IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Michael Dromin
31.05.2023

Outline

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

Executive Summary

Summary of methodologies

- SpaceX Launch Data Collection using SpaceX API
- SpaceX Launch Data Collection with Web Scraping
- Data Set Wrangling
- SpaceX Exploratory Data Analysis using SQL
- Data Visualization Using Python Pandas and Matplotlib
- Launch Sites Analysis with Folium, Visual Analytics and PlotyDash
- SpaceX Machine Learning Landing Prediction

Summary of all results

- EDA results
- Interactive Visual Analytics and Dashboards
- Predictive Analysis (Classification)

Introduction

- Project background and context

SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars which is 165 million dollars more cost-effective than other providers do as SpaceX can reuse the first stage. If we are able to determine if the first stage lands, we can define the launch cost.

- Problems you want to find answers

In this project, we are going to predict if the Falcon 9 first stage will land successfully using data from Falcon 9 rocket launches advertised on its website.

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

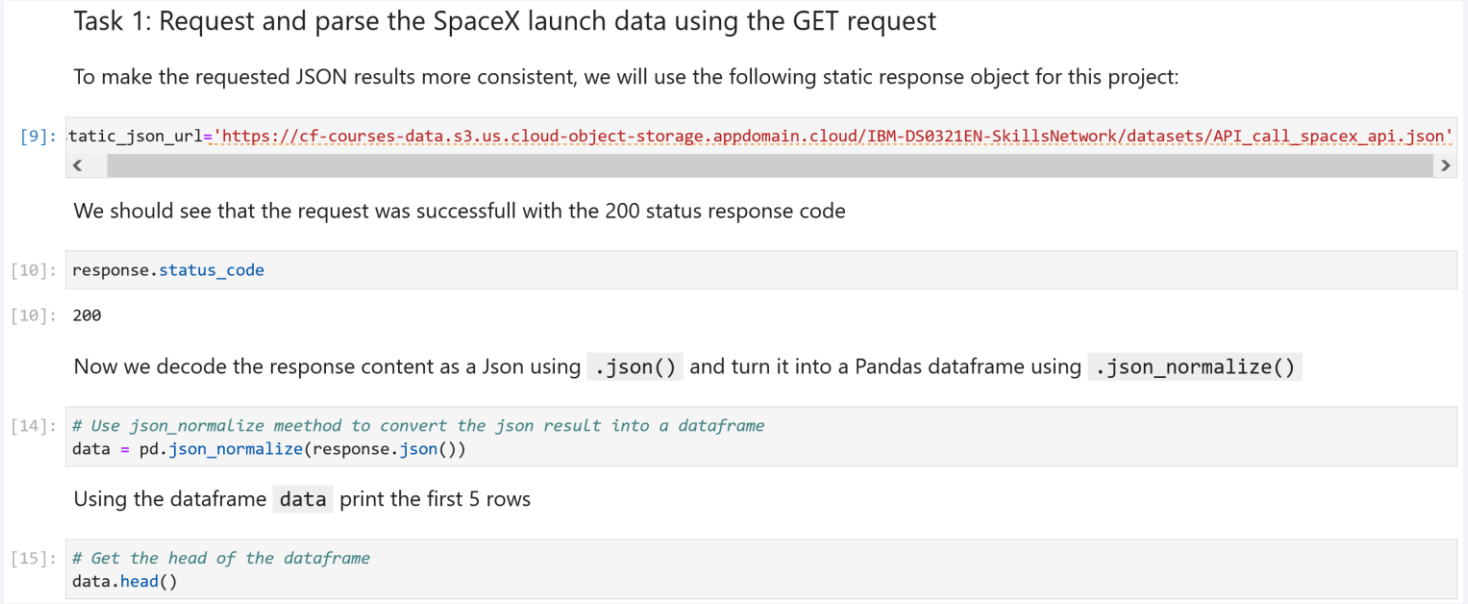
Data Collection

- Data is collected using SpaceX API (a RESTful API) by making a get request to the SpaceX API. This was done by defining a series helper functions that would help in the use of the API to extract information using identification numbers in the launch data and then requesting rocket launch data from the SpaceX API url.
- SpaceX launch data was requested and parsed using the GET request and then decoded the response content as a Json result which was then converted into a Pandas data frame.
- Used web scraping to collect Falcon 9 historical launch records from a Wikipedia page are stored in a HTML. Using Beautiful Soup and request Libraries, I extract the Falcon 9 launch HTML table records from the Wikipedia page, Parsed the table and converted it into a Pandas data frame

Data Collection – SpaceX API

- Data collected using SpaceX API with get request then requested and parsed the SpaceX launch data using the GET request and decoded the response content as a Json result which was then converted into a Pandas data frame
- Here is the GitHub URL of the completed SpaceX API calls notebook

https://github.com/MikhailDt/Capstone_project_MD/blob/main/1_Data-collection-api.ipynb



Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

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

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

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

```
[10]: 200
```

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

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

Using the dataframe `data` print the first 5 rows

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

Data Collection - Scraping

Performed web scraping to collect Falcon 9 historical launch records from a Wikipedia using Beautiful Soup and request, to extract the Falcon 9 launch records from HTML table of the Wikipedia page, then created a data frame by parsing the launch HTML.

https://github.com/MikhailDt/Capstone_project_MD/blob/main/2_Webscraping.ipynb

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

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

```
# use requests.get() method with the provided static_url
response = requests.get(static_url)

# assign the response to a object
```

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

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

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

```
# Use soup.title attribute
soup.title
```

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

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

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

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

```
# 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')
```

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

[19]:

	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], \n]	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	[[.mw-parser-output .plainlist ol,.mw-parser-o...	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	[[NASA], (, [COTS],)\n]	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	[[NASA], (, [CRS],)\n]	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	[[NASA], (, [CRS],)\n]	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

Data Wrangling

- After obtaining and creating a Pandas DF from the collected data, data was filtered using the BoosterVersion column to only keep the Falcon 9 launches, then dealt with the missing data values in the LandingPad and PayloadMass columns. For the PayloadMass, missing data values were replaced using mean value of column.
- Also performed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models

https://github.com/MikhailDt/Capstone_project_MD/blob/main/3_Data_wrangling.ipynb

```
[13]: # landing_class = 0 if bad_outcome
      # landing_class = 1 otherwise
      df['Class'] = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
      df['Class'].value_counts()

[13]: 1    60
      0    30
      Name: Class, dtype: int64
```

This variable will represent the classification variable that represents the outcome of each
Successfully

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

[15]:

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

```
[16]: df.head(5)
```

[16]:

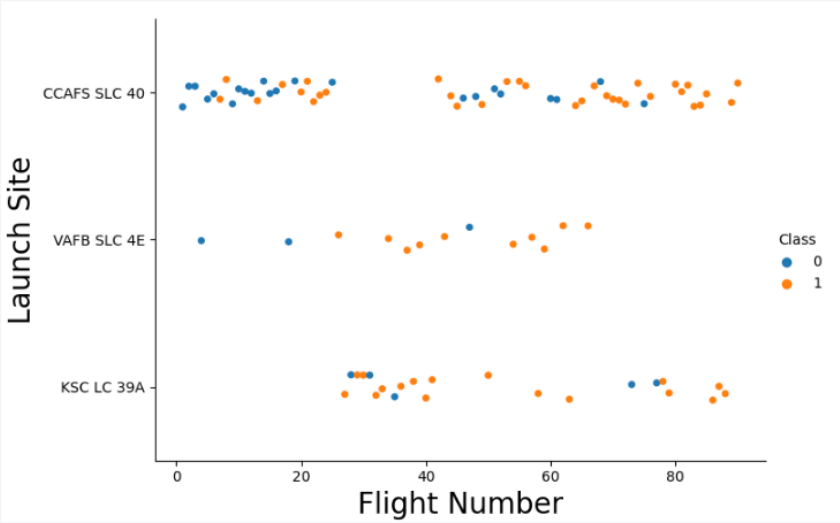
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

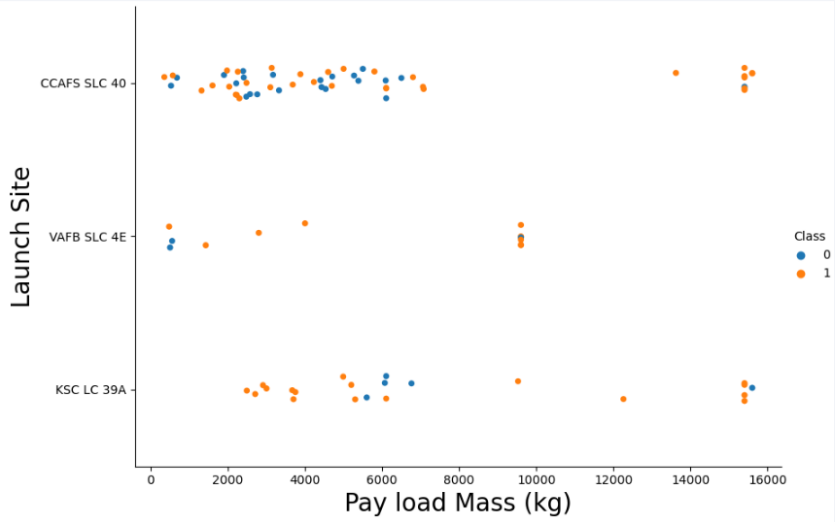
EDA with Data Visualization

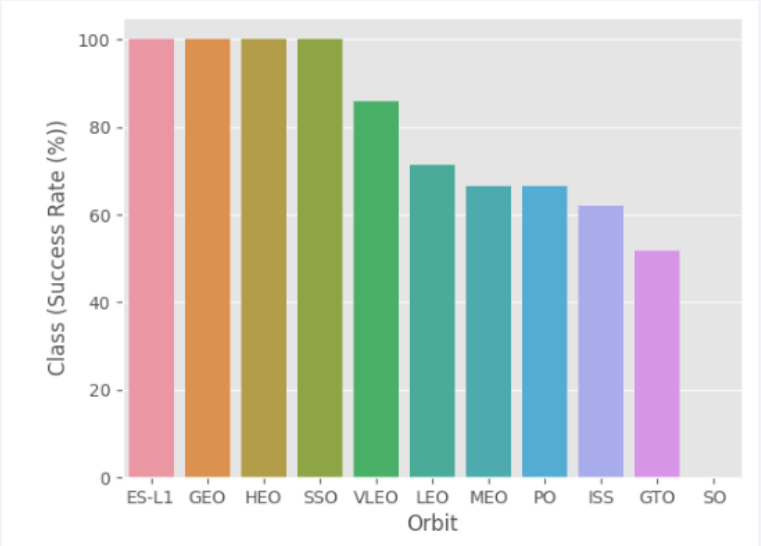
- Performed data Analysis and Feature Engineering using Pandas and Matplotlib i.e.
- Exploratory Data Analysis
- Preparing Data Feature Engineering
- Used scatter plots to Visualize the relationship between Flight Number and Launch Site, Payload and Launch Site, FlightNumber and Orbit type, Payload and Orbit type.
- Used Bar chart to Visualize the dependence between success rate of each orbit type
- Line plot to Visualize the launch success yearly trend.
- Here is the GitHub URL of your completed EDA with data visualization notebook

