IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Faris Karim
July 14, 2023

Outline

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

Executive Summary

- Summary of methodologies:
  - Data Collection (API, Web Scraping)
  - Data Wrangling
  - EDA with SQL, and Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Predictions using Classification algorithms
- Summary of all results
  - EDA results in screenshots and explanations
  - Folium map screenshots and dashboard screenshots
  - Predictive analysis results using a variety of machine learning algorithms in order to find the best and most accurate one

Introduction

- The commercial space age is here, and SpaceX stands out as the most successful company in this field.
    - Can be partly attributed to the cost of their rocket launches compared to other companies. The Falcon9 rocket launches cost $62 million, while other providers charge upwards of $162 million.
    - This is because SpaceX can recover their first stage.
    - However, SpaceX is not always successful in doing so; sometimes the first stage crashes, and other times it is sacrificed because of mission parameters.
- We aim to determine the cost of each launch by predicting the fate of SpaceX's first stage.
    - If we can determine if the first stage will land, we can determine the cost of a launch.
- Another goal of ours is to determine if SpaceX will reuse the first stage. These predictions will come from machine learning models and public information about the Falcon9 rocket launches

Section 1

Methodology

Methodology

- Perform data wrangling
    - Setting the values for the outcome column: 1 for success and 0 for failure. Then, creating a new column called class to hold these binary values. This will be our dependent variable
- Perform exploratory data analysis (EDA) using visualization and SQL
    - Visualizing the relationship between multiple independent variables in the data set
    - Performing queries using SQL in order to better understand the data and find common occurrences
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - The classification models used were:
        - logistic regression
        - Support Vector Machine
        - Decision Tree
        - K-Nearest Neighbors

Data Collection

The data gathered deals with launch data, and comes from the SpaceX REST API. The data contains information about launch specifications, landing specifications, payload delivered, and landing outcome amongst other things.

In addition, we gathered data from Wikipedia using web scraping, in order to get more information about features of Falcon9

Data Collection – SpaceX API

- Make a get request for the data using the SpaceX API
- Convert the json object into a pandas dataframe using json_normalize()
- Take a subset of the data to include relevant independent variables
- Apply relevant functions to the data and create a dictionary from it; then create a new Pandas data frame from the dictionary

https://github.com/FarisKarim/DataScienceCapstoneSpaceX/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

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

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

```
: # Create a data from launch_dict
  new_data = pd.DataFrame.from_dict(launch_dict)
```

Data Collection - Scraping

- Request the Falcon9 Launch Wiki page from its URL, and create a BeautifulSoup object from its response
- Extract the column/variable names from the table header
- Parse the HTML tables and create a data frame

```
# use requests.get() method with the provided static_url
# assign the response to a object

response = requests.get(static_url)
```

```
soup = BeautifulSoup(response.content, 'html.parser')
```

```
# 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')
```

```
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']=[]
```

https://github.com/FarisKarim/DataScienceCapstoneSpaceX/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

- Through data wrangling we want to find patterns in the data in order to determine what the label would be for training supervised models
- First we apply value_counts() on the 'LaunchSite' column in order to determine the number of launches for each site
- Next we do the same for the 'Orbit' column and the other relevant columns
- Next, set the values (0 for failure, 1 for success) of the 'Outcome' column to a new column named 'Class.' We find the mean of the new column to determine the overall success rate

https://github.com/FarisKarim/DataScienceCapstoneSpaceX/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_1_L3_labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

[7]:
```
# Apply value_counts() on column LaunchSite

df['LaunchSite'].value_counts()
```

[7]:
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

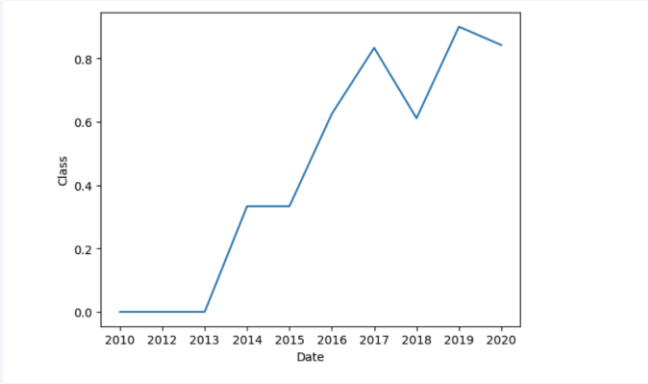
[14]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for key,value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

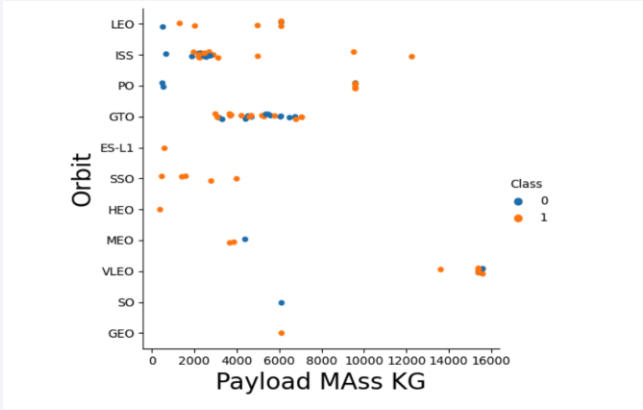
This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

