IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Márk Iván
28.03.2024.

Outline

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

Executive Summary

- ## Summary of methodologies
- Collection of SpaceX data via the SpaceX API.
- Gathering SpaceX data through web scraping techniques.
- Organizing and preparing SpaceX data for analysis.
- Exploratory data analysis of SpaceX data using SQL.
- Performing exploratory data analysis on SpaceX data using Python Pandas and Matplotlib.
- Analyzing SpaceX launch sites using Folium for interactive visual analytics and Plotly Dash.
- Employing machine learning techniques to predict SpaceX landing outcomes.
- ## Summary of all results
- Summary of the exploratory data analysis outcomes.
- Creation of interactive visual analytics and dashboards.
- Application of predictive analysis techniques, specifically classification, to SpaceX data.

Introduction

- ## Project background and context

  SpaceX advertises Falcon 9 rocket launches at 62 million dollars, significantly undercutting competitors who charge upwards of 165 million dollars due to their reusable first stage. This cost advantage hinges on the successful landing of the first stage, crucial information for potential competitors seeking to bid against SpaceX.

- ## Problems you want to find answers

  Our objective is to forecast the successful landing of the Falcon 9 first stage.

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

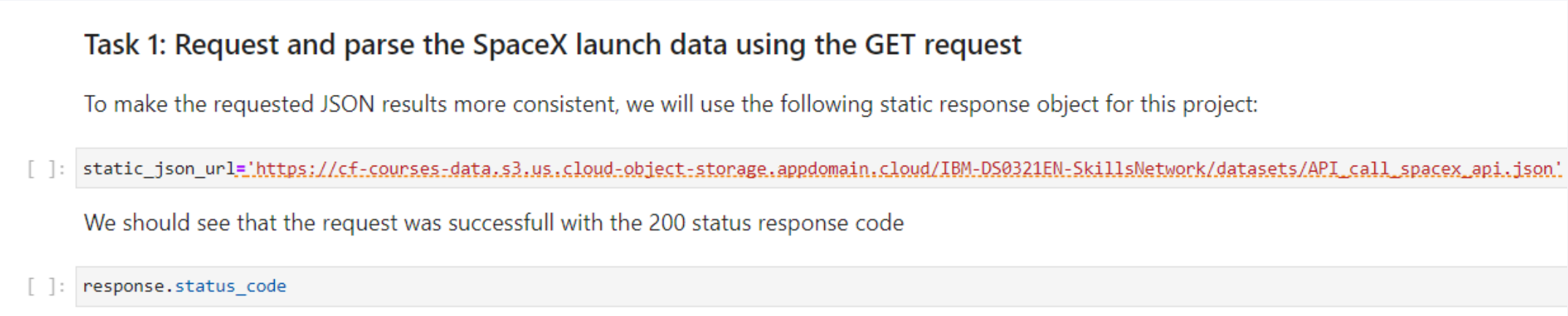
Data Collection

- Describe how data sets were collected.

- Data collection began by utilizing the SpaceX API, a RESTful API, involving the creation of helper functions to extract launch information and subsequent retrieval of rocket launch data via GET requests to the SpaceX API URL. The obtained JSON results were then converted into a Pandas dataframe for further analysis.
- In order to ensure uniformity in the collected JSON data, the SpaceX launch data was fetched and parsed through GET requests, followed by decoding the response content as JSON results. This facilitated the conversion of the retrieved data into a structured Pandas dataframe format for streamlined processing.
- Web scraping techniques were applied to procure historical launch records of Falcon 9 from a Wikipedia page titled "List of Falcon 9 and Falcon Heavy launches". Utilizing BeautifulSoup and request Libraries, the Falcon 9 launch HTML table records were extracted, parsed, and transformed into a Pandas dataframe, enabling comprehensive analysis of the gathered data.

Data Collection – SpaceX API

- Information was gathered through the SpaceX API (a RESTful API) , employing GET requests to retrieve data, which was subsequently parsed. Following the retrieval process, the response content was decoded into JSON format, then transformed into a Pandas dataframe for analysis.



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:

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

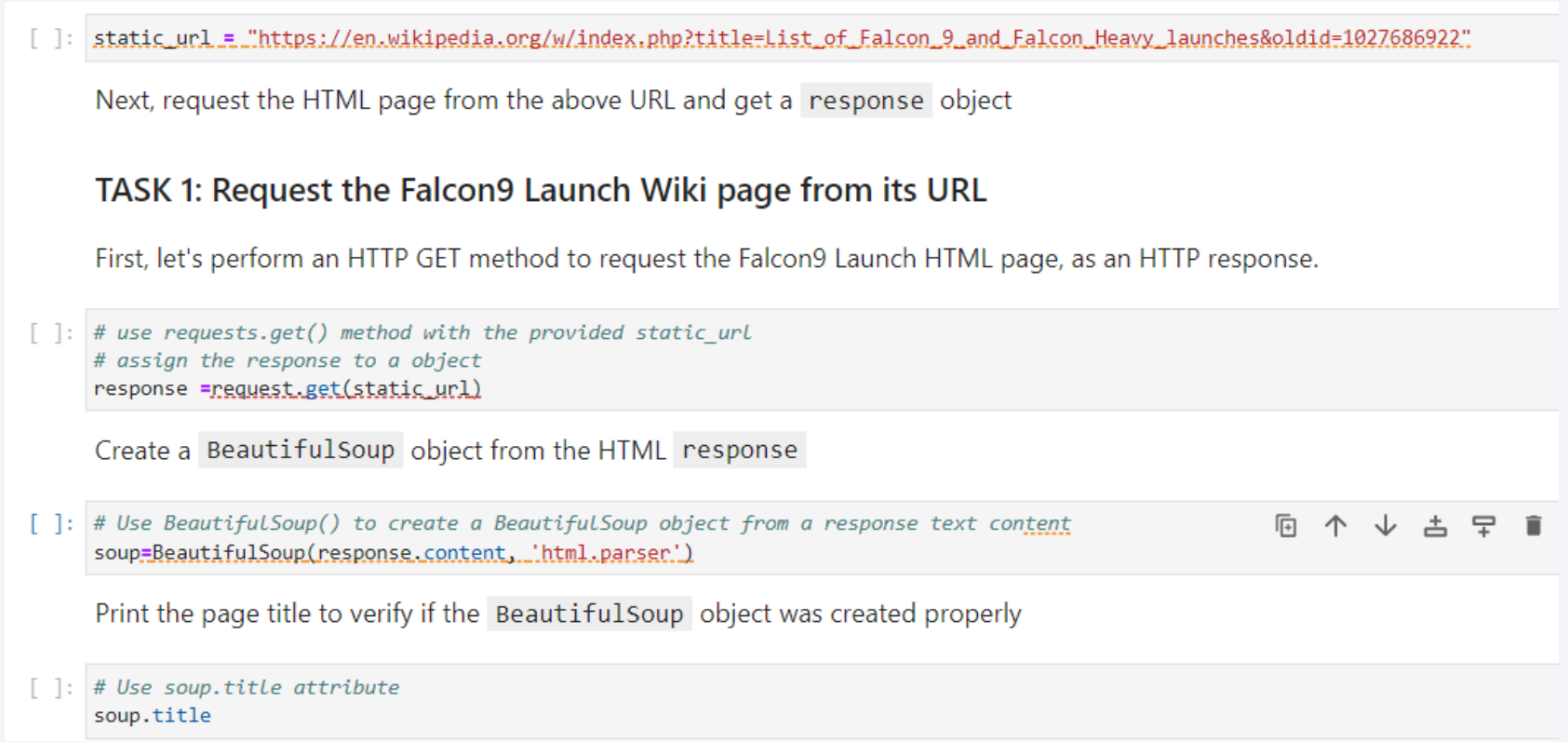
We should see that the request was successfull with the 200 status response code

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

- GitHub URL: https://github.com/1IvanMark/SpaceX_Falcon9/blob/main/1.ipynb

Data Collection - Scraping

- Utilized web scraping with BeautifulSoup and requests to gather historical Falcon 9 launch records from a Wikipedia page's HTML table. The extracted data was then parsed and organized into a dataframe for analysis.