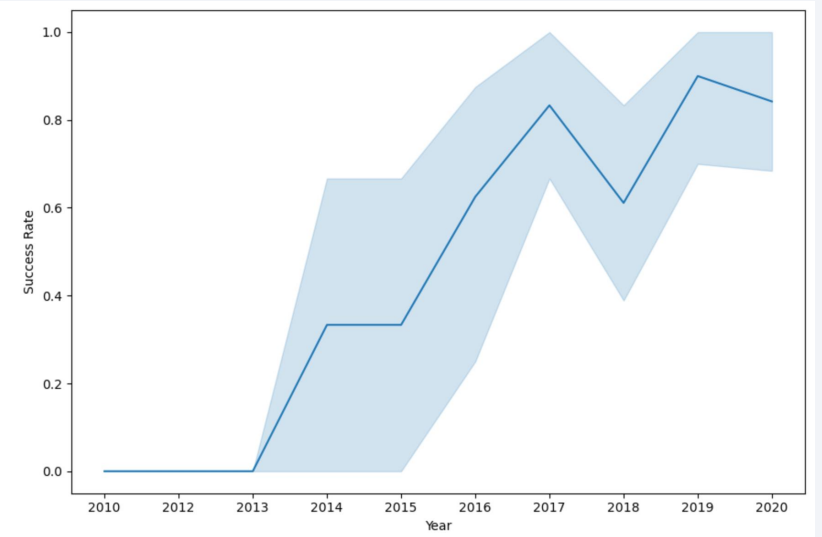
https://github.com/MikhailDt/Capstone_project_MD/blob/main/5_EDA-Data%20Visualization.ipynb

EDA with Data Visualization









EDA with SQL

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

```
[7]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

[7]:

Launch_Sites
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
None

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

```
[8]: %sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

[8]:

Date	Time (UTC)	Booster_Version	Launch_Site	
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon S
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSa
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	

Task 3

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

```
[ ]: %sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

Task 4

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