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

EDA with Data Visualization

- We plot various charts in order to try and find the relationship between the various independent variables. In addition, we plot the success rate as the years go on. Some of the variables we plotted against each other were:
  - Payload vs Flight Number
  - Flight Number vs Launch Site
  - Payload vs Launch Site
  - Orbit Type vs Payload
- https://github.com/FarisKarim/DataScienceCapstoneSpaceX/blob/main/EDAVisualizationdataviz.ipynb.jupyterlite.ipynb





EDA with SQL

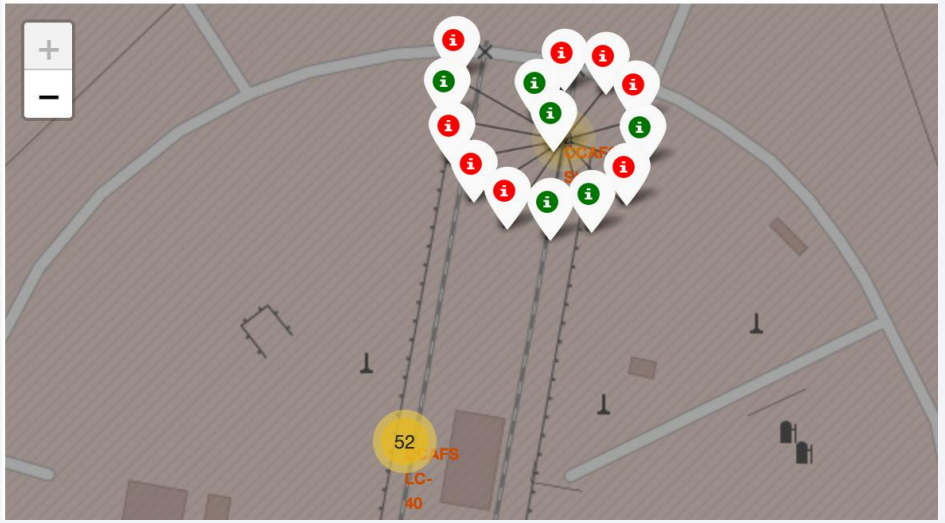
- Using SQL, these are some of the queries we made in order to explore the data further:
  - Selecting the distinct launch sites from the table
  - Selecting the distinct launch sites that start with the string 'CCA'
  - Finding the sum of all of the payload masses where the customer was NASA CRS
  - Finding the date where the first successful landing outcome in ground pad was achieved
  - Listing total number of successful and unsuccessful mission outcomes
  - Ranking the count of landing outcomes between dates in descending order

- https://github.com/FarisKarim/DataScienceCapstoneSpaceX/blob/main/EDA_SQL.ipynb

Build an Interactive Map with Folium

For the interactive map, we added circles for the launch sites and added markers to each launch sites to make it easier to see which launch sites had more successful landings then the other ones.

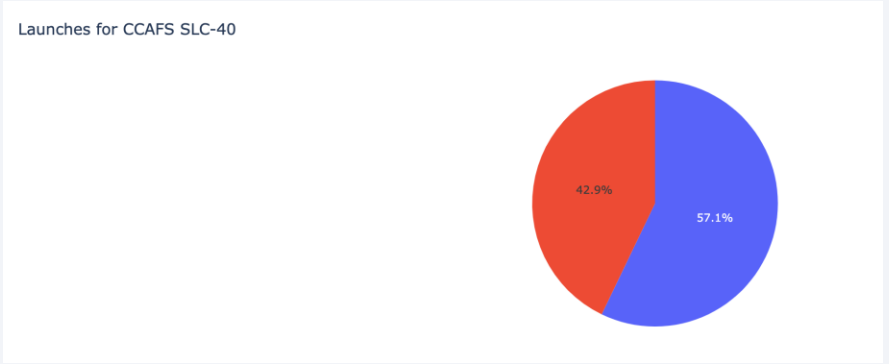
Visualizing these launch sites on a map makes it easier to see where exactly each rocket was launched from

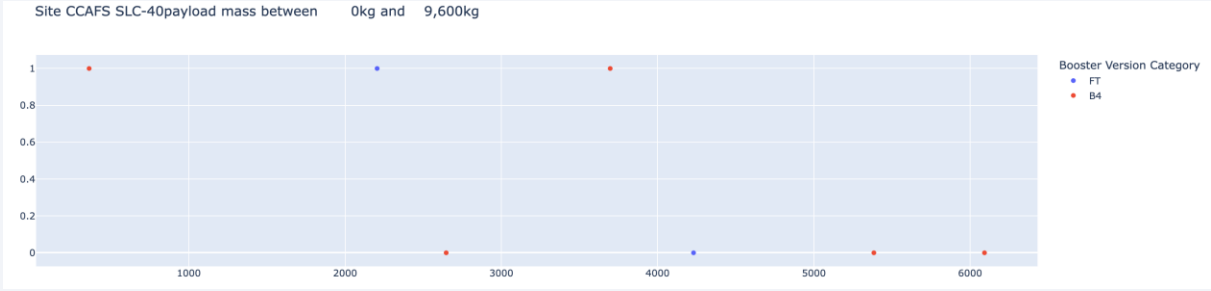


- https://github.com/FarisKarim/DataScienceCapstoneSpaceX/blob/main/FoliumMap.ipynb

