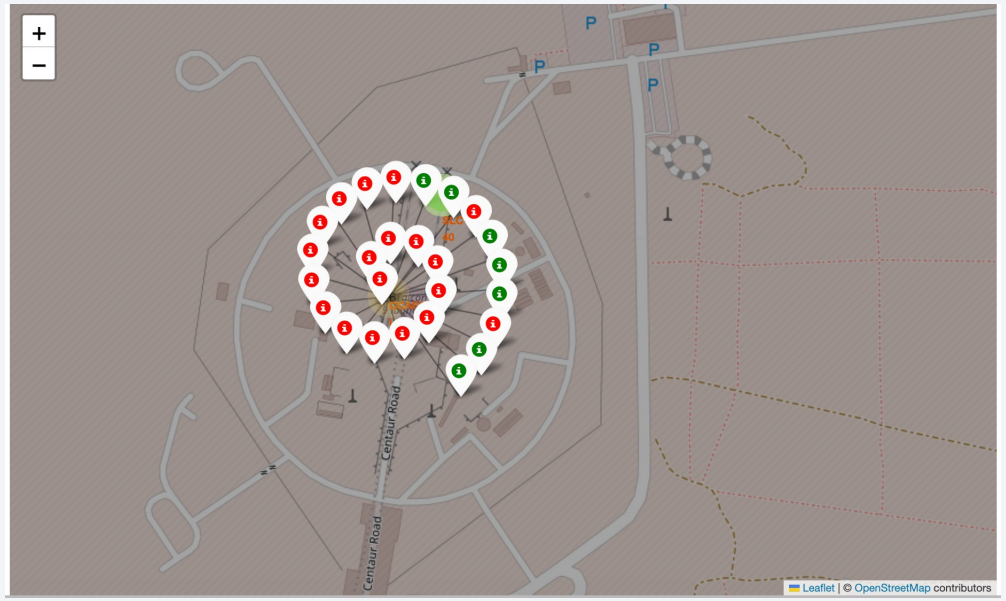
Landing_Outcome	COUNT
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1


Section 3
Launch Sites
Proximities Analysis

All Launch Sites on the map

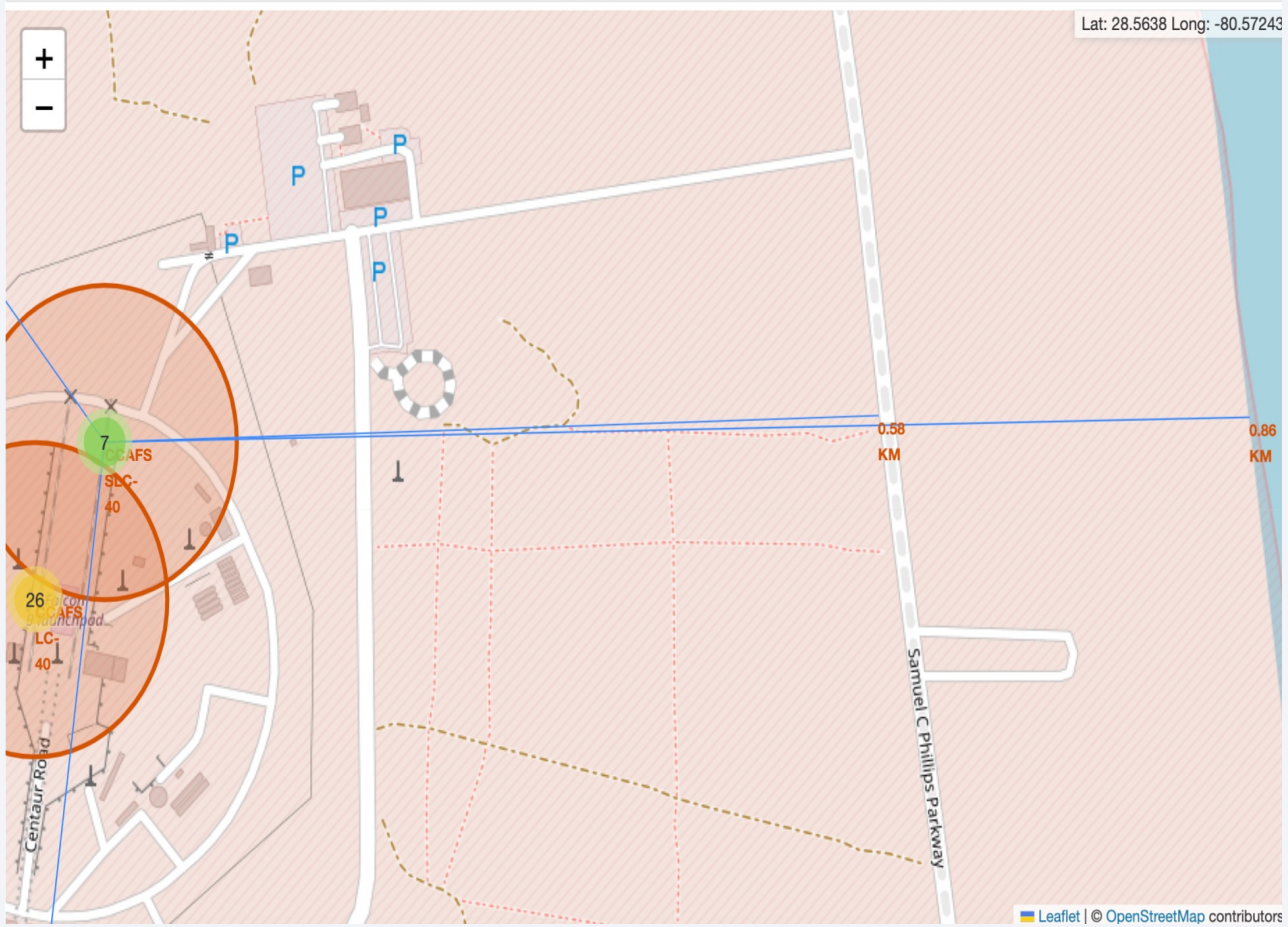
- The Launch Sites are labelled with a mark with the names on the map