GitHub URL: https://github.com/1IvanMark/SpaceX_Falcon9/blob/main/2.ipynb

Data Wrangling

- Following the acquisition and establishment of a Pandas DataFrame from the gathered dataset, filtration was applied based on the BoosterVersion column to exclusively retain Falcon 9 launches. Subsequently, attention was directed towards addressing missing data values in the LandingPad and PayloadMass columns. To handle missing values in the PayloadMass column, the mean value of the column was utilized for replacement.
- GitHub URL:

https://github.com/1IvanMark/SpaceX_Falcon9/blob/main/3.ipynb

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding r
to the variable `landing_class`:

```
# 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()
```

```
1    60
0    30
Name: Class, dtype: int64
```

This variable will represent the classification variable that represents the outcome
one means the first stage landed Successfully

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

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

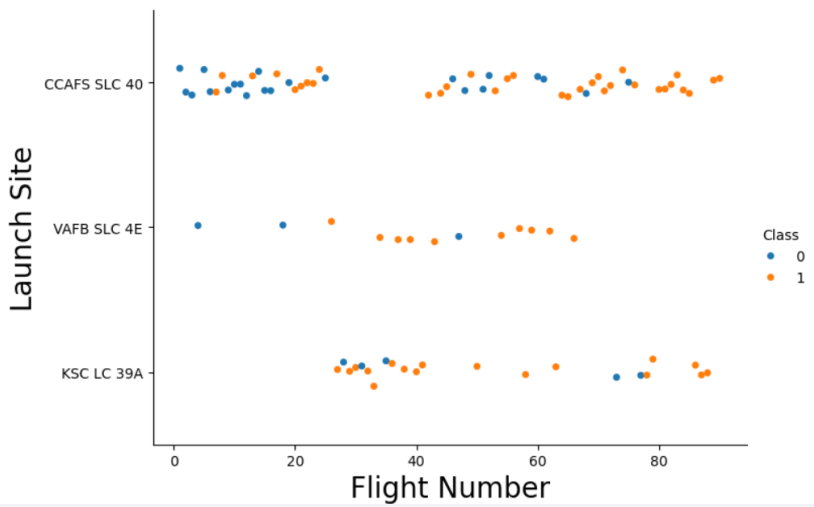
EDA with Data Visualization

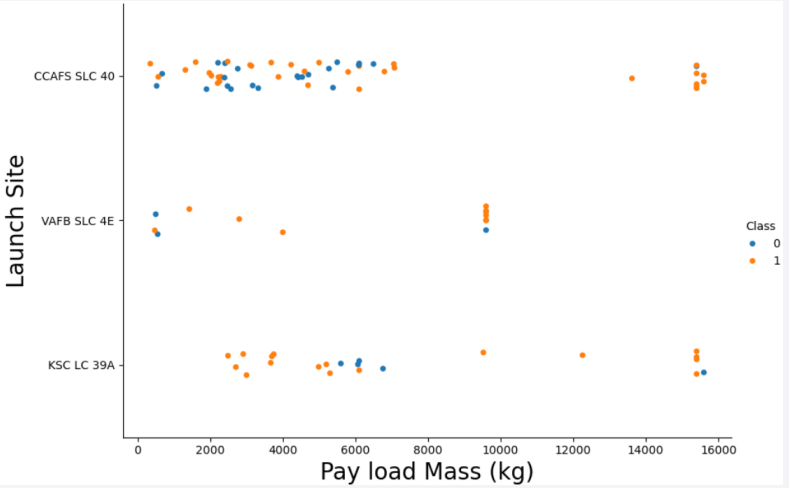
Conducted data analysis and feature engineering tasks utilizing Pandas and Matplotlib, including:

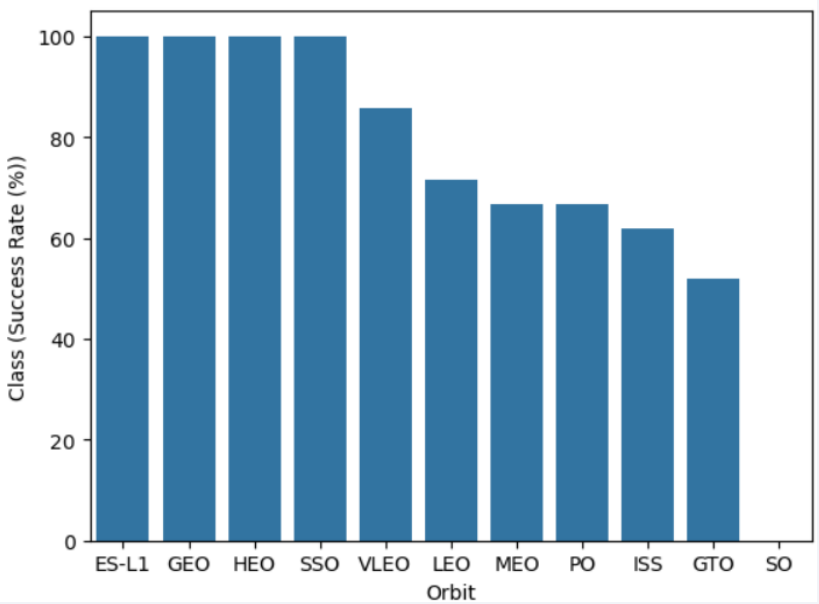
- Exploratory Data Analysis
- Preparation of data through feature engineering techniques
- Employed scatter plots to visualize correlations between Flight Number and Launch Site, Payload and Launch Site, Flight Number and Orbit type, as well as Payload and Orbit type.
- Utilized bar charts to illustrate the success rate of each orbit type.
- Employed line plots to visualize the yearly trend in launch success.

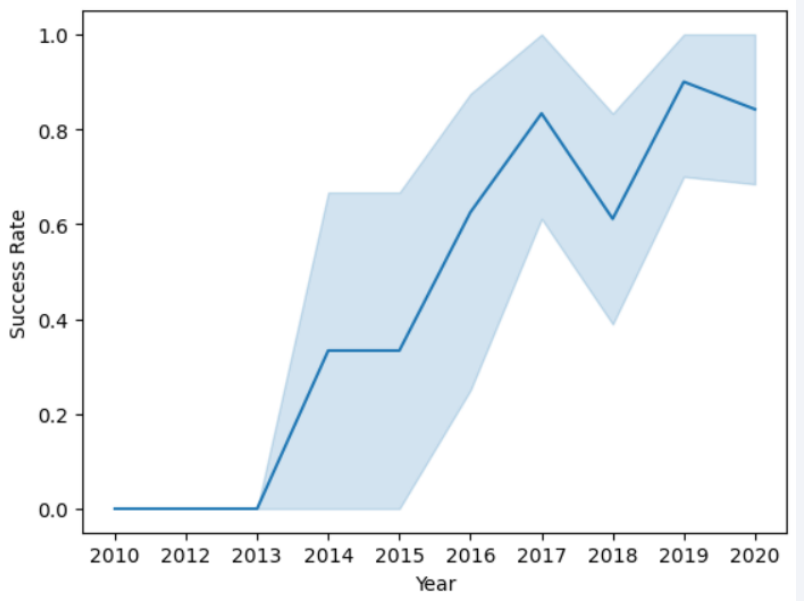
- GitHub URL: https://github.com/1IvanMark/SpaceX_Falcon9/blob/main/5.ipynb

EDA with Data Visualization (continue)









EDA with SQL

During EDA, the below listed SQL queries were executed:

- Display the names of the unique launch sites in the space mission: `%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;`
- Display 5 records where launch sites begin with the string 'CCA‘: `%sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5;`
- Display the total payload mass carried by boosters launched by NASA (CRS) `%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';`
- Display average payload mass carried by booster version F9 v1.1: `%sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';`
- List the date when the first successful landing outcome in ground pad was achieved: `%sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing _Outcome" = "Success (ground pad)";`
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000: `%sql SELECT DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing _Outcome" = "Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;`
- List the total number of successful and failure mission outcomes: `%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";`

GitHub URL: https://github.com/1IvanMark/SpaceX_Falcon9/blob/main/4.ipynb

Build an Interactive Map with Folium

- Developed a Folium map to pinpoint all launch sites, incorporating map elements like markers, circles, and lines to indicate the success or failure of launches at each site.
- Established a launch outcome dataset, where failure is represented as 0 and success as 1.

GitHub URL: https://github.com/1IvanMark/SpaceX_Falcon9/blob/main/6.ipynb

Build a Dashboard with Plotly Dash

- Constructed an interactive dashboard application using Plotly Dash by:
- Integrating a Launch Site Drop-down Input Component
- Implementing a callback function to display a success-pie-chart based on the selected site from the dropdown menu
- Incorporating a Range Slider for Payload Selection
- Implementing a callback function to visualize the success-payload-scatter-chart scatter plot based on the selected payload range.

- GitHub URL: https://github.com/1IvanMark/SpaceX_Falcon9/blob/main/7.ipynb

Predictive Analysis (Classification) 1/3.