```
[9]: %sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';
```

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

[9]:

Payload Mass Kgs	Customer	Booster_Version
2534.6666666666665	MDA	F9 v1.1 B1003

Build an Interactive Map with Folium

- Created folium map with marked all the launch sites, and created map objects such as markers, circles, lines to mark the success or failure of launches for each launch site.
- Created a launch set outcomes (failure=0 or success=1).
- Here is the GitHub URL of the completed interactive map with Folium map, as an external reference and peer-review purpose

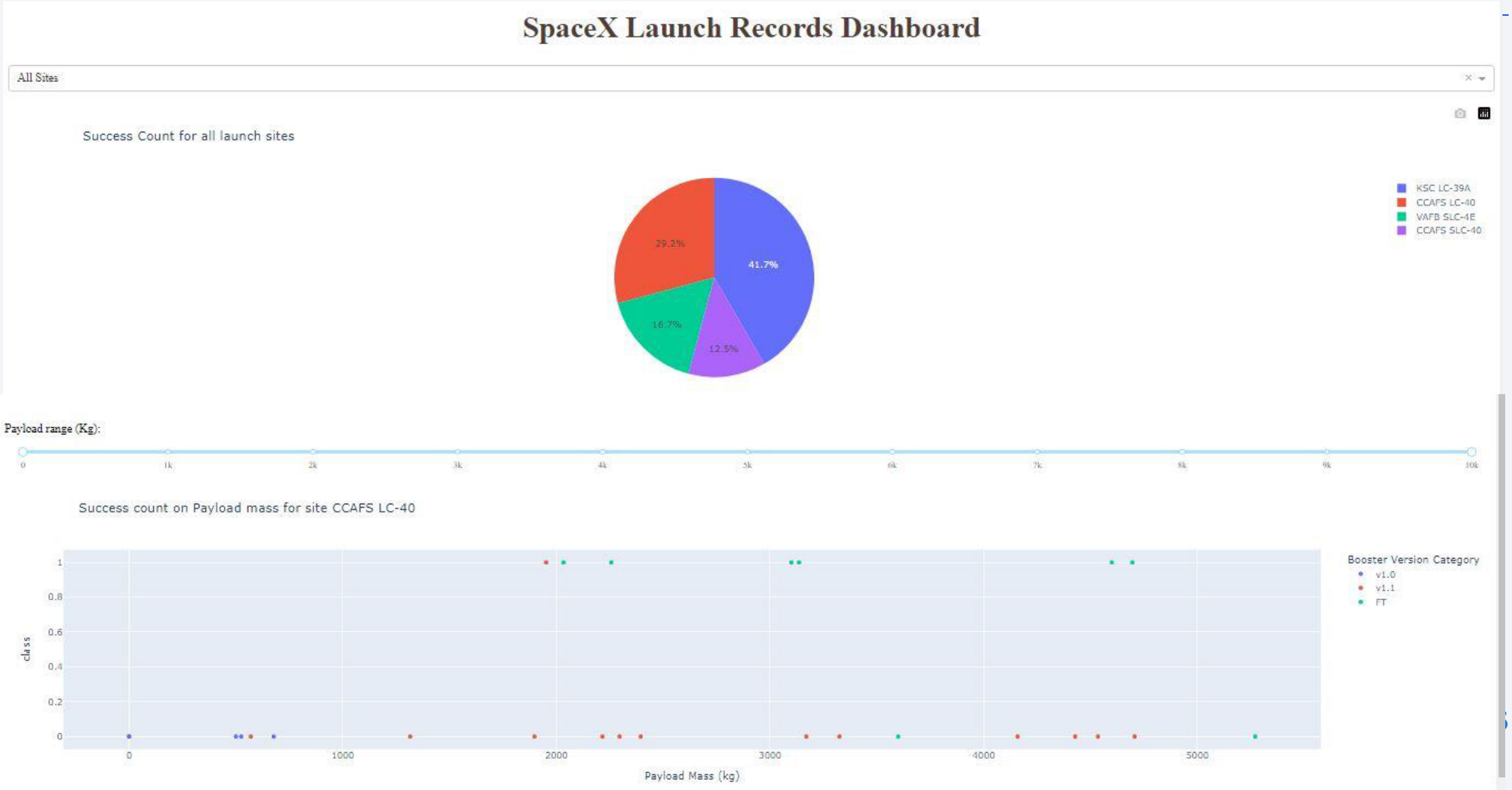
https://github.com/MikhailDt/Capstone_project_MD/blob/main/6_Launch%20site%20location_Folium.ipynb

Build a Dashboard with Plotly Dash

- Built an interactive dashboard application with Plotly Dash by:
- Adding a Launch Site Drop-down Input Component
- Adding a callback function to render success-pie-chart based on selected site dropdown
- Adding a Range Slider to Select Payload
- Addenga callback function to render the success-payload-scatter-chart scatter plot

https://github.com/MikhailDt/Capstone_project_MD/blob/main/7_Build%20an%20Interactive%20Dashboard%20with%20Ploty%20Dash

Build a Dashboard with Plotly Dash



Predictive Analysis (Classification)1

- After loading the data as a Pandas Dataframe, I set out to perform exploratory Data Analysis and determine Training Labels by;
- creating a NumPy array from the column Class in data, by applying the method to_numpy() then assigned it to the variable Y as the outcome variable.
- Then standardized the feature dataset (x) by transforming it using preprocessing.StandardScaler() function from Sklearn.
- After which the data was split into training and testing sets using the function train_test_split from sklearn.model_selection with the test_size parameter set to 0.2 and random_state to 2.

Predictive Analysis (Classification)2

In order to find the best ML model/ method that would performs best using the test data between SVM, Classification Trees, KNN and Logistic Regression:

- First created an object for each of the algorithms then created a GridSearchCV object and assigned them a set of parameters for each model.