Build a Dashboard with Plotly Dash

- We added a dashboard to help visualize the successful and unsuccessful landings from each launch site better. We did this in the form of a pie chart, with blue showing the successful launches and red showing the unsuccessful launches.
- In addition, we added a scatterplot that shows the different booster versions for each launch site. We are able to specify a certain range of payload mass by moving the range slider



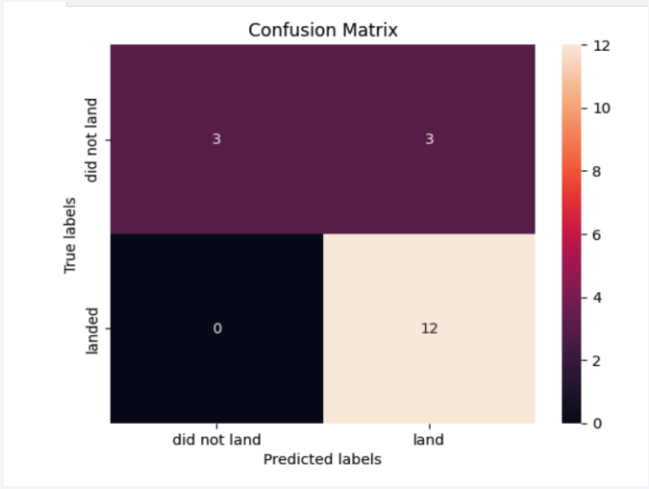


- https://github.com/FarisKarim/DataScienceCapstoneSpaceX/blob/main/spacex_dash_app.py.

Predictive Analysis (Classification)

- In our predictive analysis, first we transformed the class column to a numpy array. Then we standardize the independent variables in order to help avoid bias and improve optimization
- We use the function train_test_split in order to split the training and testing data.
- We use GridSearchCV on our machine learning algorithm in order to find the best hyperparameters for a given model. The algorithms we used in our analysis are:
  - Logistic Regression
  - Support Vector Machine
  - Decision Tree
  - K-Nearest Neighbors
- We then use a confusion matrix to analyze the performance of the model. We can also find the accuracy and best score of the model.

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()

lrgs = GridSearchCV(estimator = lr, cv=10, param_grid = parameters)

logreg_cv = lrgs.fit(X_train, Y_train)
```




```
print(f"Accuracy for Logistics Regression method is {logreg_cv.score(X_test, Y_test)}")
print(f"Accuracy for Support Vector Machine method {svm_cv.score(X_test, Y_test)}")
print(f"Accuracy for Decision tree method is {tree_cv.score(X_test, Y_test)}")
print(f"Accuracy for K nearest neighbors method is {knn_cv.score(X_test, Y_test)}")
```