Here is a list of stepf of the process the classification model:

- Initially, the data was loaded into a Pandas DataFrame.
- Exploratory Data Analysis (EDA) to gain insights into the dataset.
- The training labels were determined by creating a NumPy array from the 'Class' column in the data, utilizing the to_numpy() method, and assigning it to the variable Y as the outcome variable.
- The feature dataset (x) was standardized by applying the preprocessing.StandardScaler() function from Sklearn.
- Subsequently, the data was split into training and testing sets using the train_test_split function from sklearn.model_selection, with the test_size parameter set to 0.2 and random_state set to 2 for reproducibility purposes.
- To determine the best ML model/method for optimal performance using the test data among SVM, Classification Trees, k-nearest neighbors, and Logistic Regression, the following steps were undertaken:
- Initialized objects for each algorithm and subsequently created a GridSearchCV object, assigning them respective sets of parameters.
- For each model under evaluation, a GridSearchCV object was created with cv=10, and the training data was fitted into each GridSearch object to find the best hyperparameters.
- Upon fitting the training set, the GridSearchCV object for each model was outputted, and the best parameters were displayed using the data attribute best_params_. Additionally, the accuracy on the validation data was displayed using the data attribute best_score_.
- Finally, the accuracy on the test data for each model was calculated using the score method, and a confusion matrix was plotted for each using the test and predicted outcomes.
- Below is a comparison of the test data accuracy scores for each method, showcasing the best-performing model among SVM, Classification Trees, k-nearest neighbors, and Logistic Regression:
- Based on the test data accuracy scores, k-nearest neighbors outperformed the other methods with a score of 0.87, indicating its superior performance in this scenario.

Method	Test Data Accuracy Score
SVM	0.85
Classification Trees	0.82
k-nearest neighbors	0.87
Logistic Regression	0.83

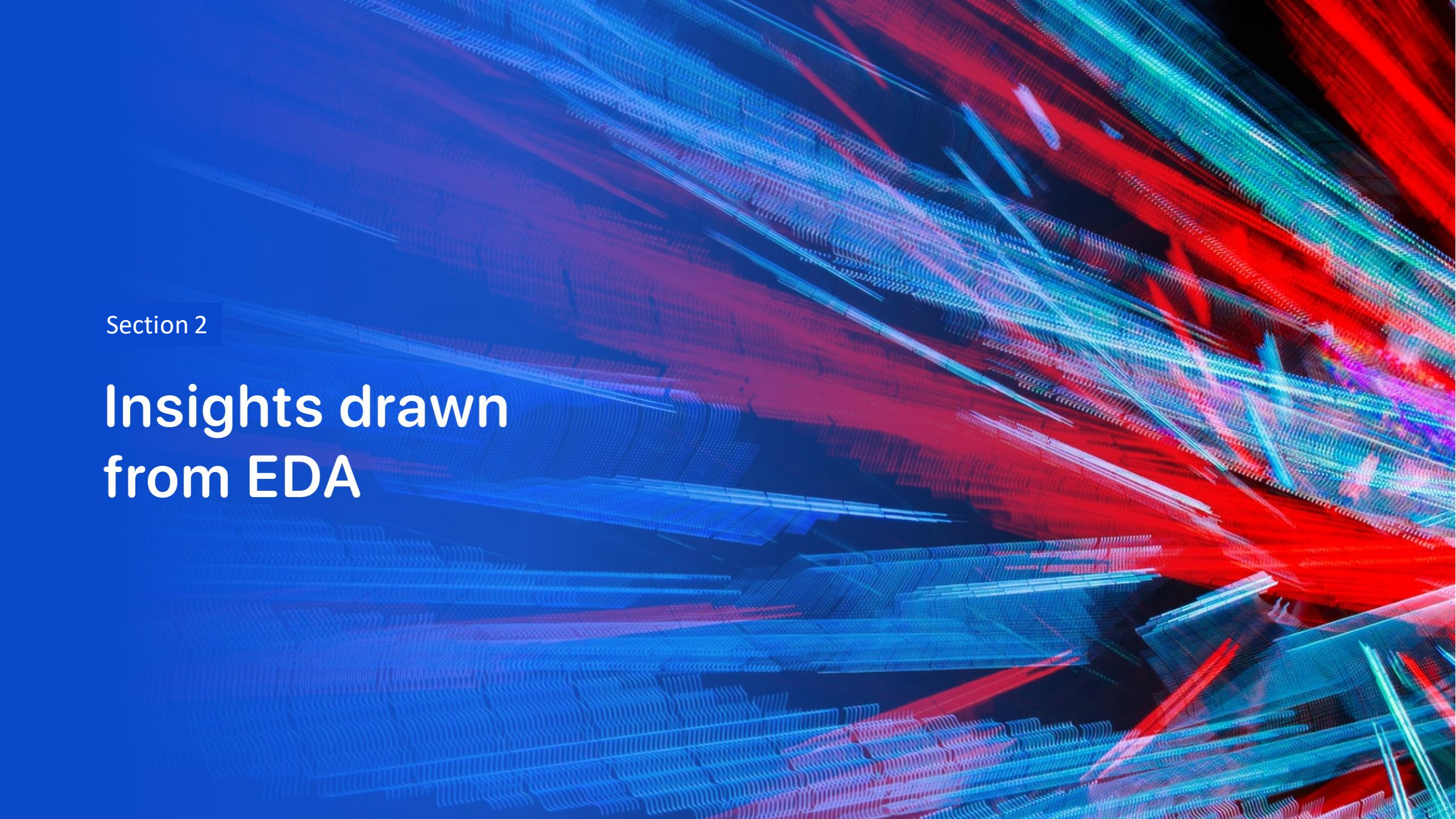
- GitHub
  URL: https://github.com/1IvanMark/SpaceX_Falcon9/blob/main/8.ipynb