- For each of the models under evaluation, the GridsearchCV object was created with cv=10, then fit the training data into the GridSearch object for each to Find best Hyperparameter.
- After fitting the training set, we output GridSearchCV object for each of the models, then displayed the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_.
- Finally using the method score to calculate the accuracy on the test data for each model and plotted a confussion matrix for each using the test and predicted outcomes.

Predictive Analysis (Classification)3

Comparing data accuracy to show which method performed best using the test data between SVM, Classification Trees, KNN and Logistic Regression;

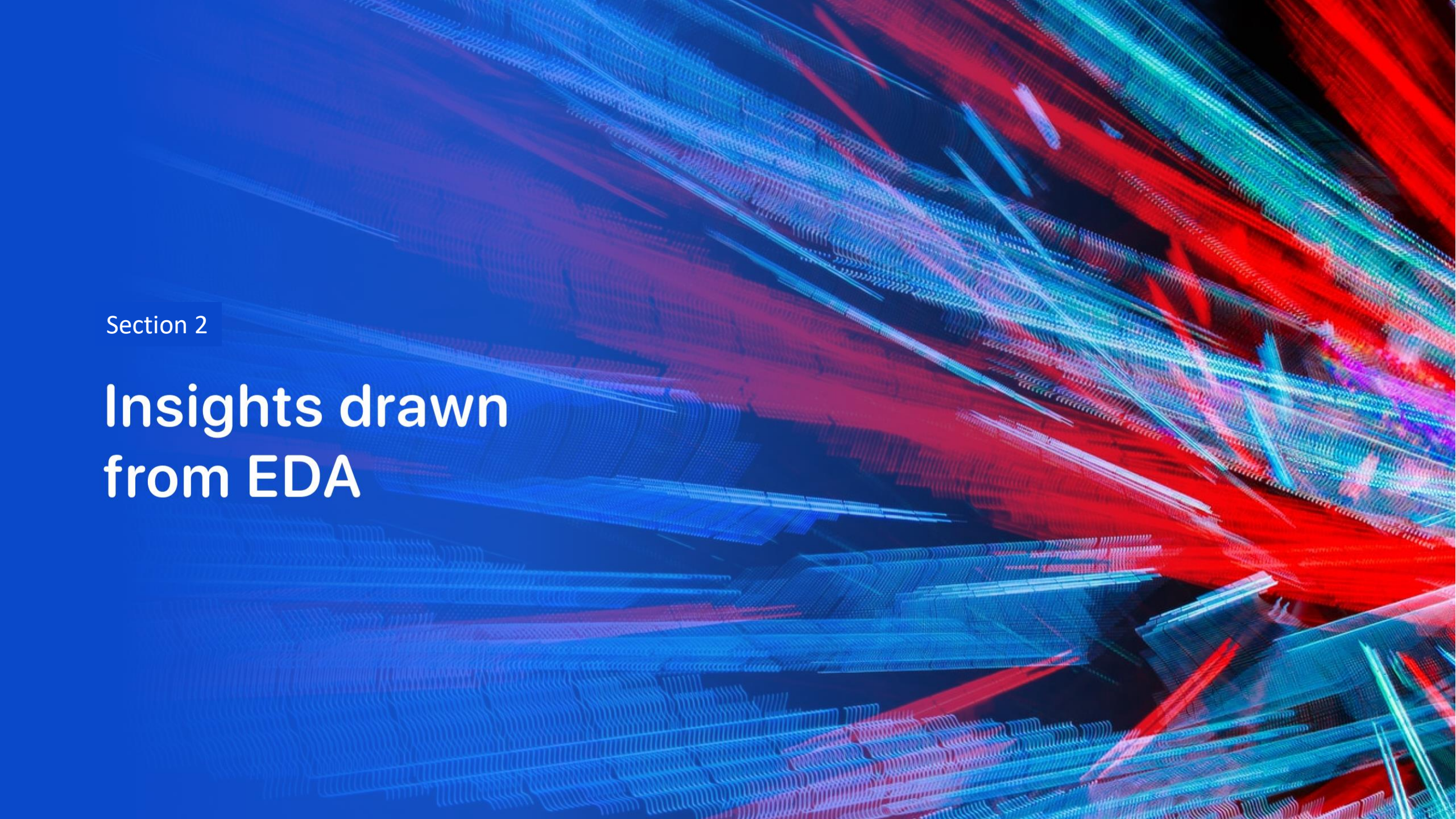
[32]:

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

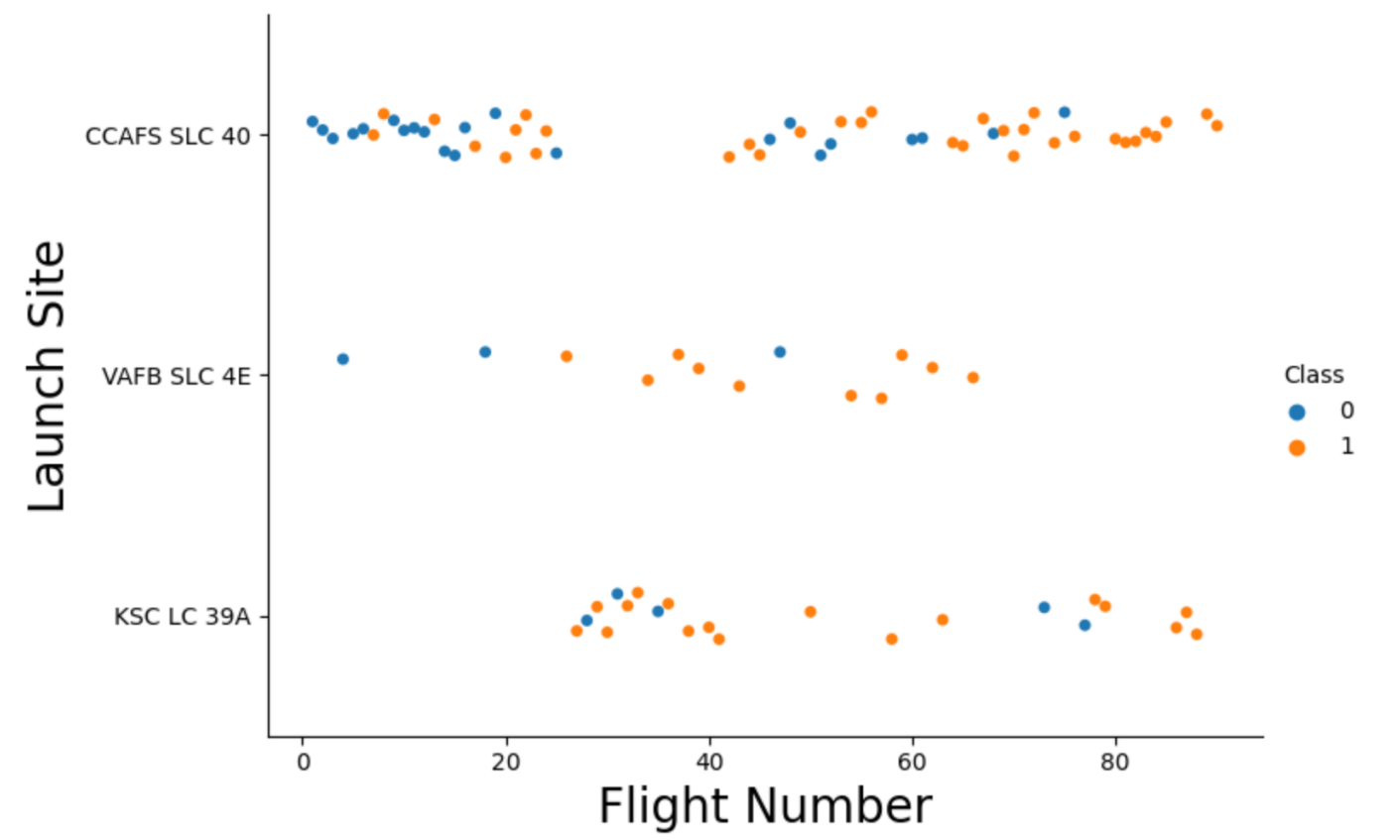
https://github.com/MikhailDt/Capstone_project_MD/blob/main/8_Machine_Learning_Prediction.ipynb

Results

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

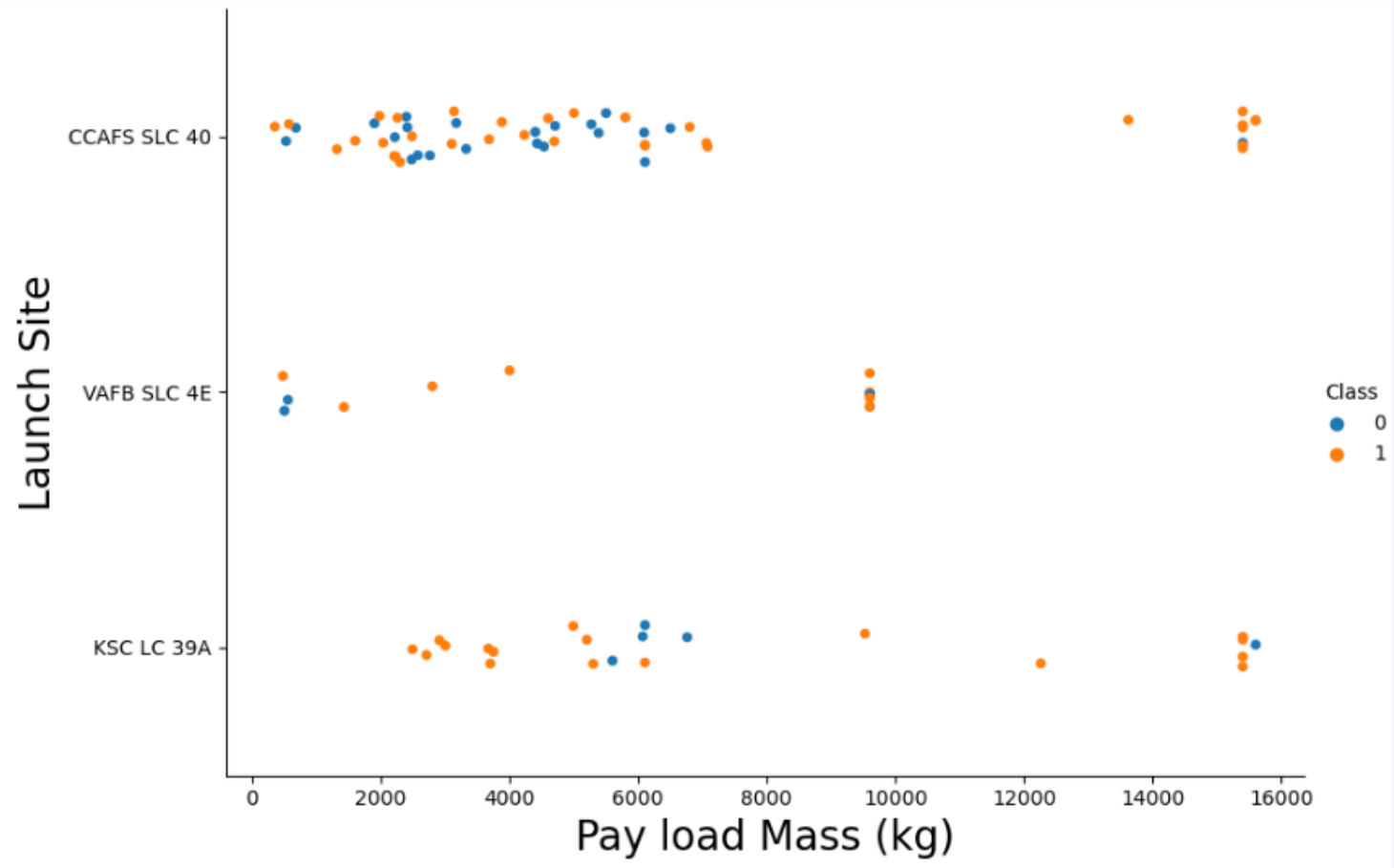

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



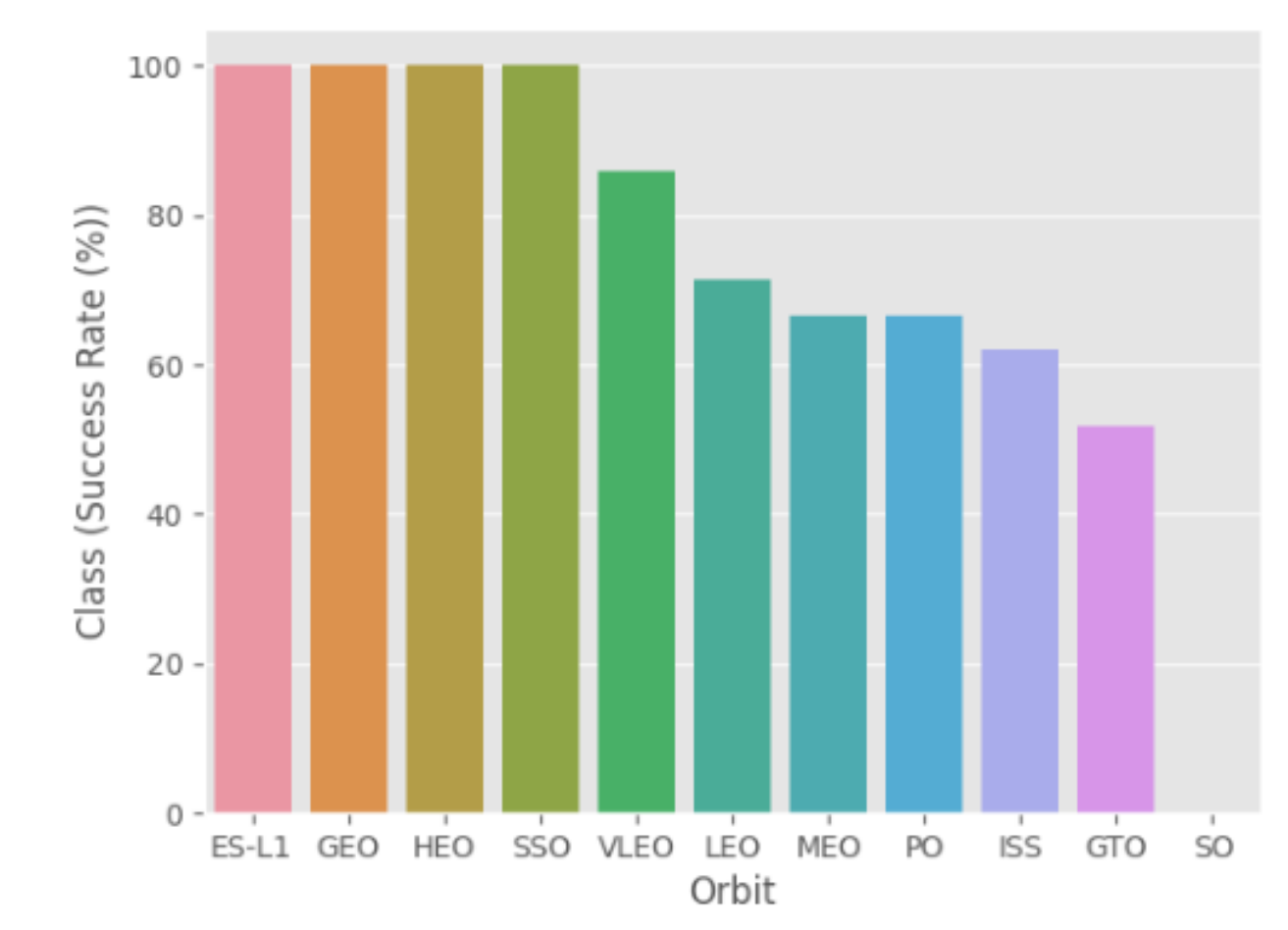
Success rate and Flight number increase for each launching site. For CCAFS 100% successful flights after 75 launches, for VAFB – after 50 flights, for KSC – after 80 launches

Payload vs. Launch Site



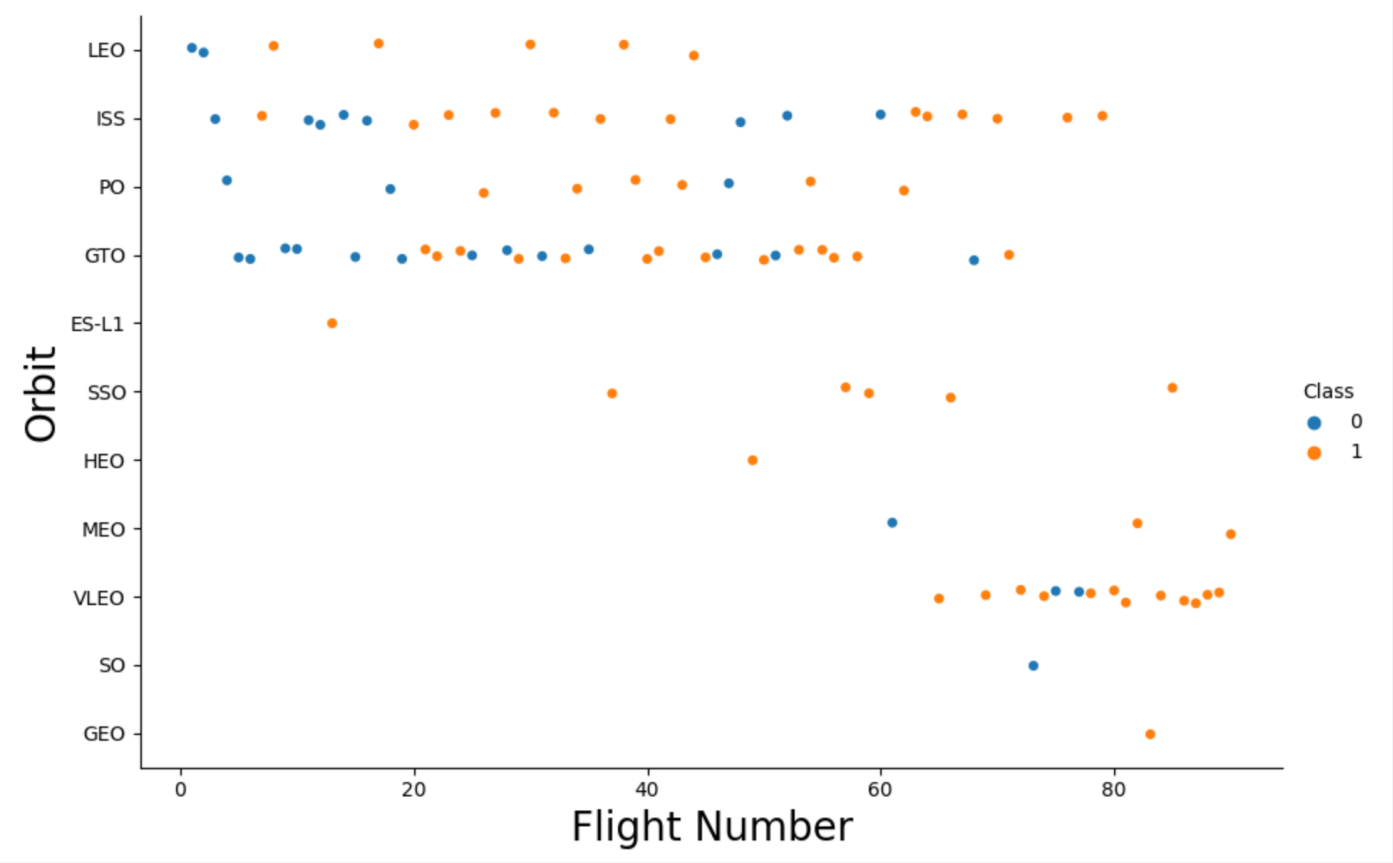
There are no rockets launched for high payload mass (> 10,000 kg)

Success Rate vs. Orbit Type



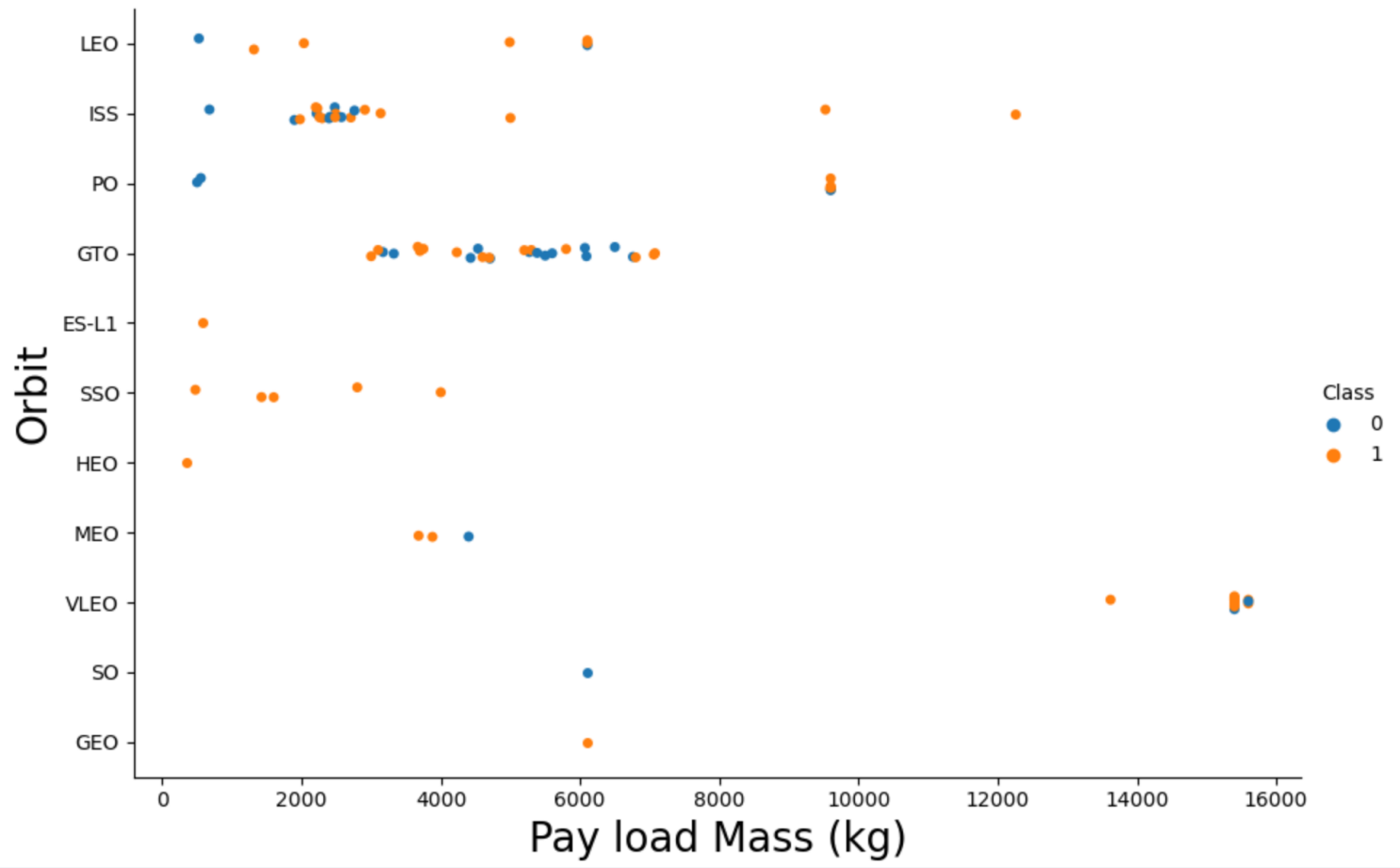
The highest success rate 100% is for ES-L1, GEO, HEO and SSO, 0% is for SO.

Flight Number vs. Orbit Type



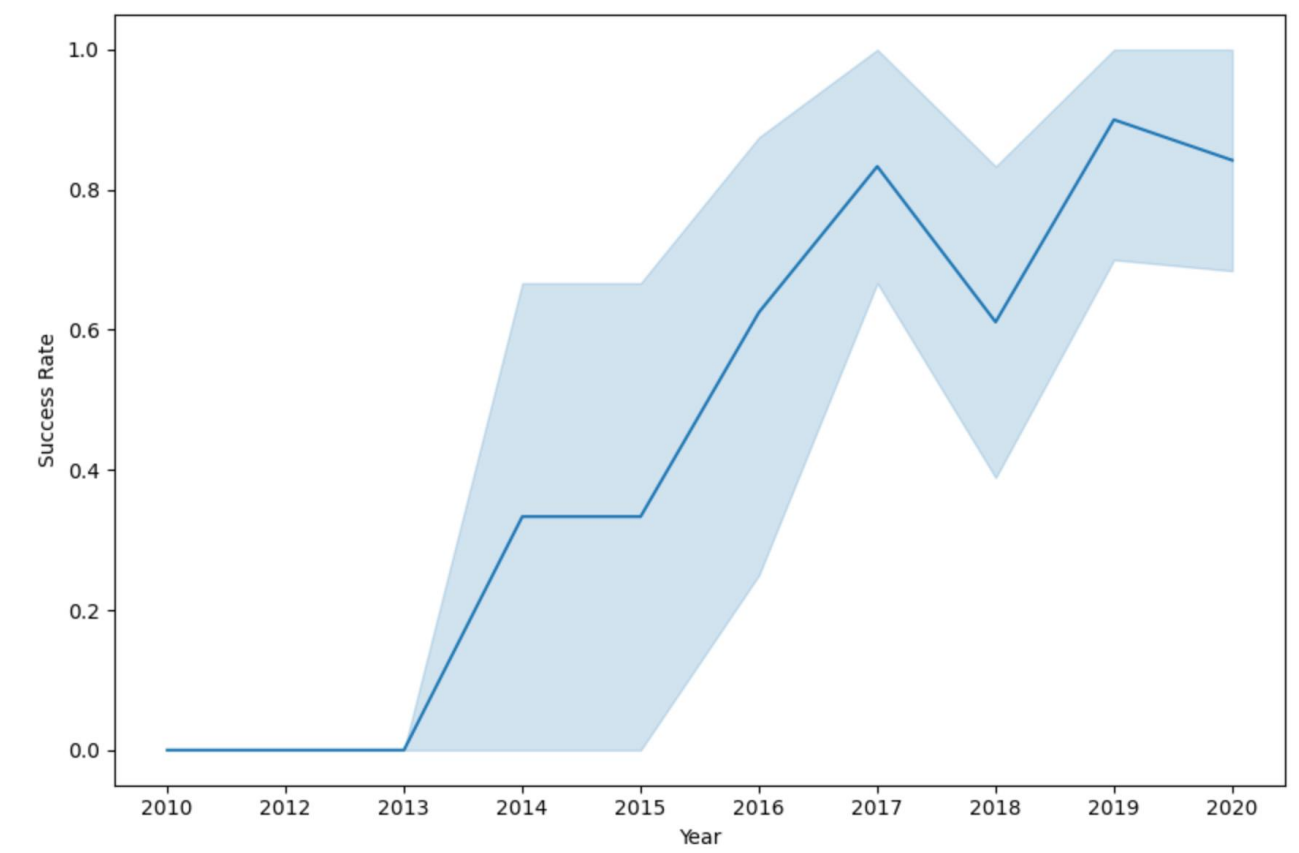
Orbit LEO: success rate is increasing with the number of flights.
Almost no dependence between Flight number and Orbit type for GTO.
All other orbits have increasing success rate with more flight numbers.

Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for PO, LEO and ISS.
Almost no dependence between Payload mass and Orbit type for GTO.

Launch Success Yearly Trend



Success rate is going up since 2013 with a temporary decrease in 2018.

All Launch Site Names

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