```
Accuracy for Logistics Regression method is 0.8333333333333334
Accuracy for Support Vector Machine method 0.8333333333333334
Accuracy for Decision tree method is 0.7777777777777778
Accuracy for K nearest neighbors method is 0.8333333333333334
```

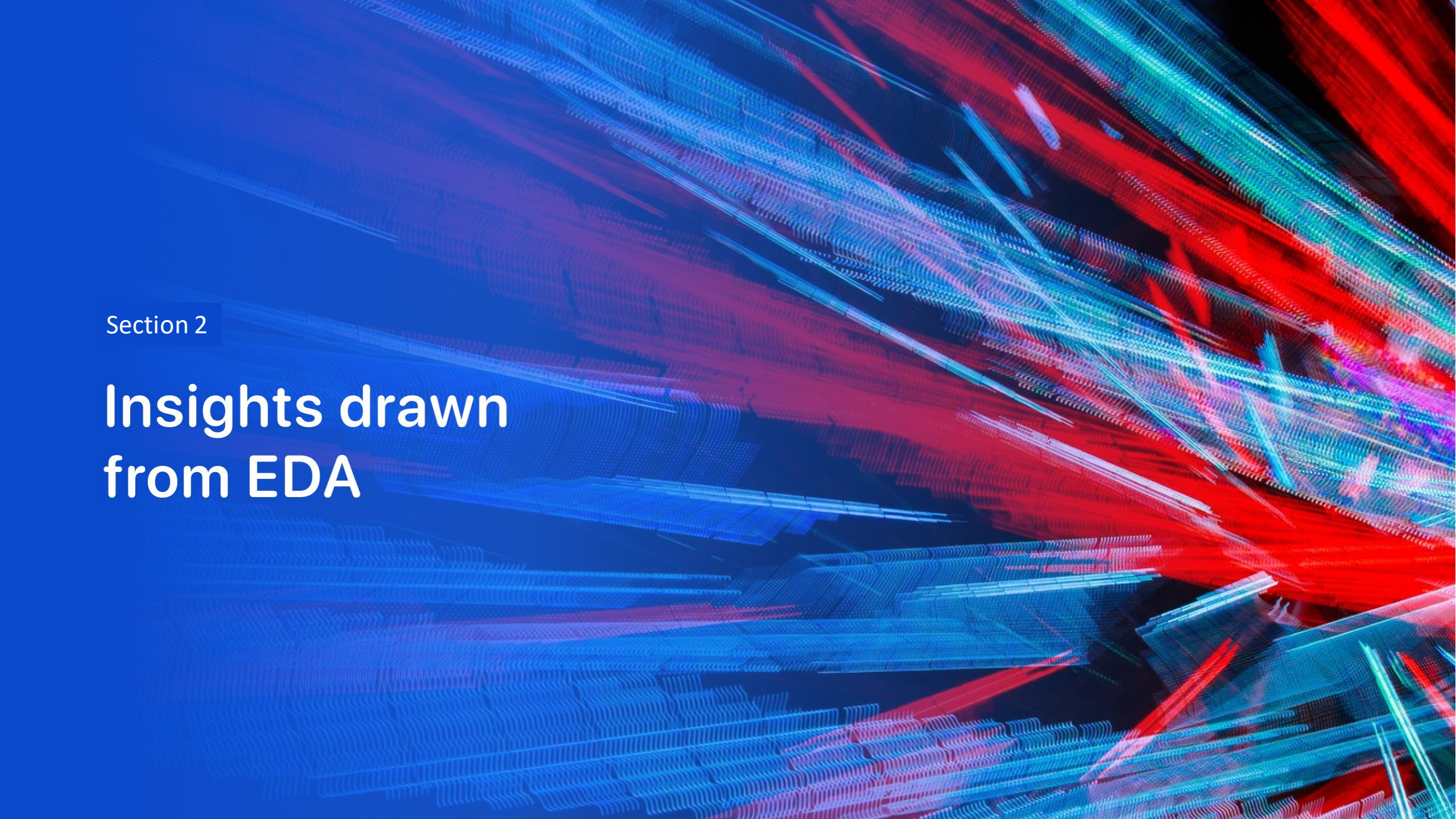
- https://github.com/FarisKarim/DataScienceCapstoneSpaceX/blob/main/MachineLearningPrediction.ipynb