Success/Failed launch sites on the map



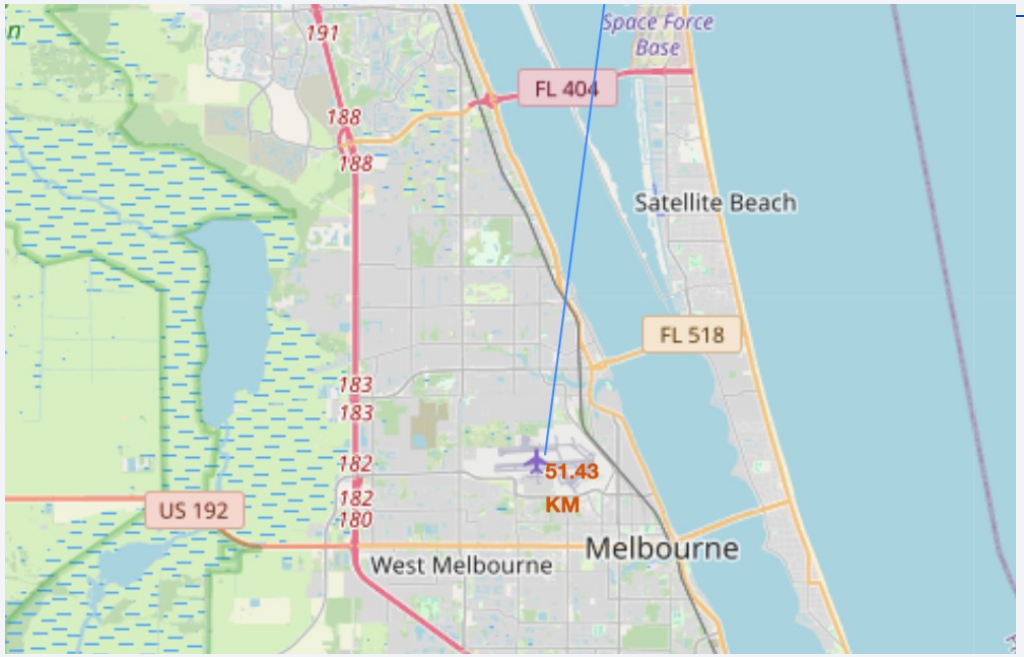
- The launch sites are assembled as a group on the map. Then when you press on the launch site you will see either a red dot or a green dot. The red dot resembles a unsuccessful launch and a green label resembles a successful launch.