```
[7]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

```
[7]: ,,,,,,,,,,,,,,,,,,,,,,,,
```

Launch_Sites
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
None

Used ‘SELECT DISTINCT’ statement to return only the unique launch sites from the ‘LAUNCH_SITE’ column of the SPACEXTBL table

Launch Site Names Begin with 'CCA'

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

```
[8]: %sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

[8]: ''

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

Used ‘LIKE’ command with ‘WHERE’ clause to select and display a table of all records where launch sites begin with the string 'CCA'

Total Payload Mass

Task 3

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

```
[11]: %sql SELECT sum(payload_mass__kg_) as sum_payload from SPACEXTBL where (customer) = 'NASA (CRS)'
```

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

[11]:

sum_payload
45596.0

Used the ‘SUM()’ function to return and display the total sum of ‘PAYLOAD_MASS__KG_’ column for Customer ‘NASA (CRS)’

Average Payload Mass by F9 v1.1

Task 4

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

```
[33]: %sql SELECT avg(payload_mass__kg_) as average_payload from SPACEXTBL where (booster_version) = 'F9 v1.1'
```

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

[33]:

average_payload
2928.4

Used the ‘AVG()’ function to return and dispaly the average payload mass carried by booster version F9 v1.1

First Successful Ground Landing Date

```
[16]: %sql SELECT min(date) from SPACEXTBL where landing_outcome = 'Success (ground pad)'
```

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

[16]:

min(date)
01/08/2018

Used the ‘MIN()’ function to return and display the first (oldest) date when first successful landing outcome on ground pad ‘Success (ground pad)’ happened.

Successful Drone Ship Landing with Payload between 4000 and 6000

[34]:
```
%sql select BOOSTER_VERSION from SPACEXTBL where LANDING_OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000
```

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

[34]:

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

Used ‘Select’ statement to return and list of the names of boosters with operators >4000 and <6000 to only list booster with payloads between 4000-6000 with landing outcome of ‘Success (drone ship)’.

Total Number of Successful and Failure Mission Outcomes

```
[22]: %sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

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

[22]:

Mission_Outcome	Total
None	0
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Used the ‘COUNT()’ and ‘GROUP BY’ statement to return total number of Mission Outcomes

Boosters Carried Maximum Payload

```
[23]: %sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL);
 * sqlite:///my_data1.db
Done.
```

[23]:

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600.0
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600.0
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600.0
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600.0
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600.0
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600.0
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600.0
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600.0
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600.0
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600.0
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600.0
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600.0

Using a Subquery to return Max payload and list all the boosters that have carried the Max payload of 15600kgs

2015 Launch Records

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

[31]:
```
%sql SELECT substr(Date,7,4), substr(Date, 4, 2),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_", "Mission_Outcome", "Landing_Outcome" FROM SPACEXTBL WHERE substr(Date,7,4
```

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

[31]:

substr(Date,7,4)	substr(Date, 4, 2)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Mission_Outcome	Landing_Outcome
2015	10	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395.0	Success	Failure (drone ship)
2015	04	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898.0	Success	Failure (drone ship)

Used the ‘subsrt()’ in the select statement to get the month and year from the date column where substr(Date,7,4)='2015' for year and Landing_outcome was ‘Failure (drone ship’) and return the records matching the filter.