Results

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

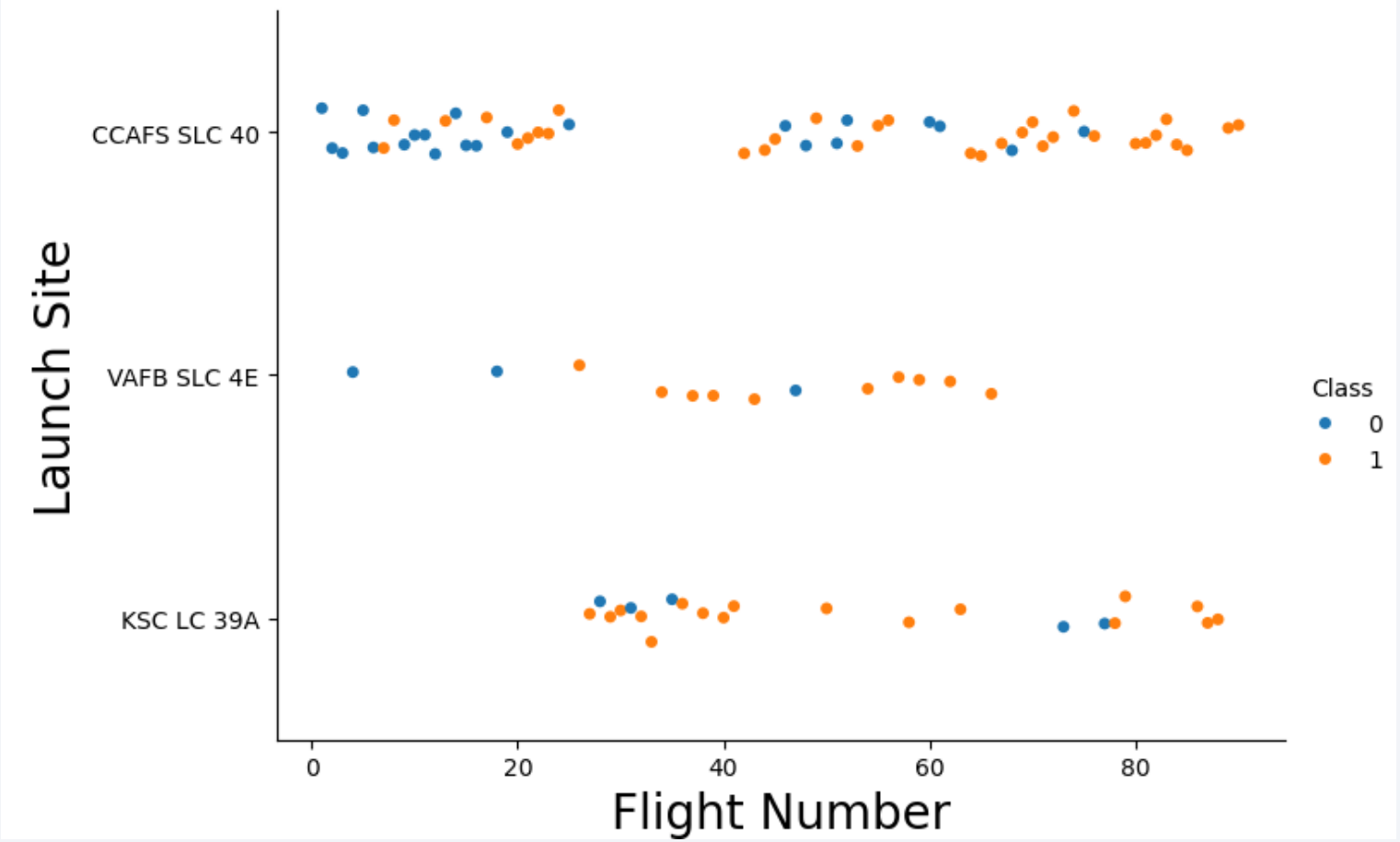
Section 2

Insights drawn from EDA



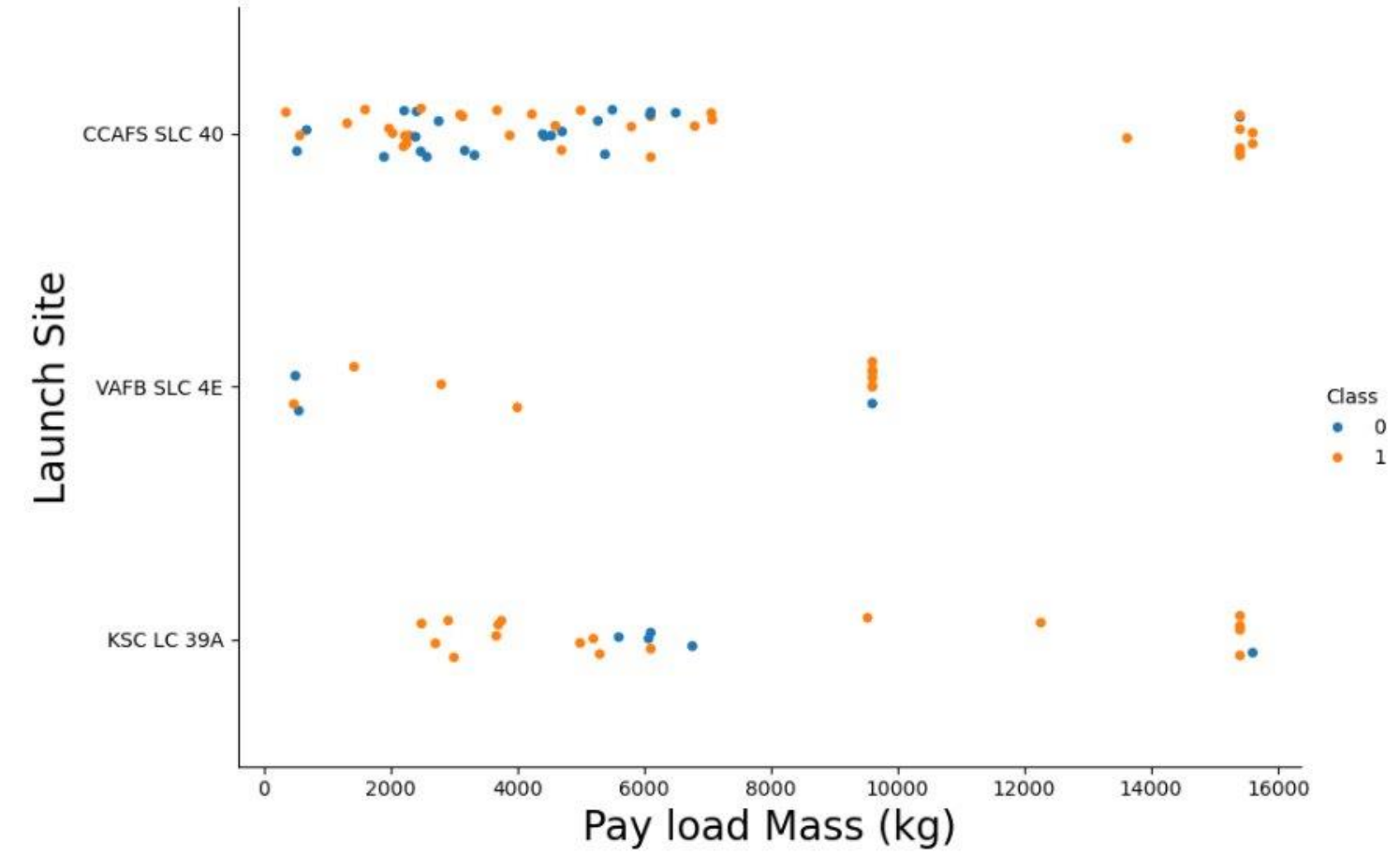
Flight Number vs. Launch Site

As flight number increases in each the launch places the success rate also.



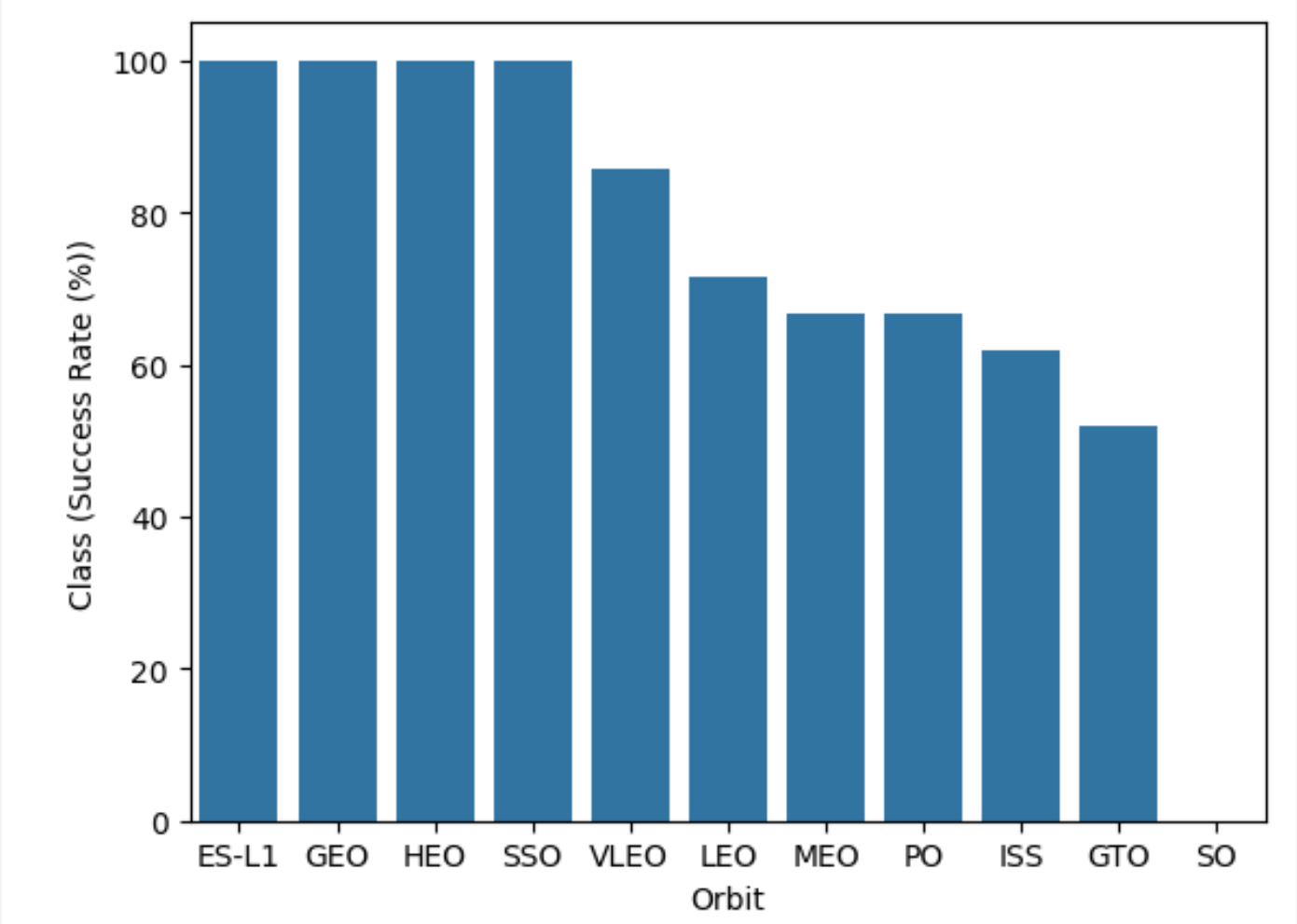
Payload vs. Launch Site

- In launch place of VAFB-SLC there is no heavypayload mass >10000 kg.