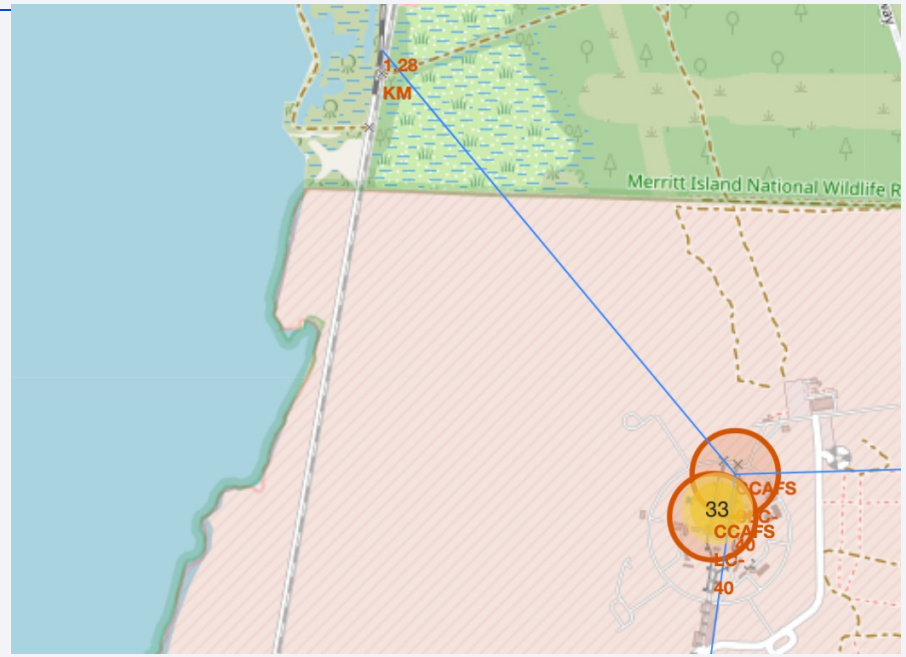
Distance between Launch Site to their Proximities



- The distance between the launch site of CCAFSLC-40 is 0.86 KM.