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

[30]:
```
%sql SELECT * FROM SPACEXTBL WHERE "Landing_Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-2017') ORDER BY Date DESC;
```

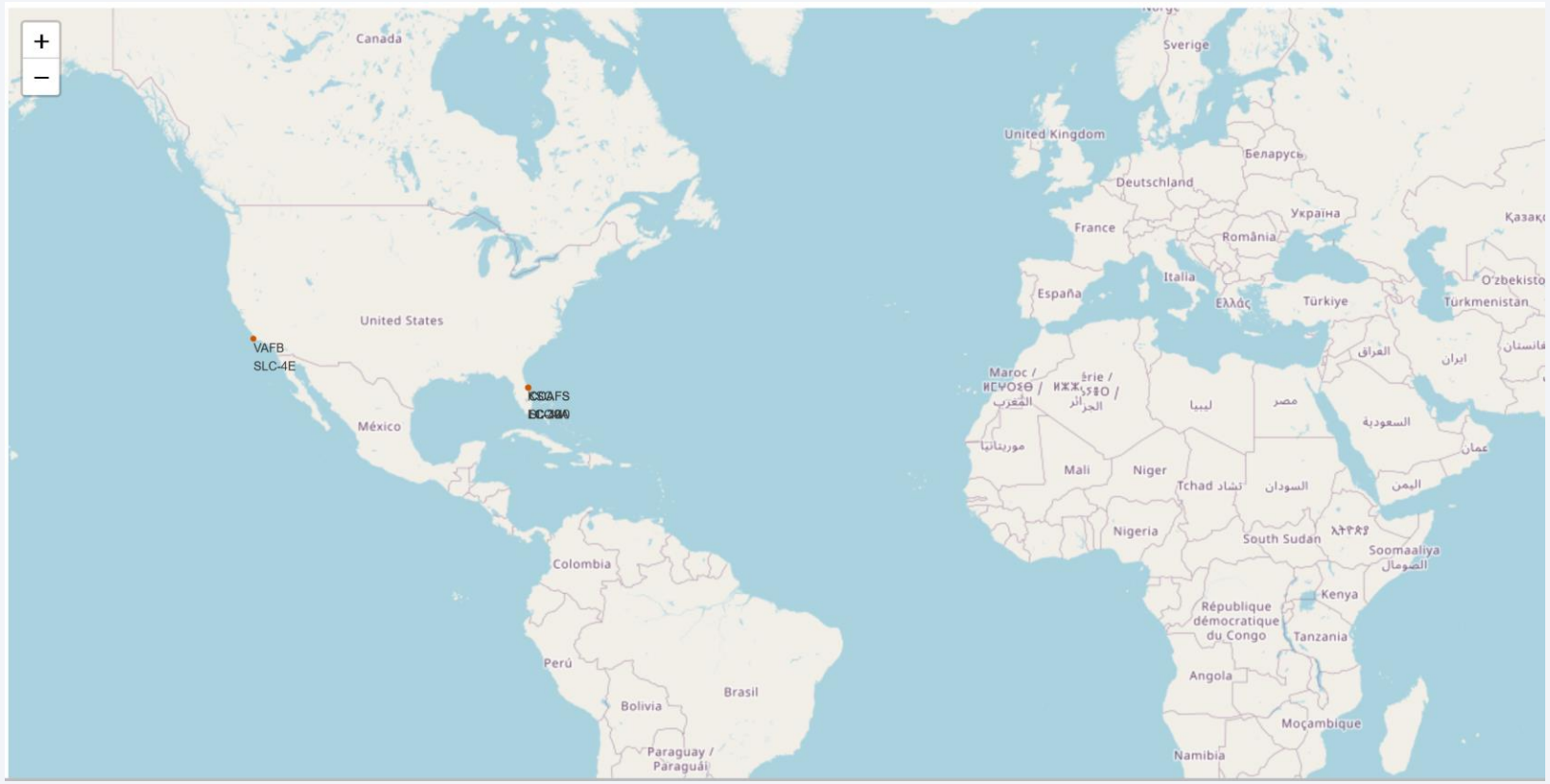
```
 * sqlite:///my_data1.db
Done.
```

[30]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
19/02/2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490.0	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
18/10/2020	12:25:57	F9 B5 B1051.6	KSC LC-39A	Starlink 13 v1.0, Starlink 14 v1.0	15600.0	LEO	SpaceX	Success	Success
18/08/2020	14:31:00	F9 B5 B1049.6	CCAFS SLC-40	Starlink 10 v1.0, SkySat-19, -20, -21, SAOCOM 1B	15440.0	LEO	SpaceX, Planet Labs, PlanetIQ	Success	Success
18/07/2016	4:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257.0	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
18/04/2018	22:51:00	F9 B4 B1045.1	CCAFS SLC-40	Transiting Exoplanet Survey Satellite (TESS)	362.0	HEO	NASA (LSP)	Success	Success (drone ship)
17/12/2019	0:10:00	F9 B5 B1056.3	CCAFS SLC-40	JCSat-18 / Kacific 1, Starlink 2 v1.0	6956.0	GTO	Sky Perfect JSAT, Kacific 1	Success	Success
16/11/2020	0:27:00	F9 B5B1061.1	KSC LC-39A	Crew-1, Sentinel-6 Michael Freilich	12500.0	LEO (ISS)	NASA (CCP)	Success	Success

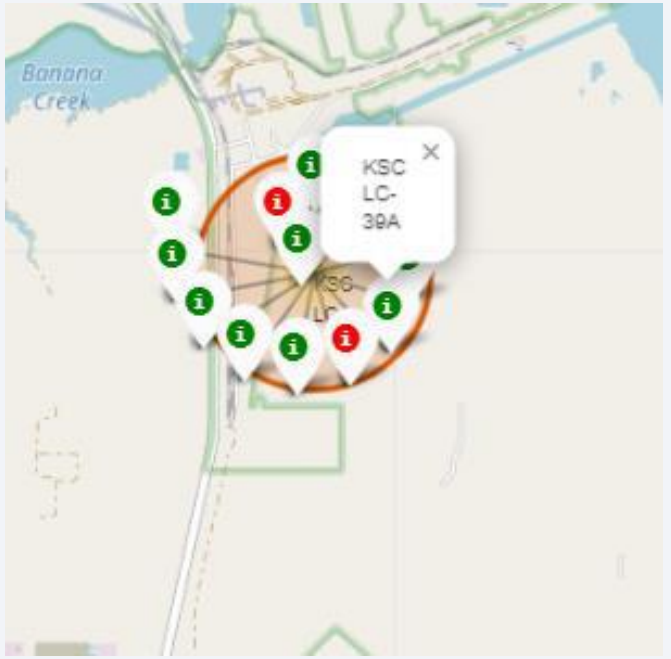

Section 3
Launch Sites
Proximities Analysis

All launch sites on global map



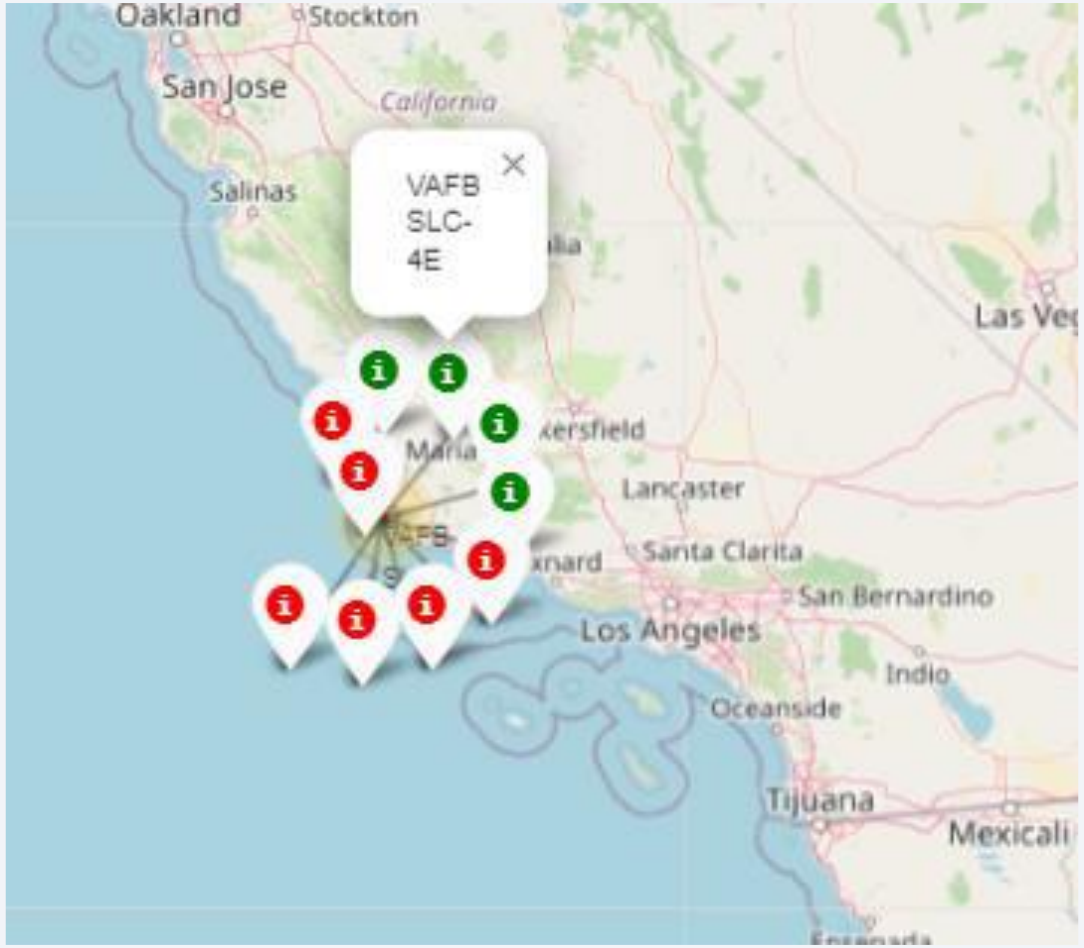
All launch sites are close to the Equator and to the coast (California and Florida).