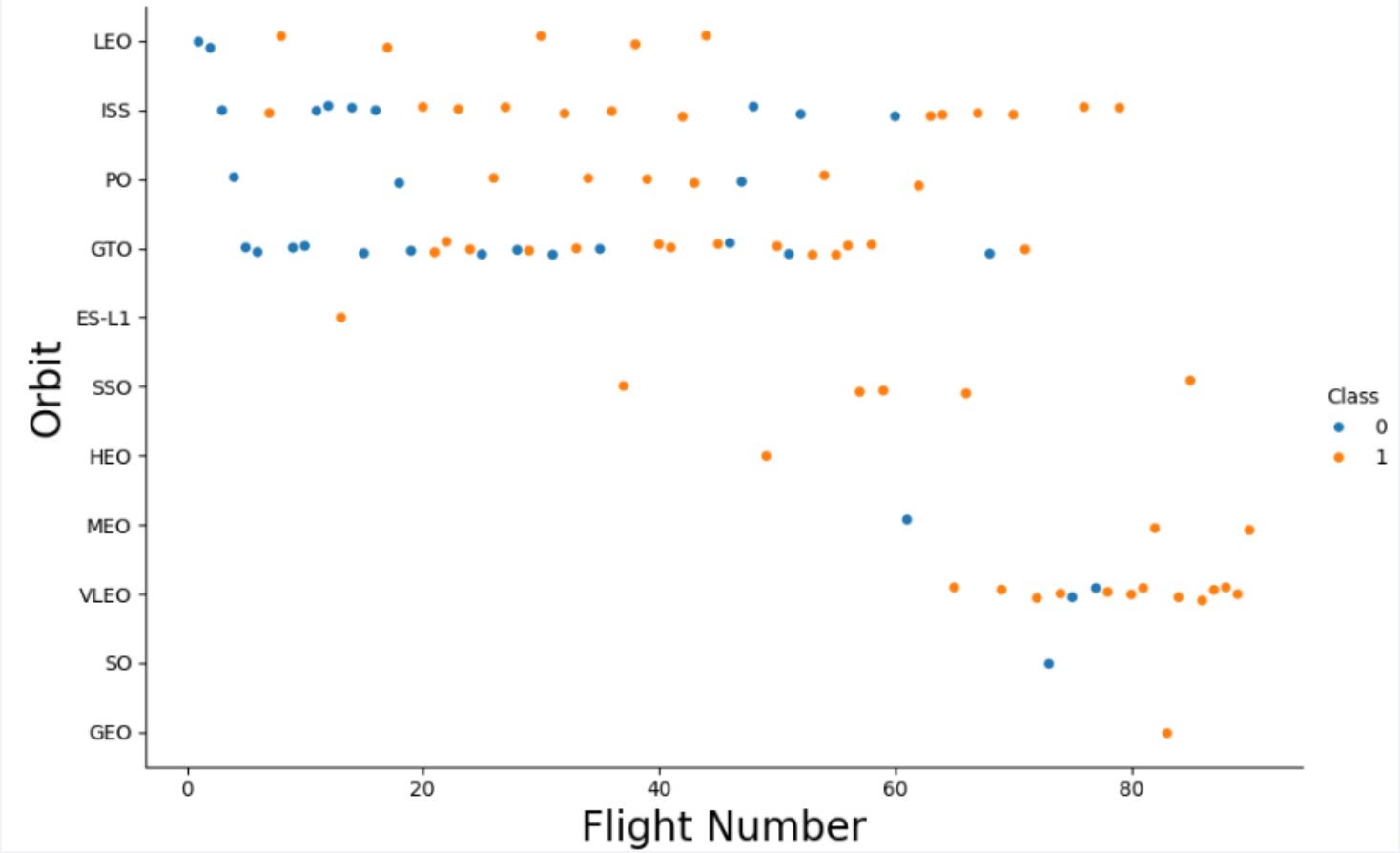
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, SSO with 100%, while on the less percent with 50% GTO is.



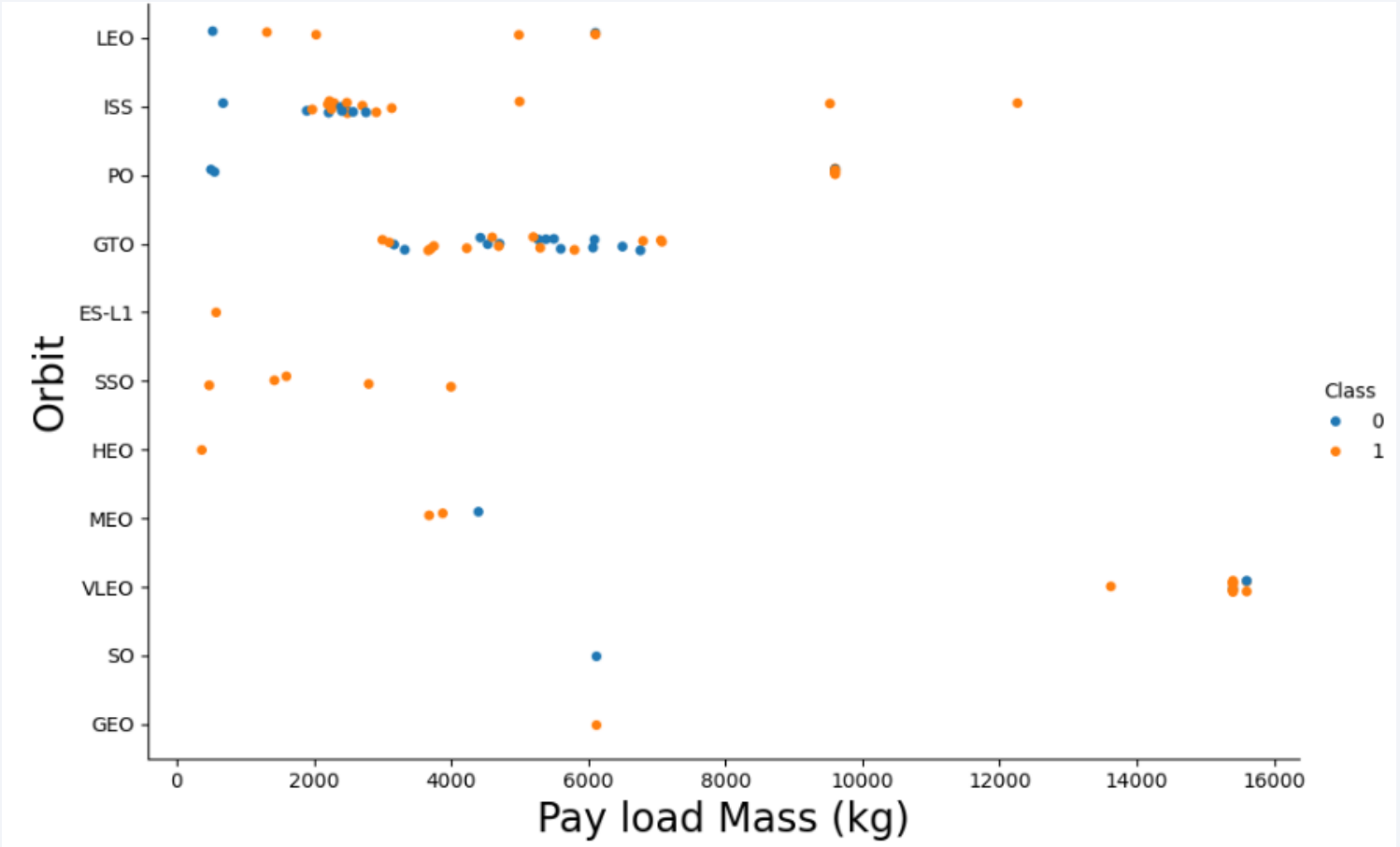
Flight Number vs. Orbit Type

- Comparing LEO and GTO there are differneces. Possible to see relationship in LEO between success and number of flights.