Distance between Launch Site and closest Highway, Railroad and City

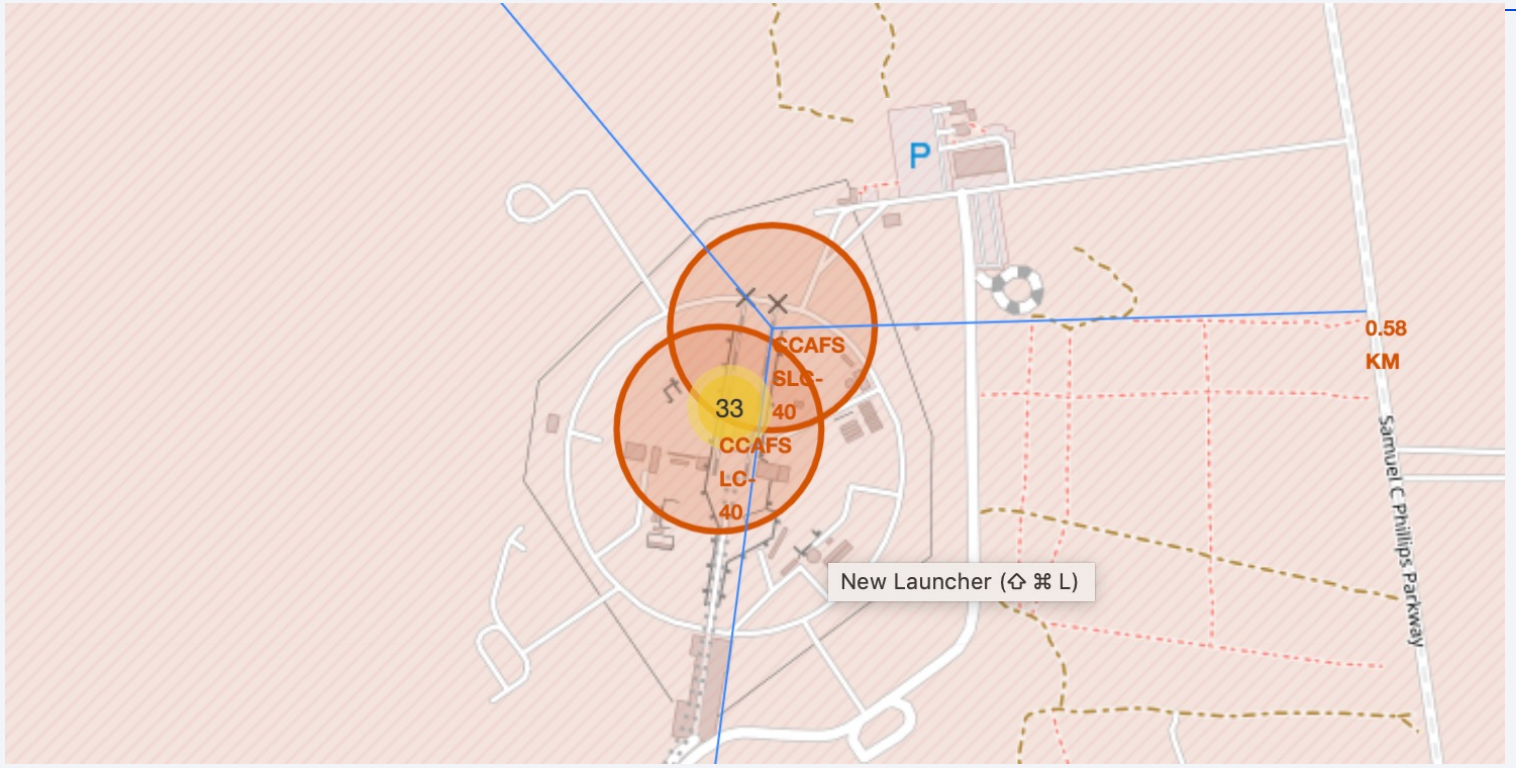




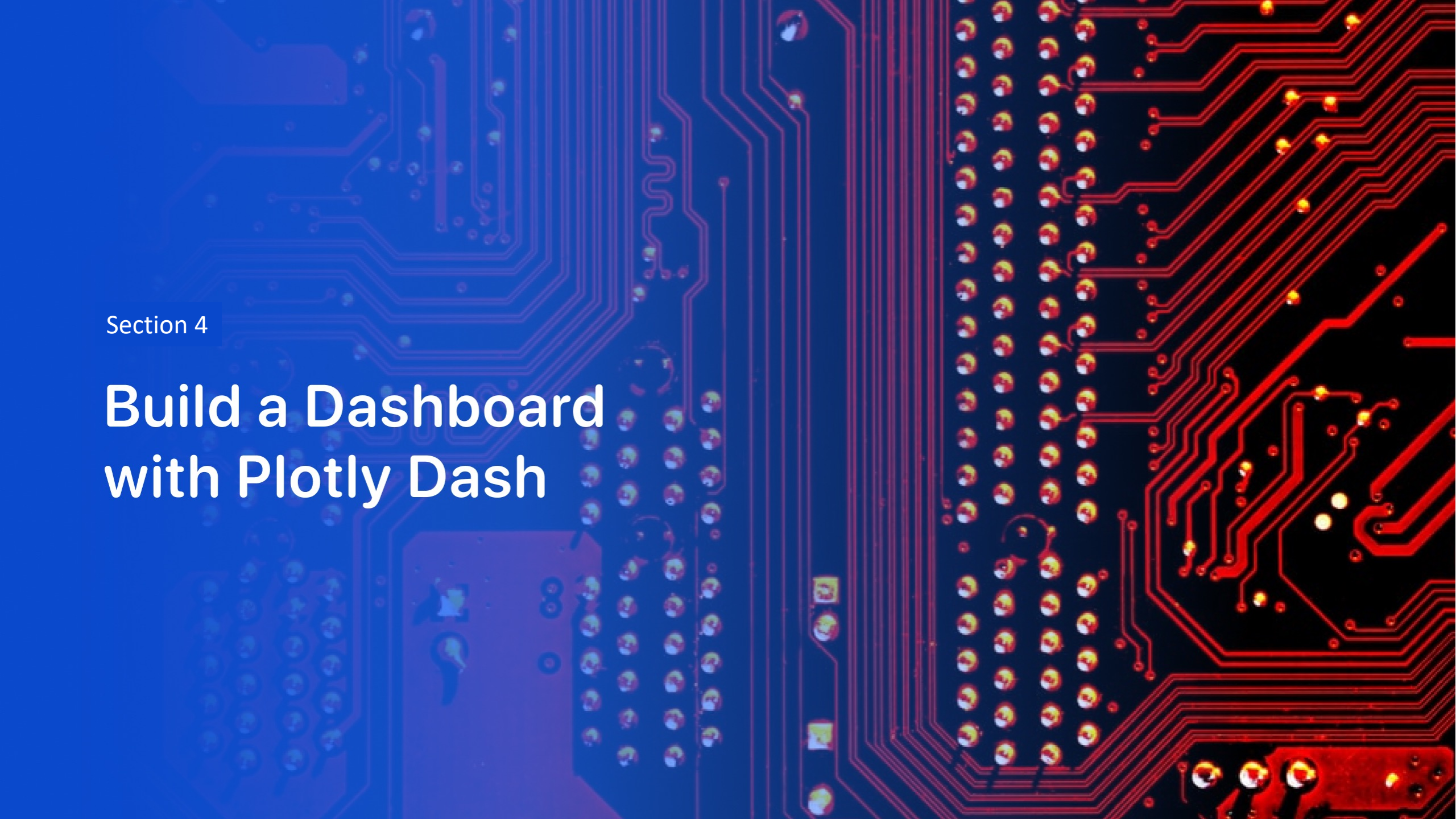
- The distance between the launch site and closest highway is 51.43 KM