Results

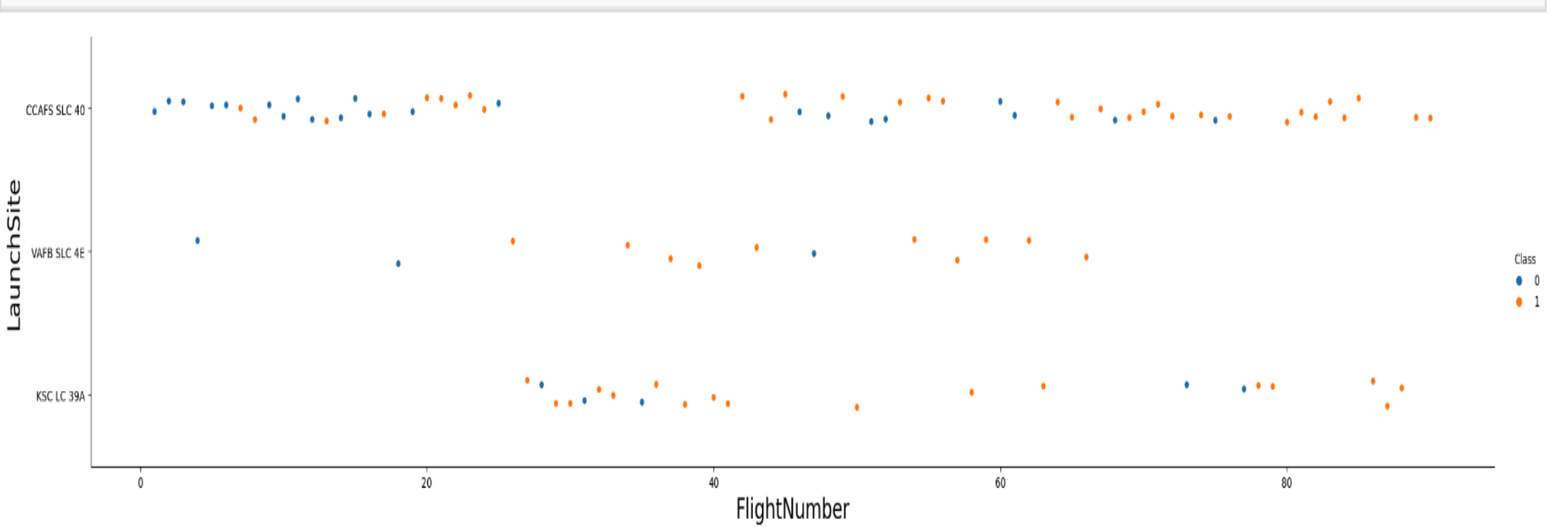
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA



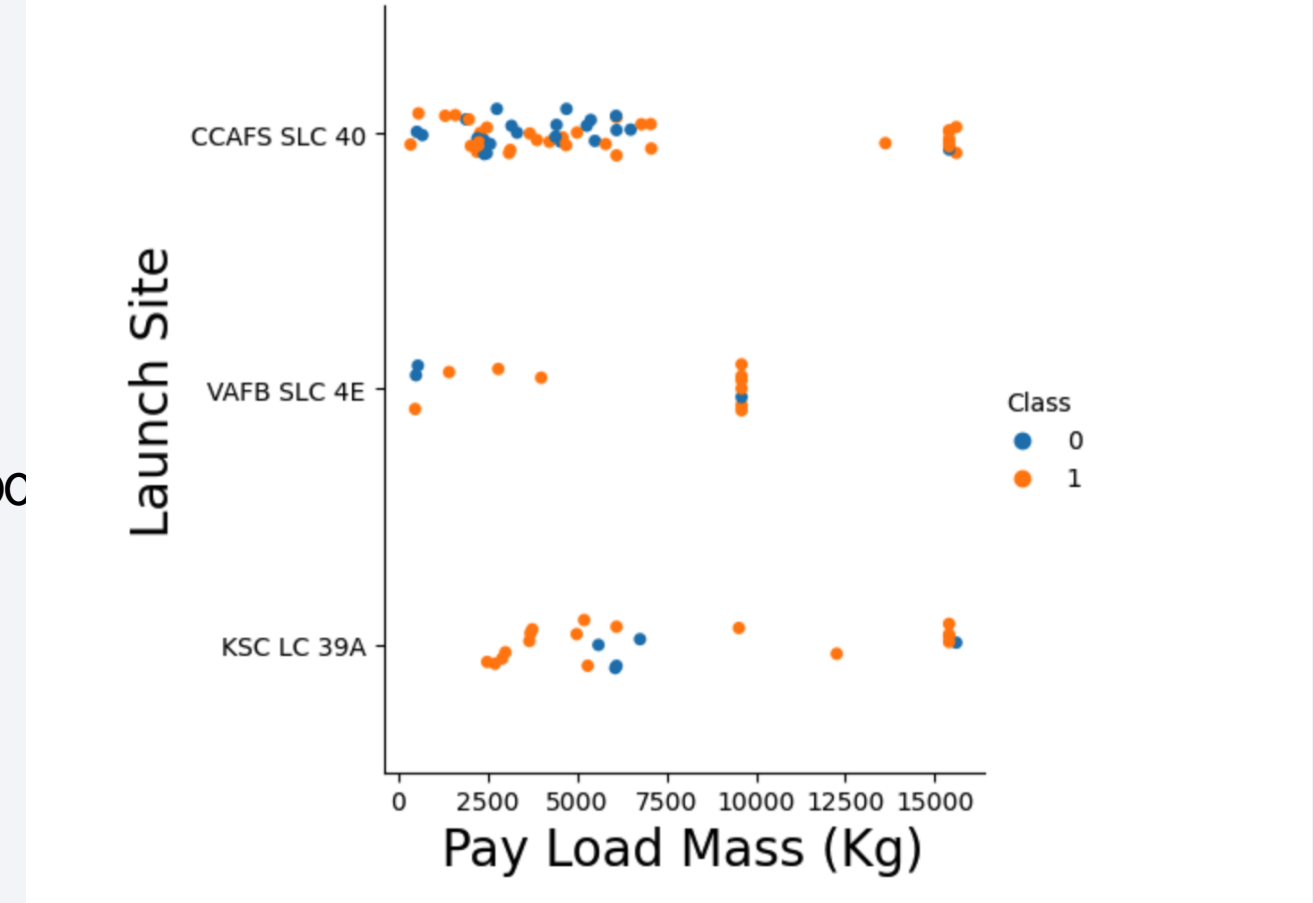
Flight Number vs. Launch Site



From this scatter plot we can see that different Launch Sites have different success rates. As the flight number increases, the Launch Site VAFB SLC 4E is not used anymore