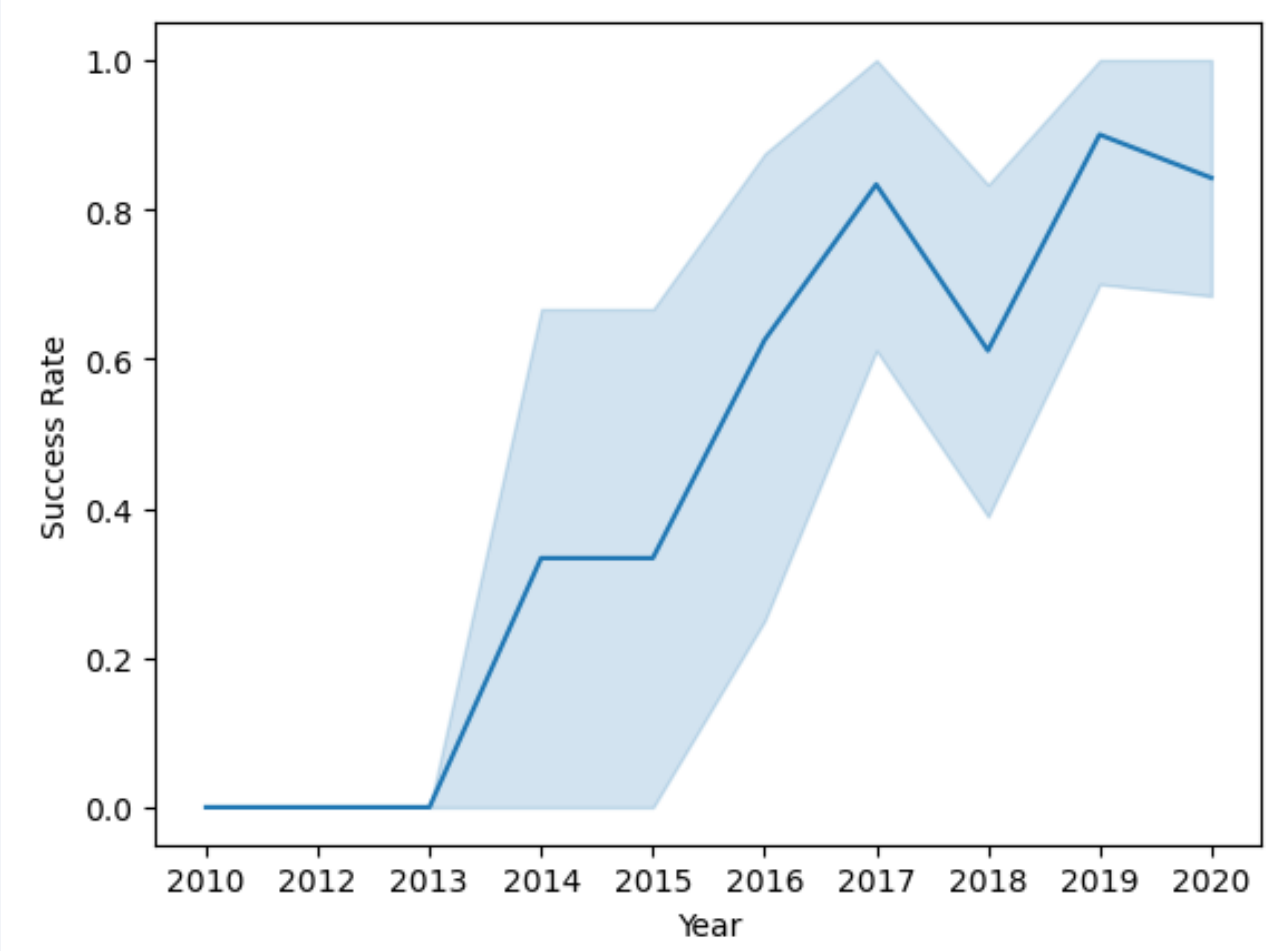
Payload vs. Orbit Type

- From the heavy payloads perspective Polar, LEO and ISS the rate of positive landing is huge. In GTO the the result is different.



Launch Success Yearly Trend

- Huge increase shown in the graph from 2013 to 2017 then a short decrease and increase phases. The last year from 2019 to 2020 little bit decrease.



All Launch Site Names

- The key point was to use the DISTINCT expression. Result shows the Launch sites/locations.

```
In [8]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
 * sqlite:///my_data1.db
Done.
```

Out[8]:

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

Launch Site Names Begin with 'CCA'

- The crucial point was to use launch_site LIKE 'CCA%'.

In [9]:
```
%sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Out[9]:

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

The key point: using the 'SUM()' function.

```
In [10]: %sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';
 * sqlite:///my_data1.db
Done.
```

Out[10]:

Total Payload Mass(Kgs)	Customer
45596	NASA (CRS)

Average Payload Mass by F9 v1.1

The key point: using the 'AVG()' function.

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

In [11]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version
```

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

Out[11]:

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

First Successful Ground Landing Date

- The key point was to use the ‘SUM()’ function

```
In [12]: %sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing _Outcome" = "Success (ground pad)";
 * sqlite:///my_data1.db
Done.
Out[12]: MIN(DATE)
         None
```

Successful Drone Ship Landing with Payload between 4000 and 6000

The key points was to use 'AND' and 'DISTINCT' expressions.

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

In [13]:
```
%sql SELECT DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing _Outcome" = "Success (drone ship)" AND PAYLOAD_
```

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

Out[13]:

Booster_Version	Payload

Total Number of Successful and Failure Mission Outcomes

The key point was to use the 'COUNT()' and 'GROUP BY' expressions.