- The distance between the launch site and closest railroad is 1.28 KM

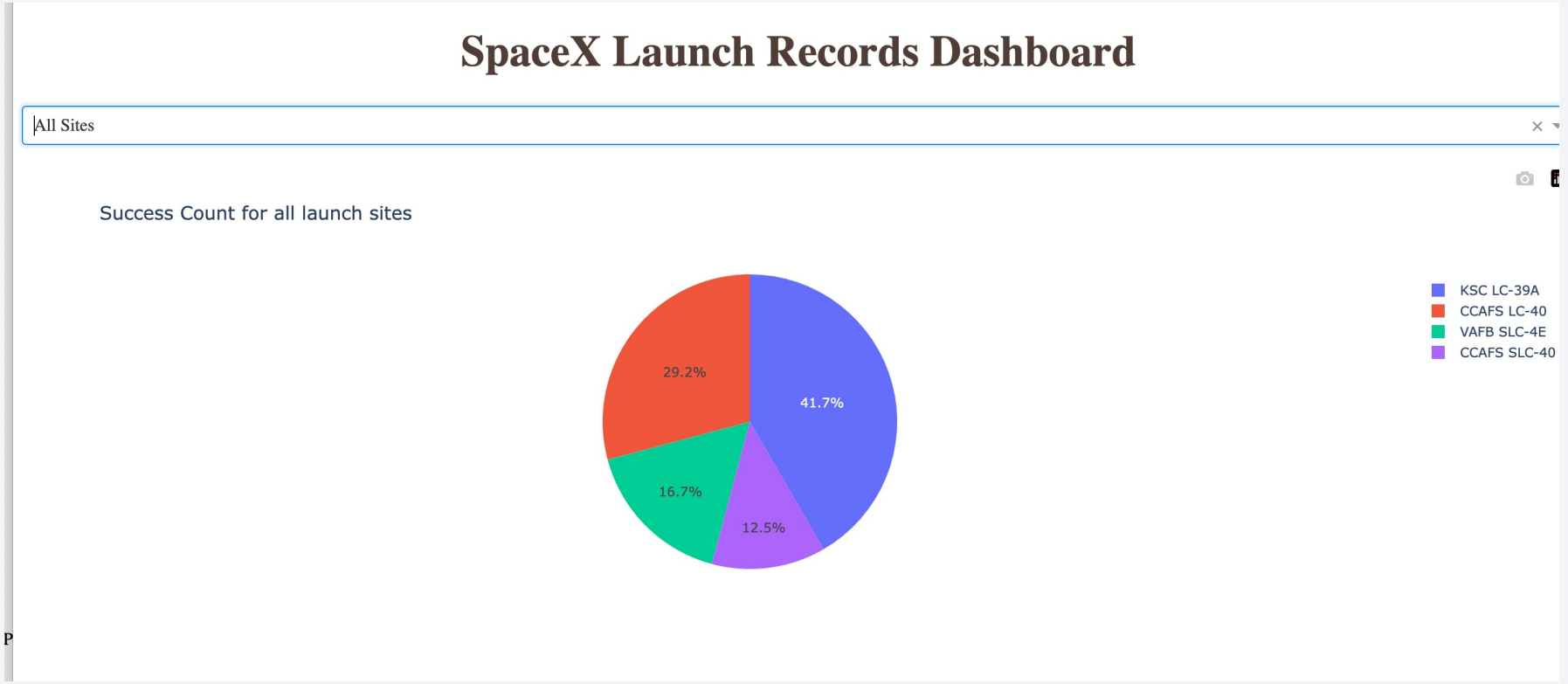
Distance between Launch Site and closest Highway, Railroad and City Cont.