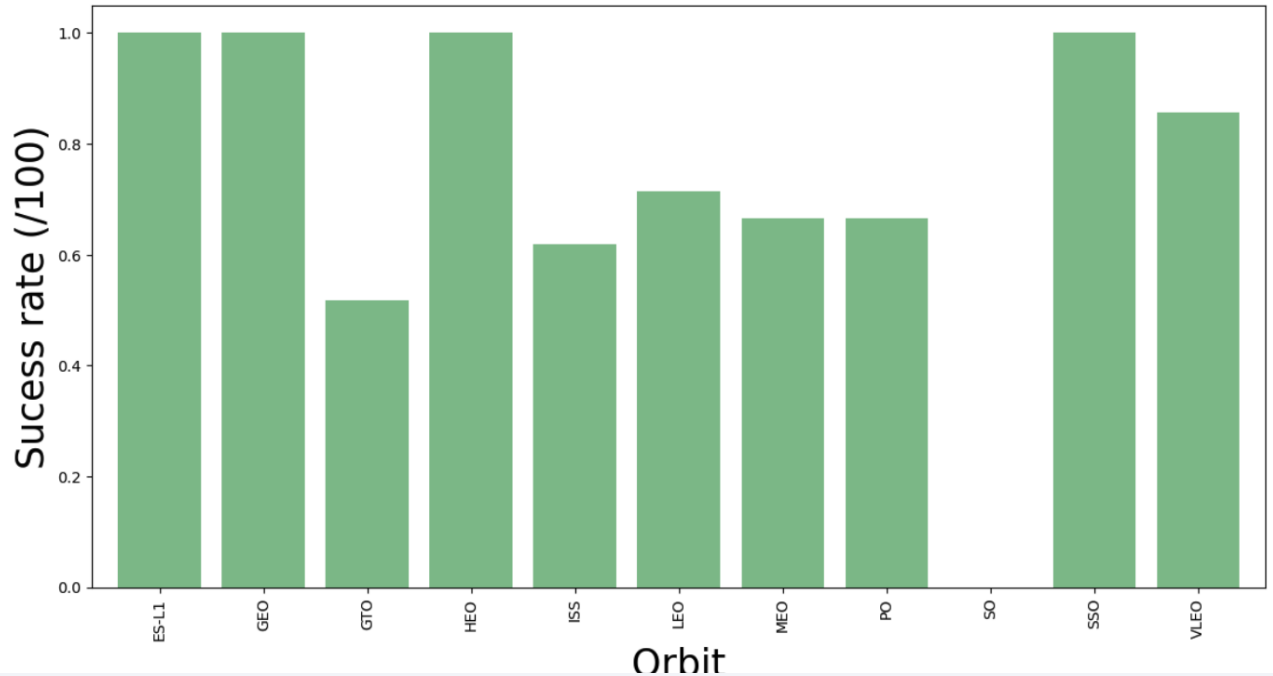
Payload vs. Launch Site

From this scatter plot we can see the payload mass vs flight number. There are no rockets launched for payload masses greater than 10000 for the VAFB SLC 4E launch site.



Success Rate vs. Orbit Type

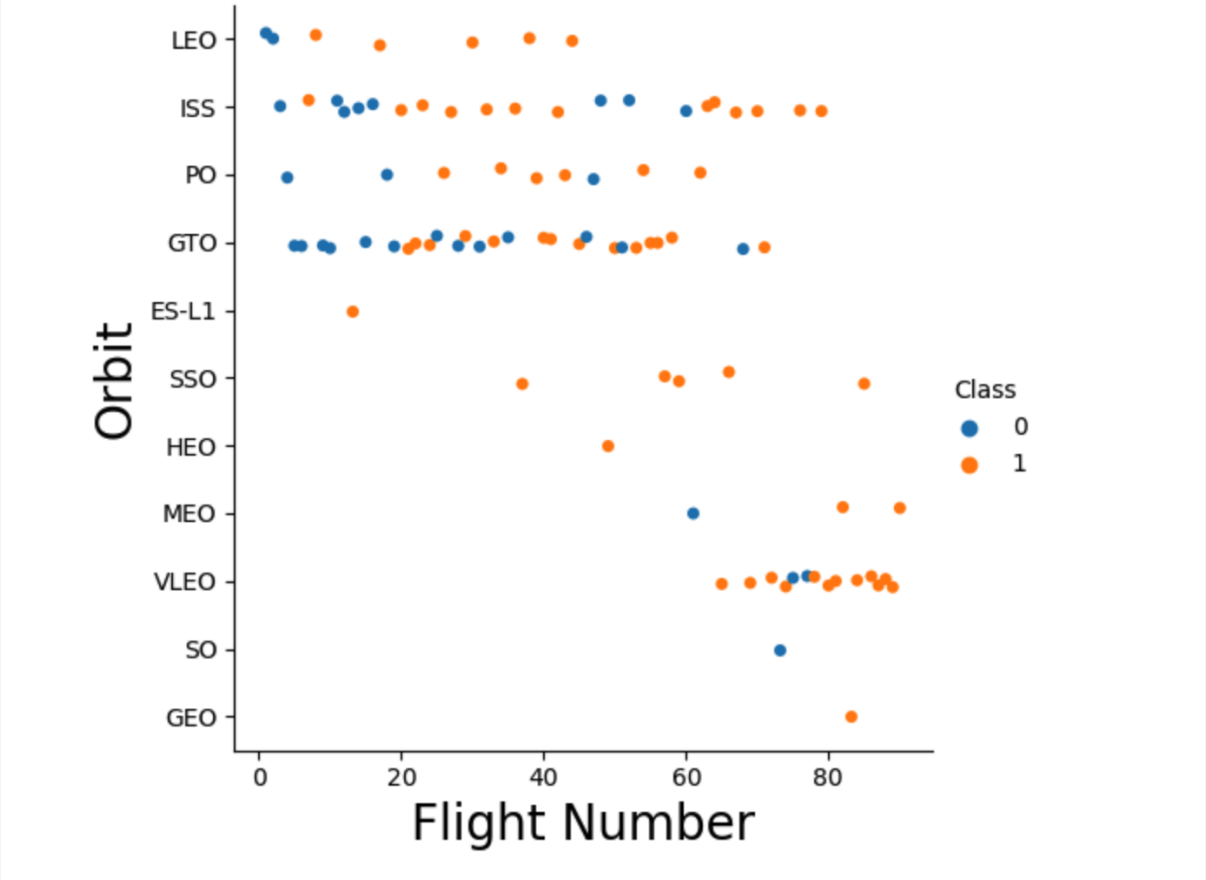
```
[9]: # HINT use groupby method on Orbit column and get the mean of Class column