List the total number of successful and failure mission outcomes

In [14]:
```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

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

Out[14]:

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

Boosters Carried Maximum Payload

- The point was to use subquerry and 'MAX()' function.

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

In [15]:
```
%sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_M
```

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

Out[15]:

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

2015 Launch Records

- The key point was to use 'substr()' function.

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.

In [16]:
```
%sql SELECT substr(Date,7,4), substr(Date, 4, 2),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_", "Mission_Ou
```

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

Out[16]:

substr(Date,7,4)	substr(Date, 4, 2)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Mission_Outcome	"Landing _Outcome"

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

- The key point was to use the ORDER BY expression.

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 [17]:
```
%sql SELECT * FROM SPACEXTBL WHERE "Landing _Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-2017') ORDEI
```

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

Out[17]:

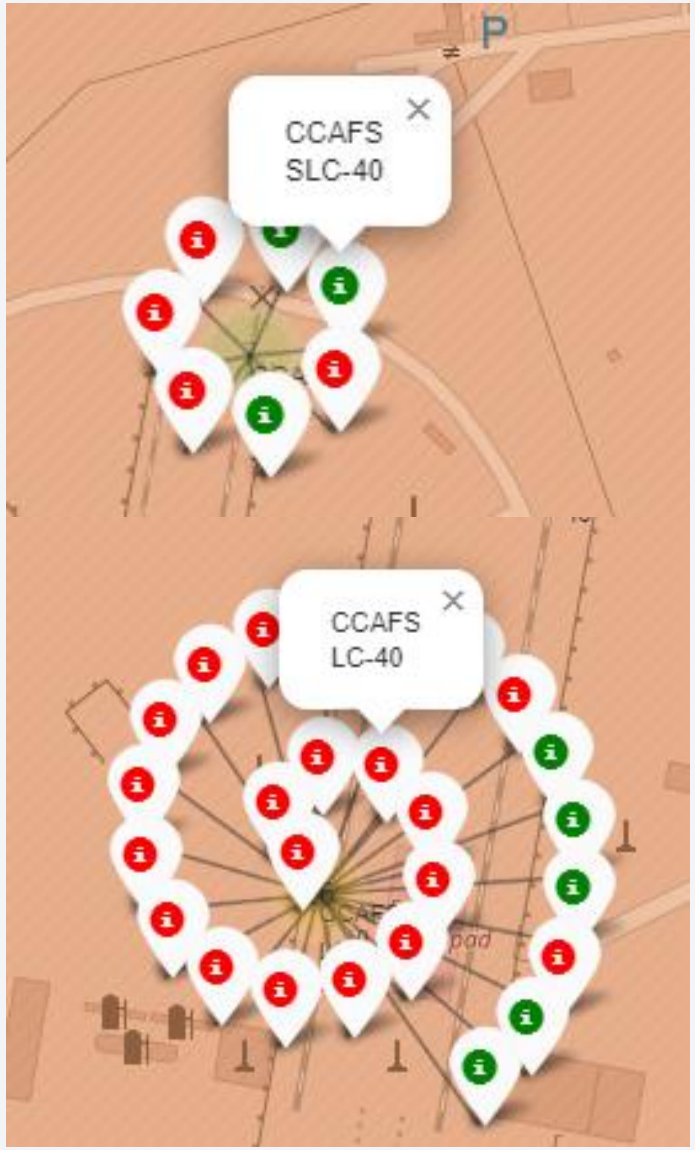
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome


Section 3
Launch Sites
Proximities Analysis

Launch sites in global map

- All launch sites are l the northern hemisphere in the southern part of the USA.

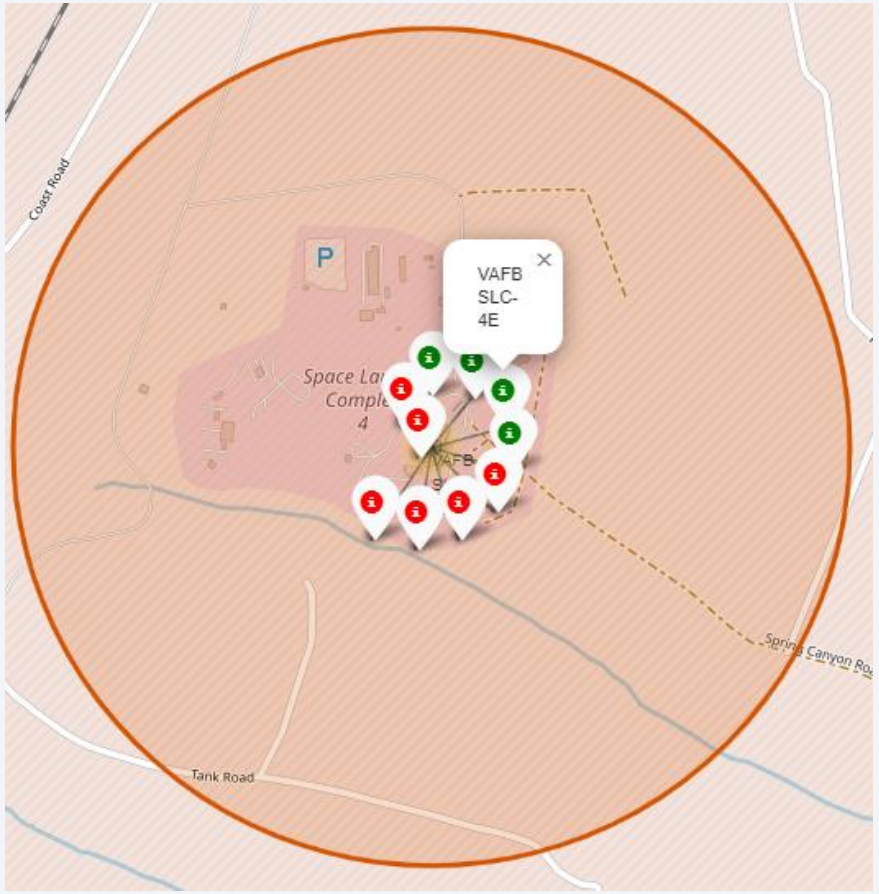
Launches in the eastern part of the USA

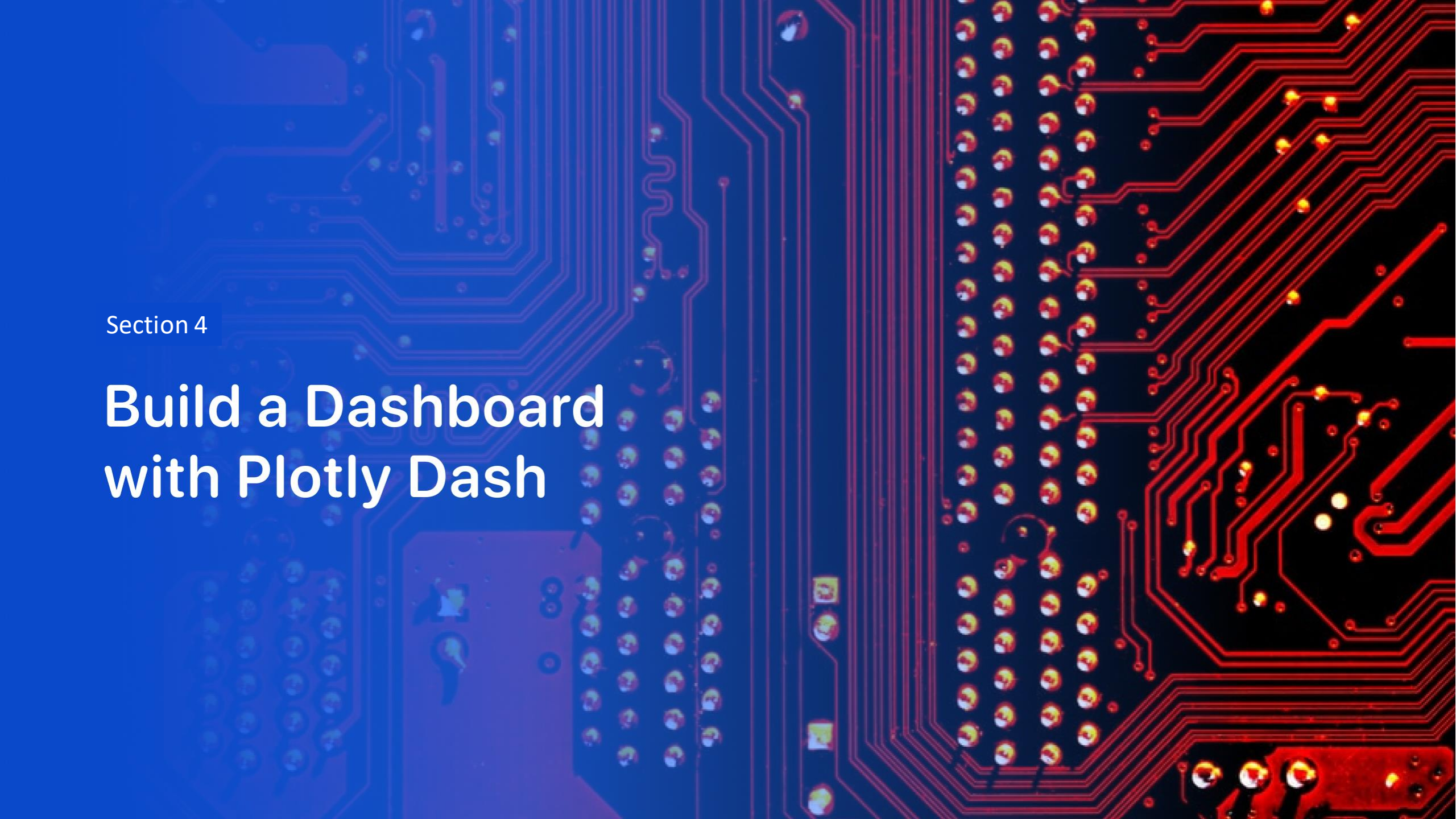


- KSC LC-39A has higher success rate than CCFS LC-40 or CCAFS SLC-40

Launches in the western part of the USA

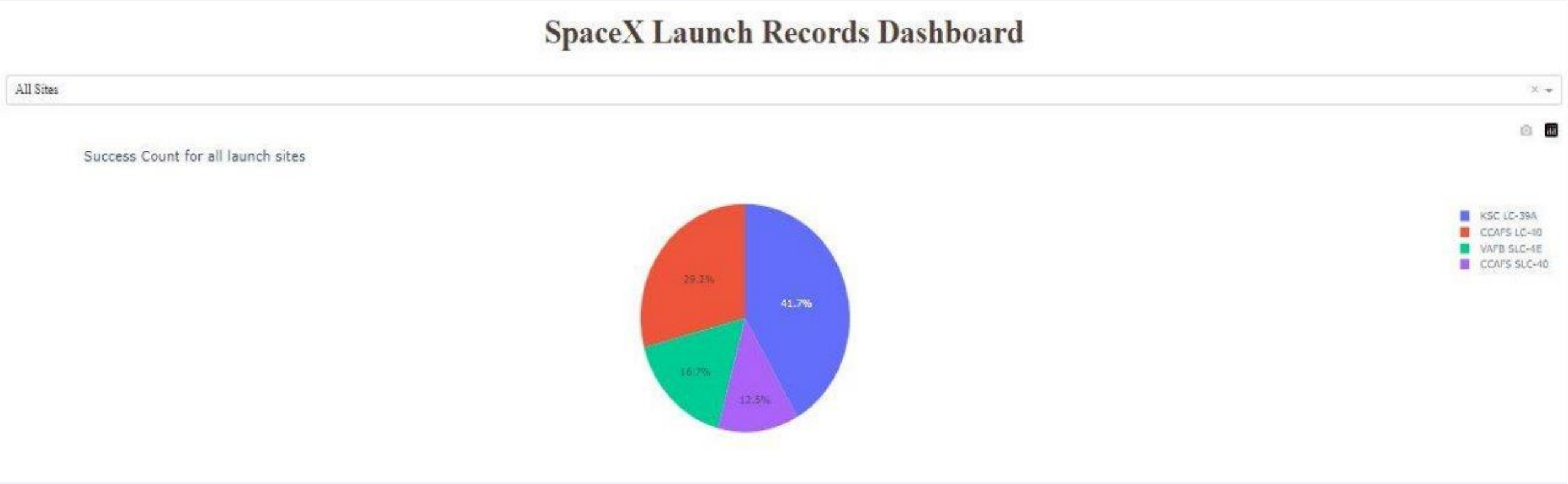
- Lower success rate in California, in VAFB SLC-4E than in Florida.




Section 4
Build a Dashboard with Plotly Dash

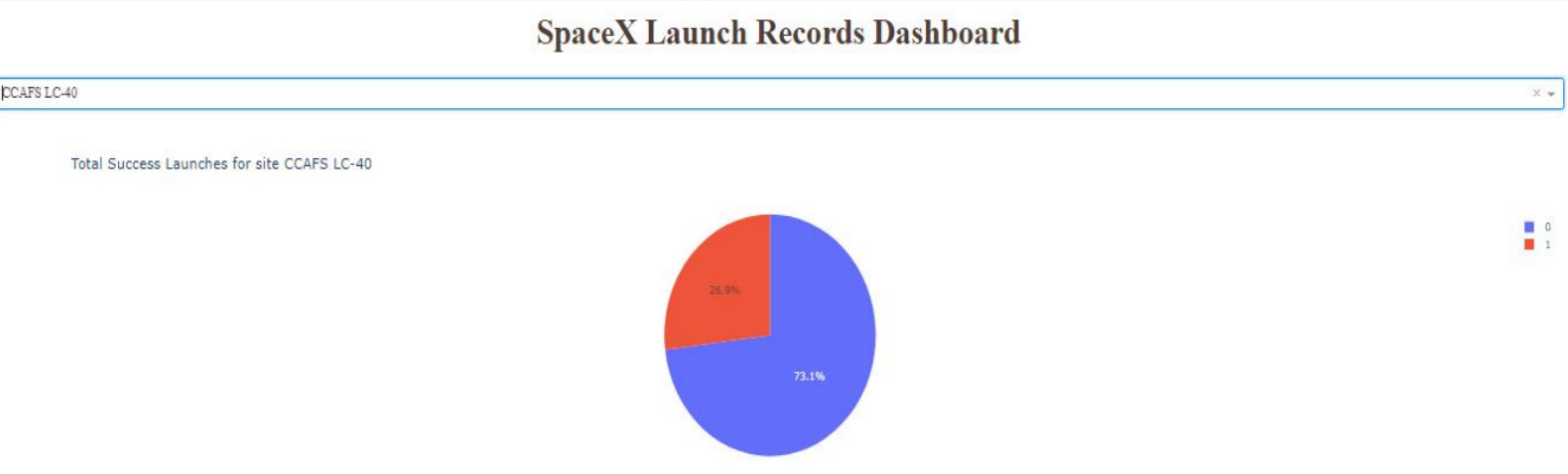