- The distance between the launch site and closest city is 0.58 KM

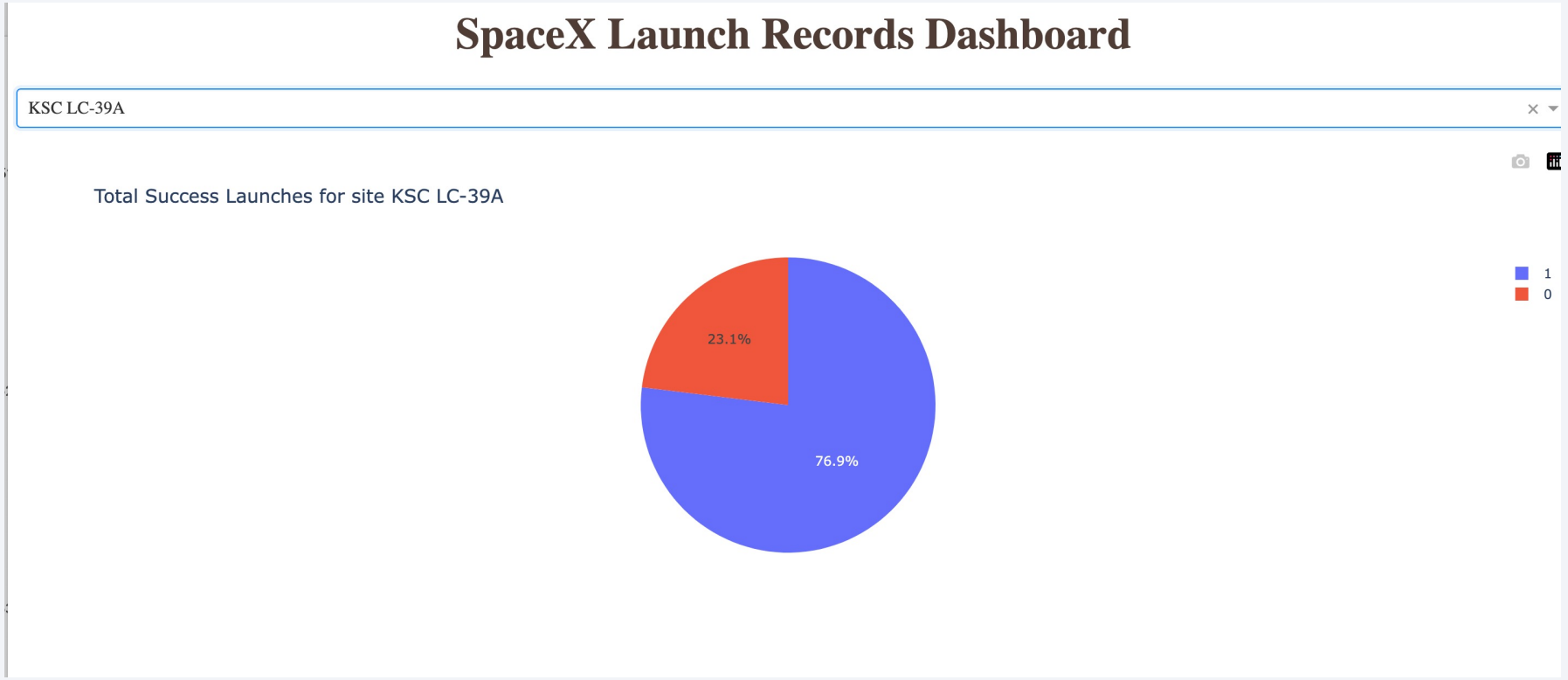

Section 4
Build a Dashboard with Plotly Dash

Launch Success for all sites



- Based on the total launch success for all of the Launch Sites, the launch site with the highest success rate is KSC-LC-39A with 41.7%. The second highest is CCAFS LC-40 with 29.2% and the third highest is VAFB SLC-4E with 16.7%. The lowest success rate is CCAFS SLC-40 with 12.5%

Highest Launch Site Success Ratio



- The highest success rate is KSC LC-39A with success rate is 41.7%.The successful landing rate is 76.9% and the unsuccessful landing rate is 23.1%.