Launch outcomes for each site



In the Eastern coast (Florida) Launch site KSC has relatively high success rates compared to CCAFS SLC-40 & CCAFS LC-40.

Launch outcomes for each site

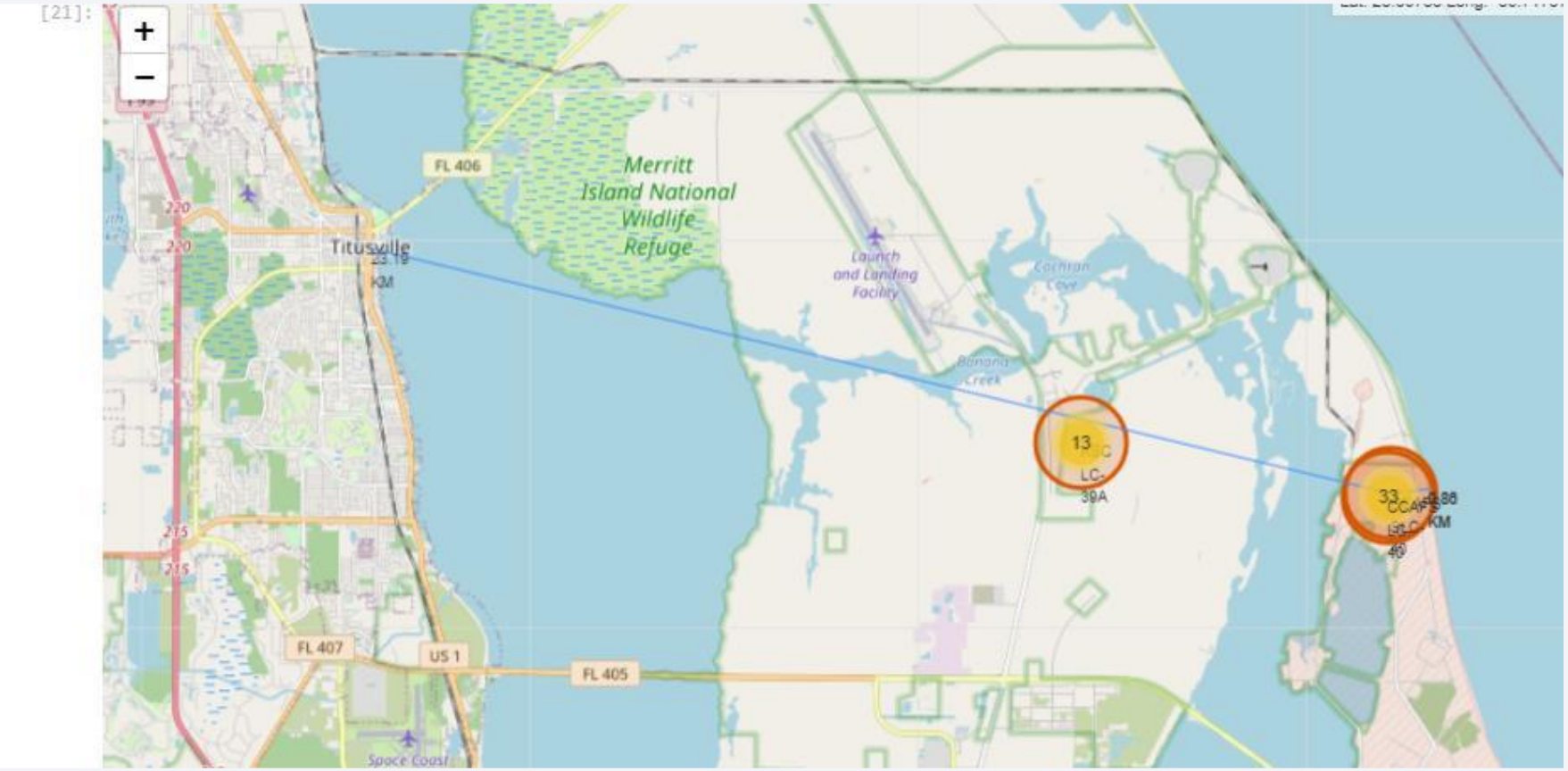


Californian launch site VAFB SLC-4E has lower success rate 25% compared to KSC LC-39A in Florida.

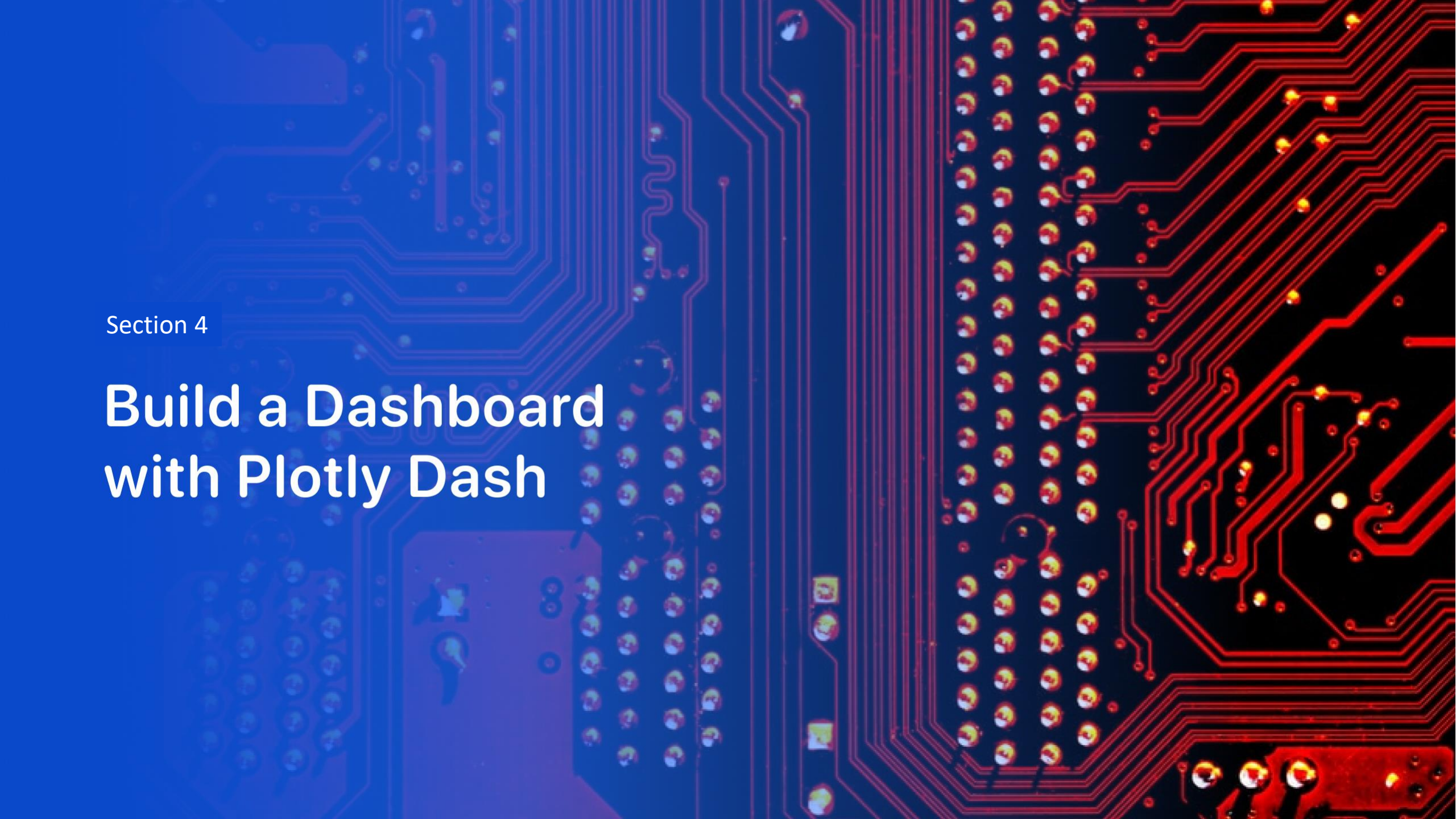
Distances between a launch site to its proximities