success = df.groupby('Orbit')['Class'].mean()
bar = success.plot(kind='bar', figsize=(14, 7), color='#86bf91', zorder=2, width=0.8)
bar.set_xlabel("Orbit", size = 25)
bar.set_ylabel("Sucess rate (/100)", size=25);
```



We can see the success rates for the various orbit types. There are several orbit types with a 100% success rate

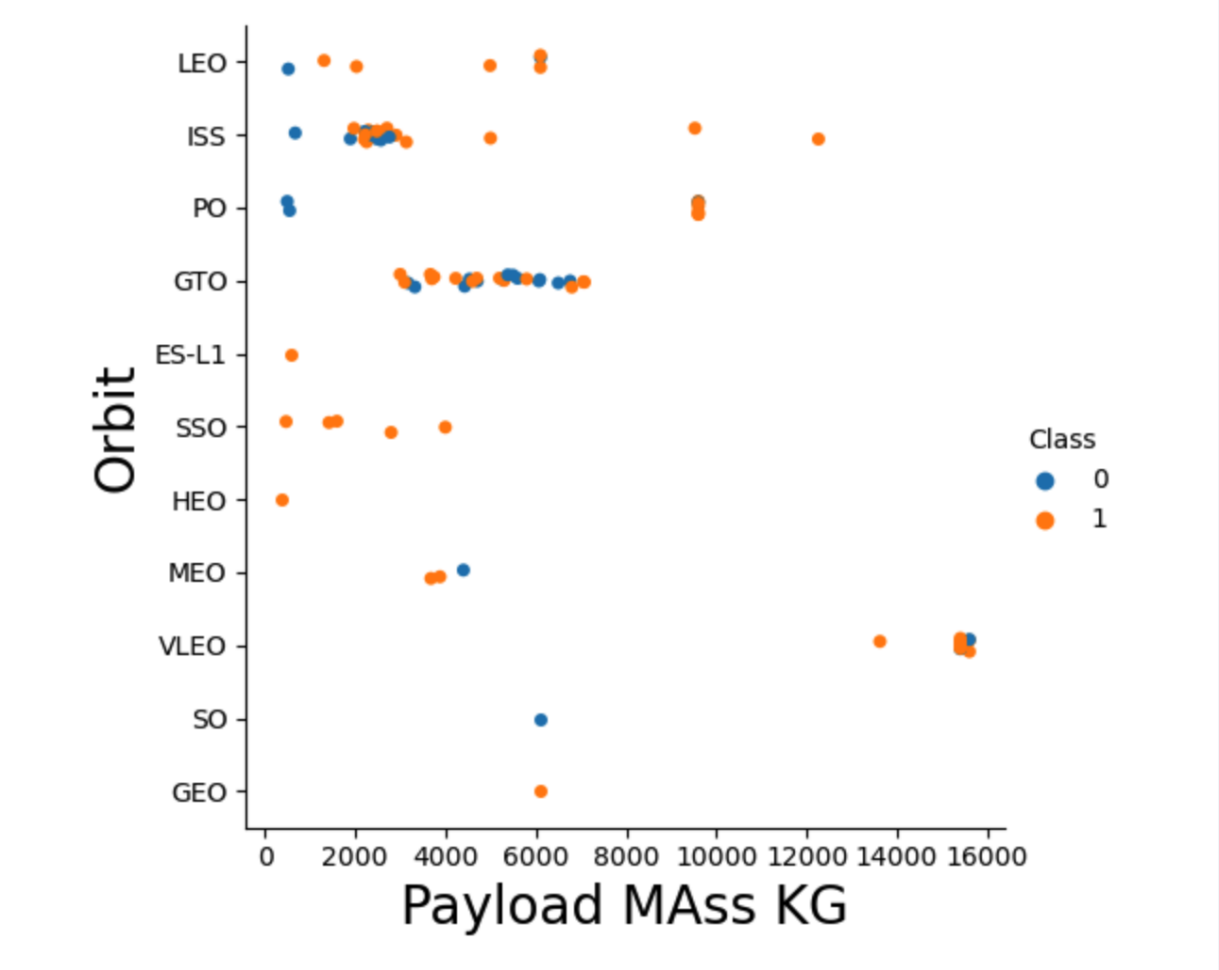
Flight Number vs. Orbit Type

- There seems to be no relationship between flight number and success rate for the orbit type GTO



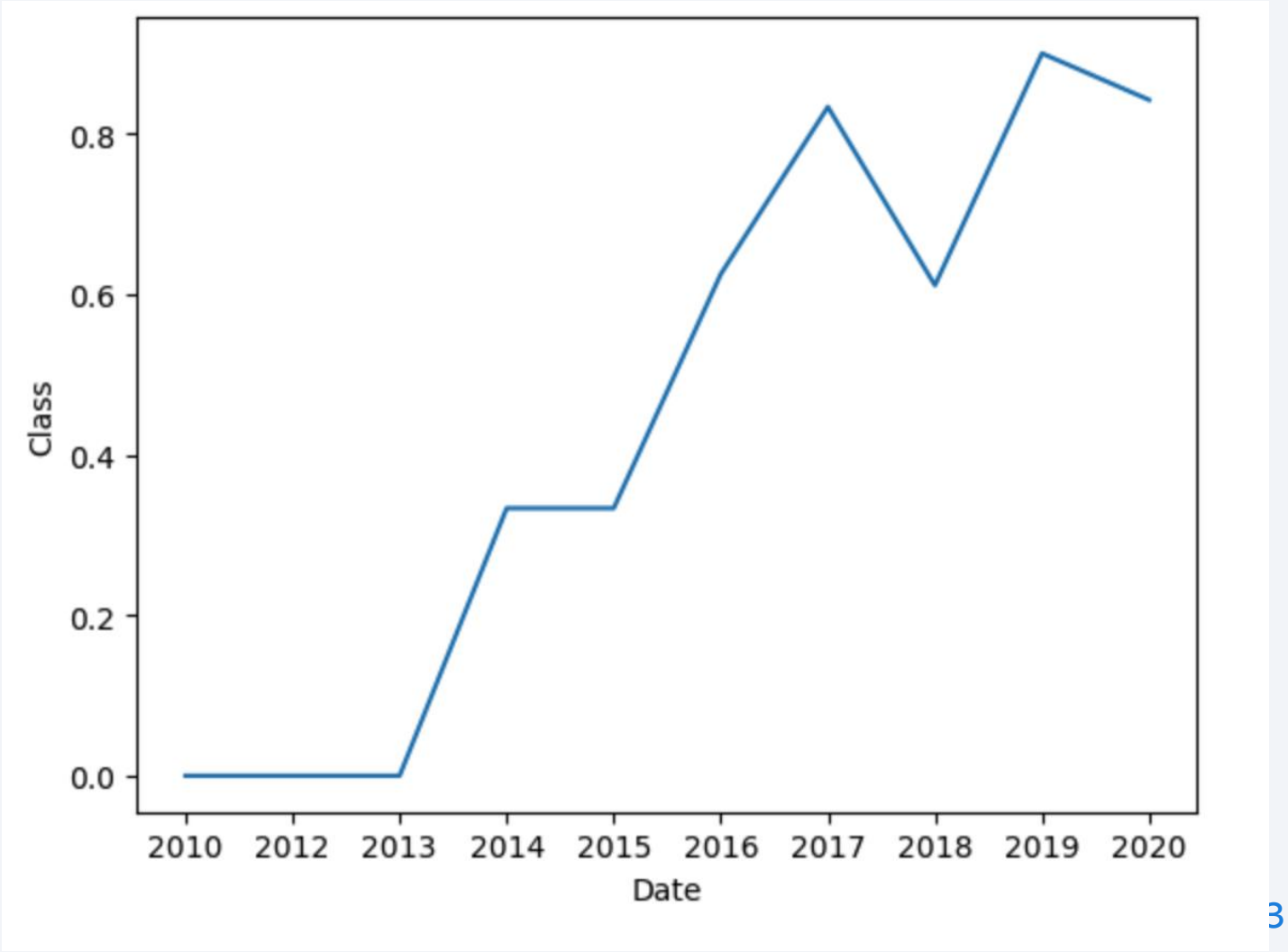
Payload vs. Orbit Type

- Heavy payloads seem to be more successful for orbit types LEO and ISS



Launch Success Yearly Trend

- As you can see from the graph, the success rate tends to increase as the years go by



All Launch Site Names

- This query shows the distinct launch sites in the dataset