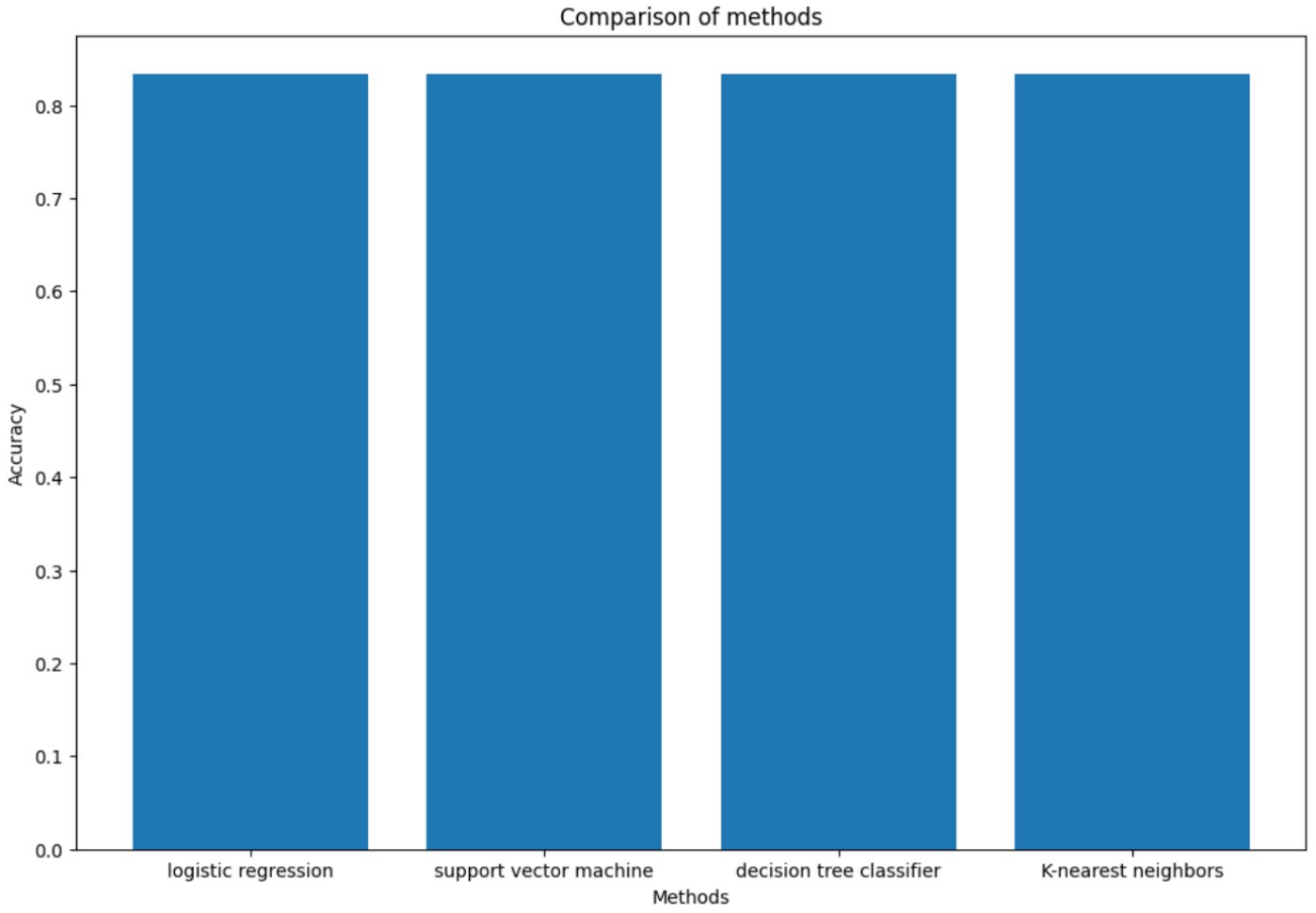
Payload vs. Launch Outcome scatter plot for all sites