Launch site CCAFS SLC-40 proximity to coastline is 0.86km

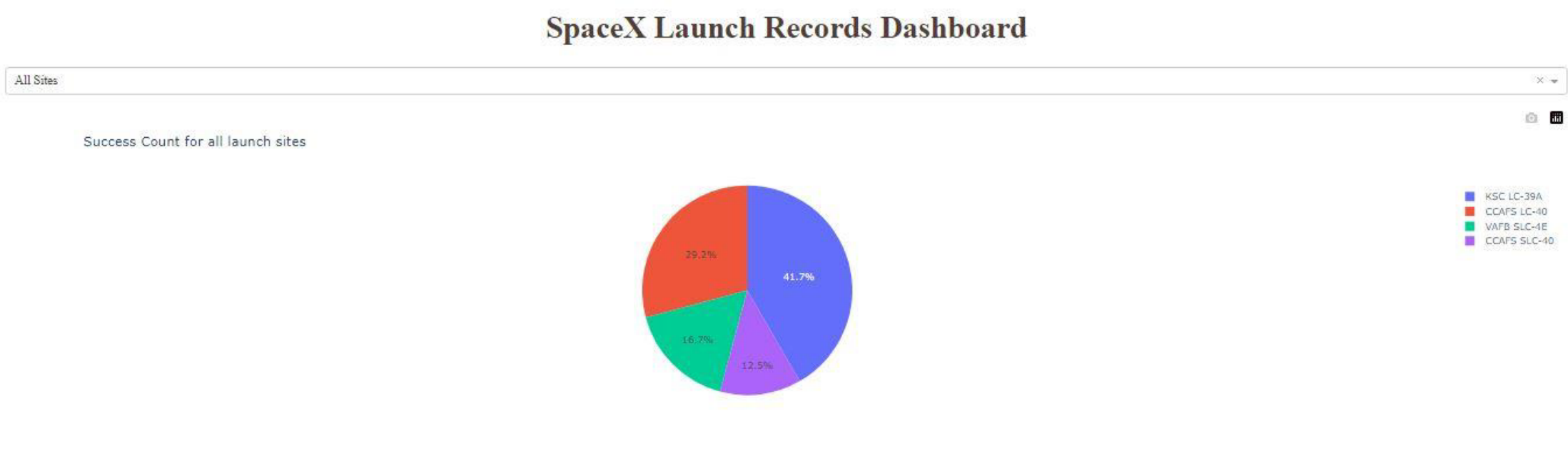
Distances between a launch site to its proximities



Launch site CCAFS SLC-40closest to highway (Washington Avenue) is 23.19km

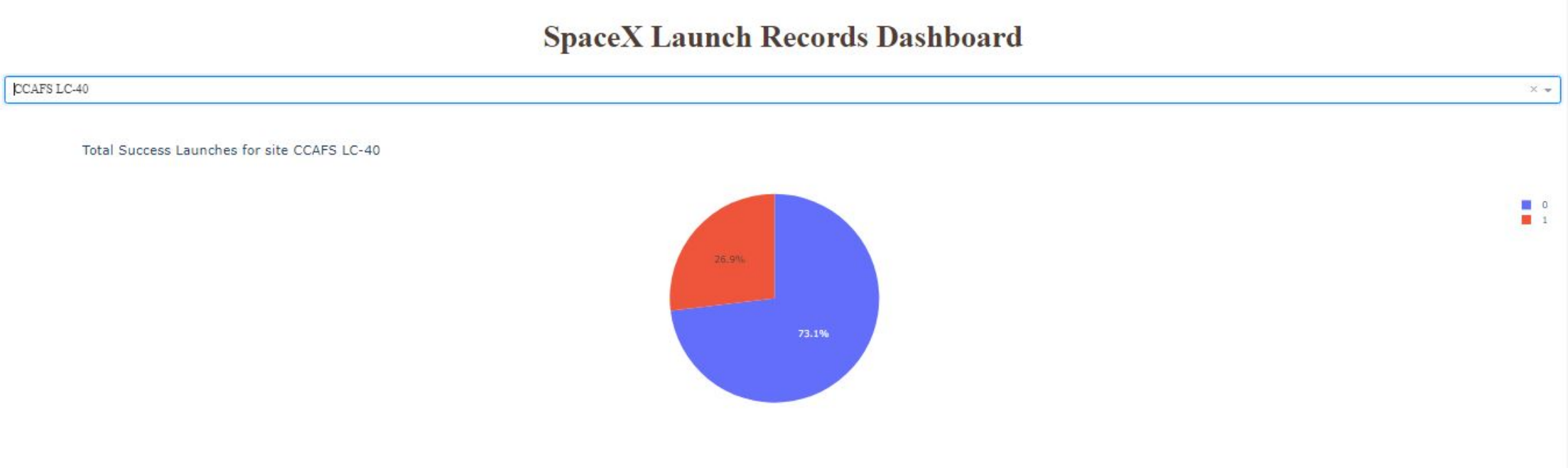

Section 4
Build a Dashboard with Plotly Dash

Pie-Chart for launch success count for all sites



Launch site KSC LC-39A has the highest launch success rate 42% followed by CCAFS LC-40 at 29%, VAFB SLC-4E at 17% and lastly launch site CCAFS SLC-40 with a success rate of 13%

Pie chart for CCAFS LC-40 site



Launch site CCAFS LC-40 has the 2nd highest success ratio of 73%

Payload vs. Launch Outcome scatter plot for all sites

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

Section 5

Predictive Analysis (Classification)

Classification Accuracy

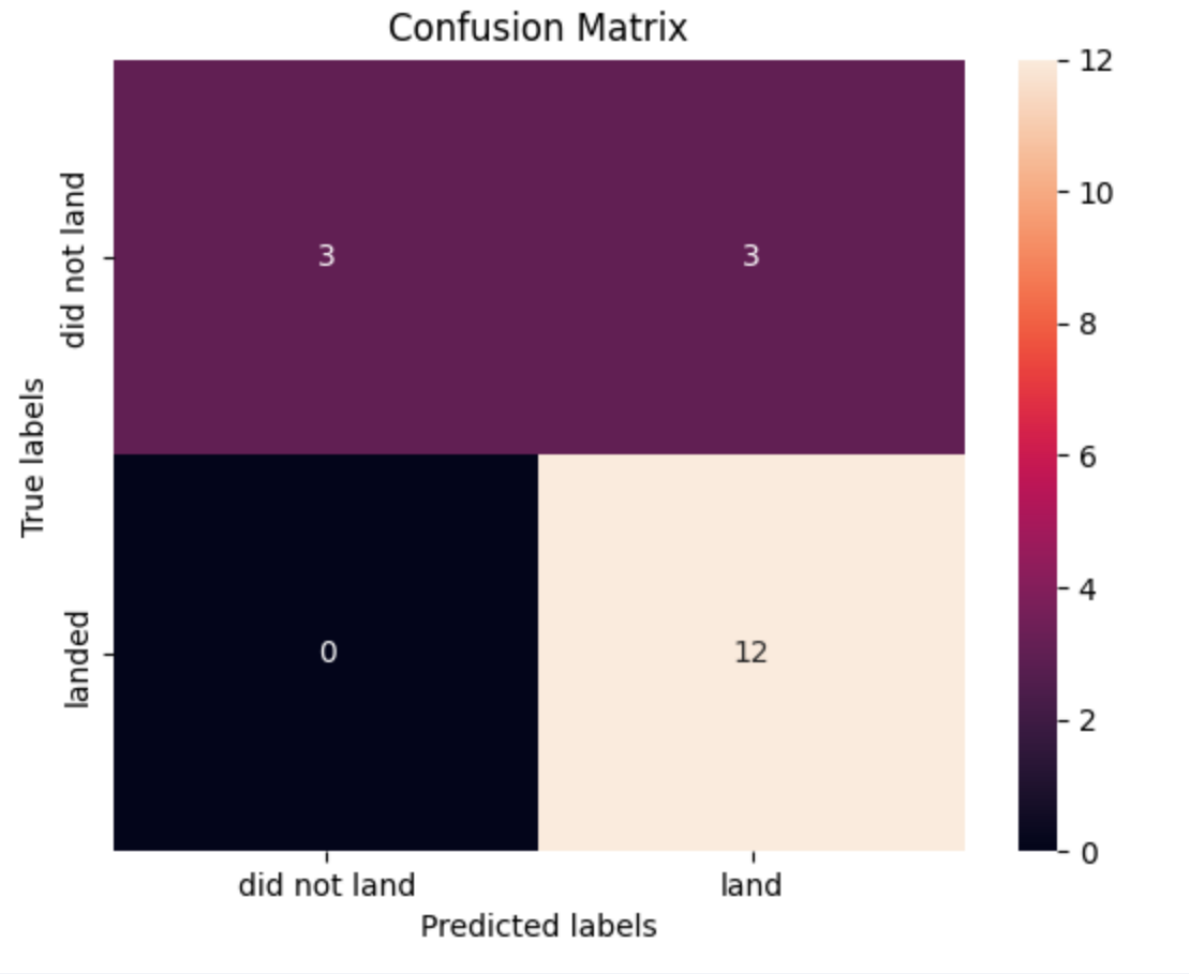
[32]:
```
Report = pd.DataFrame({'Method' : ['Test Data Accuracy']})

knn_accuracy=knn_cv.score(X_test, Y_test)
Decision_tree_accuracy=tree_cv.score(X_test, Y_test)
SVM_accuracy=svm_cv.score(X_test, Y_test)
Logistic_Regression=logreg_cv.score(X_test, Y_test)

Report['Logistic_Reg'] = [Logistic_Regression]
Report['SVM'] = [SVM_accuracy]
Report['Decision Tree'] = [Decision_tree_accuracy]
Report['KNN'] = [knn_accuracy]

Report.transpose()
```

[32]:

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

All the methods with the same accuracy – 0.833333

Confusion Matrix



All the classification models have the same confusion matrices however major problem is false positives for all of those.

Conclusions

- Different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.
- Success rate and Flight number increase for each launching site. For CCAFS 100% successful flights after 75 launches, for VAFB – after 50 flights, for KSC – after 80 launches
- If you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for high payload mass (>10000).
- The highest success rate 100% is for ES-L1, GEO, HEO and SSO, the lowest – 0% is for SO.
- 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
- With heavy payloads the successful landing or positive landing rate are more for PO, 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
- Success rate is going up since 2013 with a temporary decrease in 2018.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!