Pie chart: distribution of the launches

- The launch site KSC LC-39A boasts the highest launch success rate at 42%



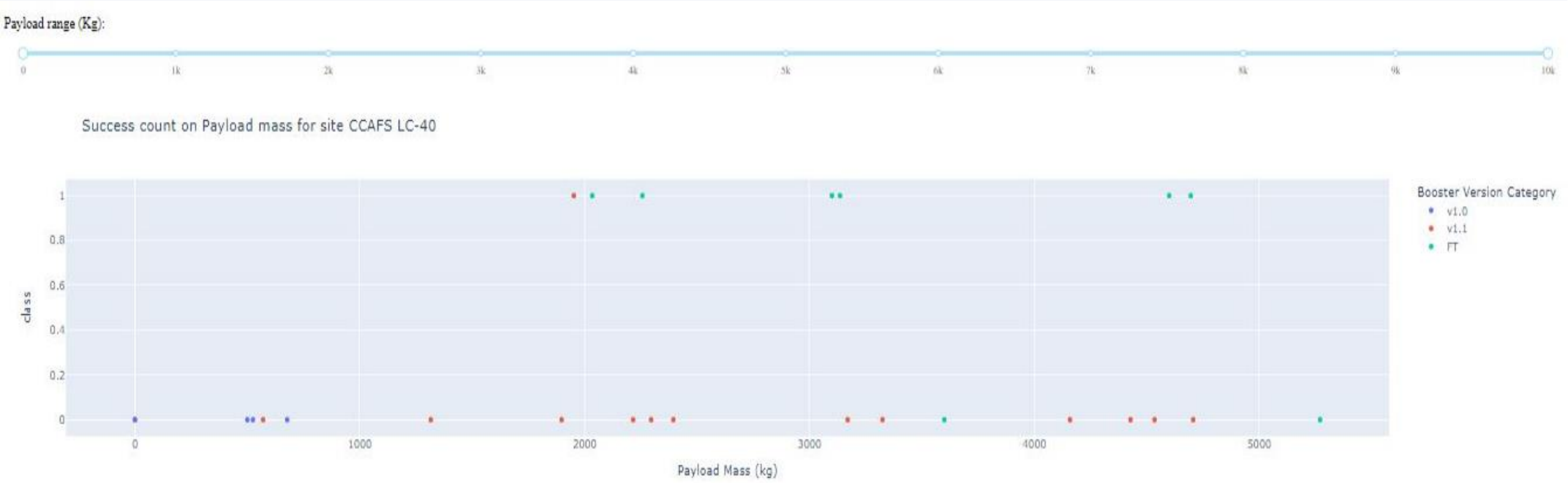
Pie chart, second highest success rate

- The launch site CCAFS LC-40 achieved the second-highest success ratio, with a 73% success rate compared to 27% of failed launches.



Scatter plot (payload - success of the lanches)

- The CCAFS LC-40 launch site exhibits the highest success rate with booster version FT for payloads exceeding 2000kg in mass.



Section 5

Predictive Analysis (Classification)

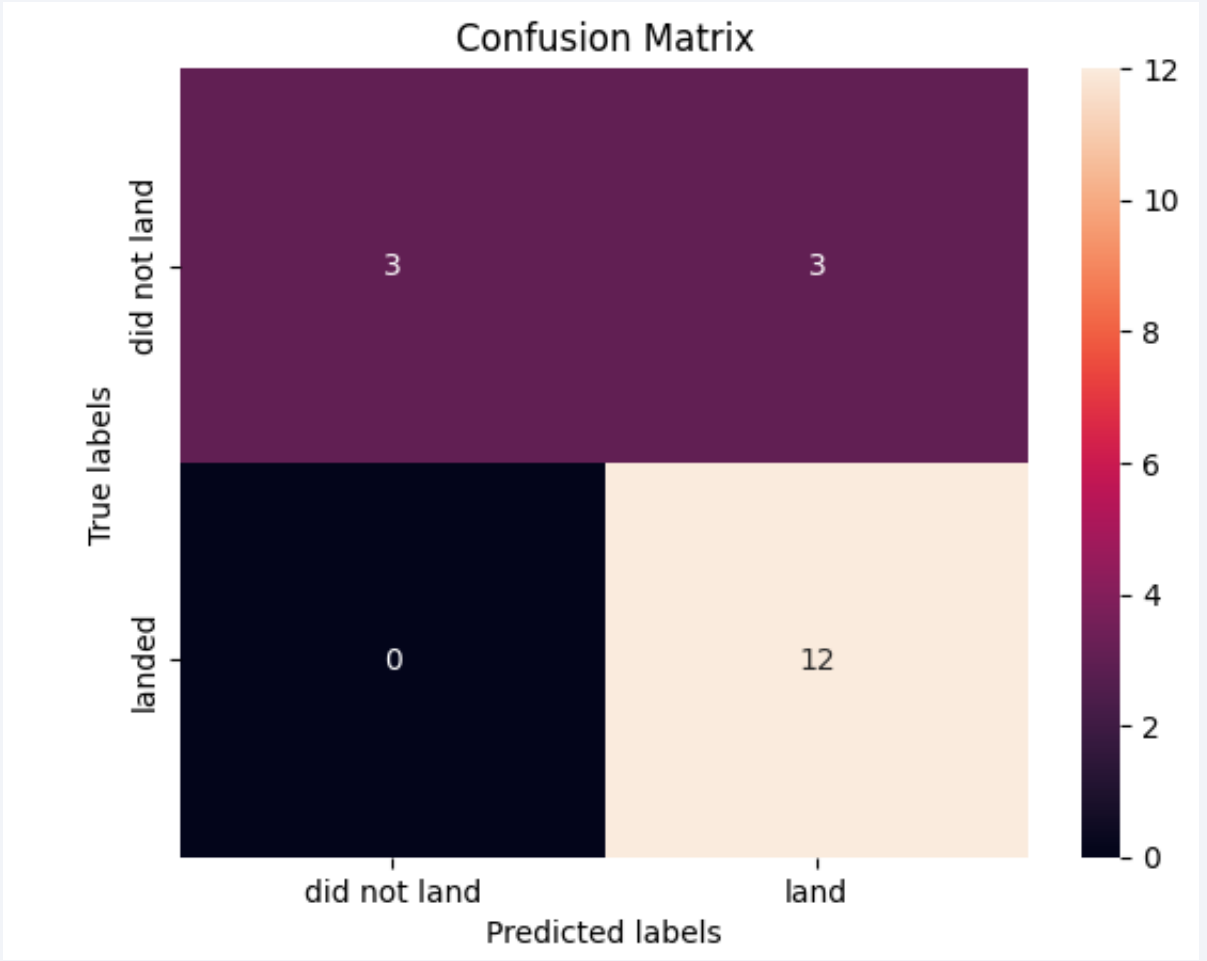
Classification Accuracy

- All the methods accuracy is the same. Its value is 0.83333.

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

Confusion Matrix

- Each of the four classification models exhibited identical confusion matrices, indicating an equal ability to distinguish between different classes. However, a common issue across all models was the occurrence of false positives.



Conclusions

- There are different success rates for each launching sites: CCAFS LC-40 60%, KSC LC-39A and VAFB SLC 4E 77%.
- With flight number increases the success rate also increases. Notably, VAFB SLC 4E achieves a 100% success rate after the 50th flight, while both KSC LC 39A and CCAFS SLC 40 reach this milestone after the 80th flight.
- Payload Vs. Launch Site scatter plot, it becomes evident that no rockets with heavy payload masses (exceeding 10,000) are launched from the VAFB-SLC site.
- Orbits ES-L1, GEO, HEO, and SSO exhibit impeccable success rates of 100%, contrasting with the SO orbit's dismal 0% success rate.
- In the LEO orbit, success appears to be positively correlated with the number of flights, whereas no such relationship is observed in the GTO orbit.
- When dealing with heavy payloads, positive landing rates are notably higher for Polar, LEO, and ISS orbits. However, distinguishing between positive and negative landings is challenging for GTO, as both outcomes are prevalent.
- Lastly, the success rate has shown a consistent upward trend since 2013, steadily increasing until 2020.

Thank you!