- According to the scatter plot the payload range with the highest success launch is from 2,000kg and 4,000kg because it has the most plot points. The second highest payload range is from 4,000kg to 6,000kg because it has the second most plot points.
- The booster version with the highest successful launches is FT which are the green dots. FT has approximately 20 dots. The second highest is B4 which are the purple dots and they have approximately 14 dots

Section 5

Predictive Analysis (Classification)

Classification Accuracy

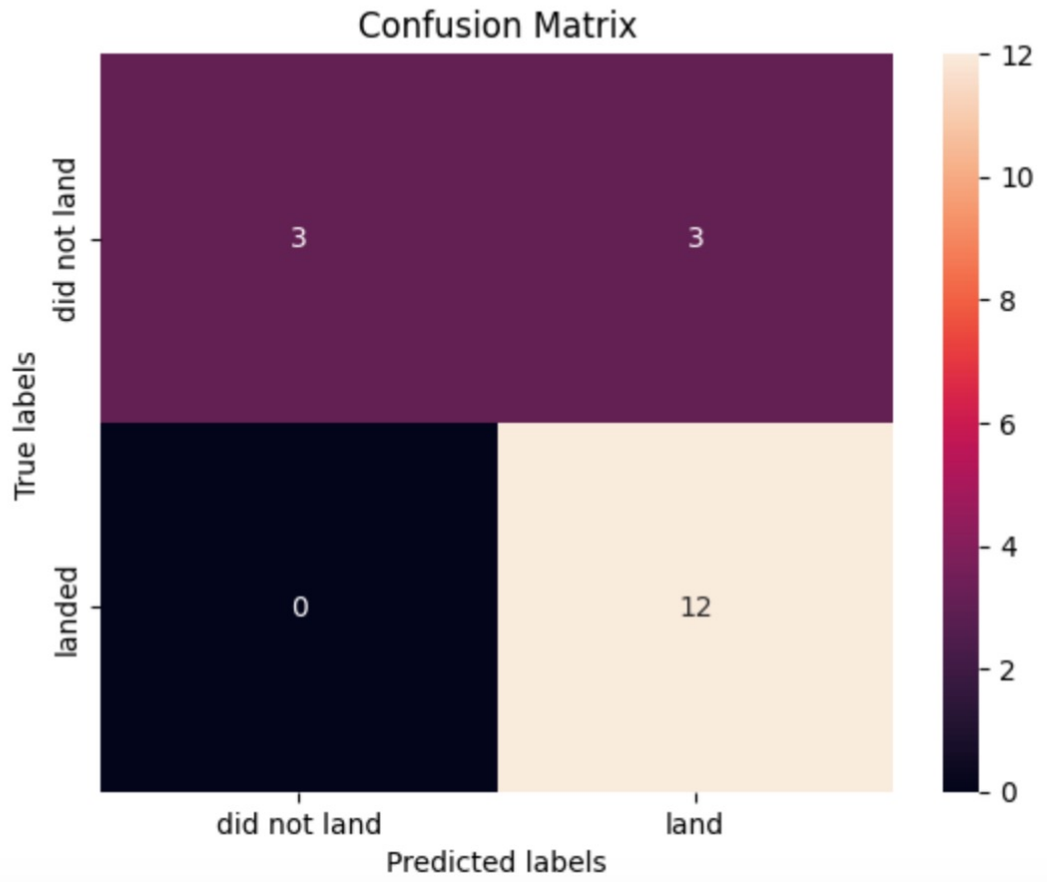


- The model with best accurate score is the tree classifier with a score of 0.8625. Therefore, we must use decision tree classifier as the model.

Confusion Matrix

We can plot the confusion matrix

```
[86]: yhat = tree_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```



- The confusion matrix for the decision tree classifier illustrates that the classifier can interpret the different classes. However, the biggest problem is it cannot distinguish the difference between the false positives.
- For example, was it really a successful landing or was it actually an unsuccessful landing.

Conclusions

- ES-L1,GEO,HEO and SSO are the orbit types with the highest success launch rates.
- Lower payloads leads to a higher performance compared to heavier payloads
- The model with best score was the decision tree classifier.
- The launch site with the highest success rate was the KSC-LC-39A with 41.7%.
- Lastly, the more experience that SpaceX has the greater the chance that it has to land successfully.

Thank you!