```
[7]: %sql Select DISTINCT Launch_site from 'SPACEXTBL';
 * sqlite:///my_data1.db
Done.
```

[7]:

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

Launch Site Names Begin with 'CCA'

- From this query we find the first 5 rows where the launch site starts with '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_M
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	

Total Payload Mass

```
[9]: %sql SELECT SUM(PAYLOAD_MASS__KG_) as SUM from SPACEXTBL WHERE Customer = 'NASA (CRS)'
 * sqlite:///my_data1.db
Done.
[9]:    SUM
    45596.0
```

This calculates the total payload carried by boosters from NASA

The sum of the payload masses where the customer is 'NASA (CRS)' is 45596

Average Payload Mass by F9 v1.1

```
0]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.1%'
 * sqlite:///my_data1.db
Done.
0]:
```

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

This query calculates the average payload mass carried by booster version F9 v1.1

The average payload mass for the booster version is ~2535kg.

First Successful Ground Landing Date

```
%sql SELECT MIN(DATE) as firstLanding from SPACEXTBL WHERE Landing_Outcome LIKE 'Success%'
```

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

firstLanding
01/07/2020

This query shows the first successful landing date

Successful Drone Ship Landing with Payload between 4000 and 6000

```
]: %sql SELECT Booster_Version FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000
 * sqlite:///my_data1.db
Done.
```

]:

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

This query lists the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

There are only four booster versions that have successfully landed with a payload mass in between 4000kg and 6000kg.

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT COUNT(*), MISSION_OUTCOME FROM SPACEXTBL GROUP BY MISSION_OUTCOME
```

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

COUNT(*)	Mission_Outcome
898	None
1	Failure (in flight)
98	Success
1	Success
1	Success (payload status unclear)

This query calculates the total number of successful and failure mission outcomes

As we. Can see, a lot of the data is None, which may mean they have failed, however the majority of the labeled ones are a success.

2015 Launch Records

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE, substr(Date, 4, 2) FROM SPACEXTBL WHERE substr(Date,7,4)='2015' AND LANDING_OUTCOME = 'Failure (drone ship)';
```

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

Booster_Version	Launch_Site	substr(Date, 4, 2)
F9 v1.1 B1012	CCAFS LC-40	10
F9 v1.1 B1015	CCAFS LC-40	04

This query selects the booster version and launch site of failed outcomes in 2015 and gives the corresponding month number

Boosters Carried Maximum Payload

```
4]: %sql SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
 * sqlite:///my_data1.db
Done.
```

4]:

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

This query list the names of the booster which have carried the maximum payload mass

As we can see, the F9 B5 Booster versions all carried the maximum payload mass of all the booster versions

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

```
%sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) FROM SPACEXTBL WHERE DATE BETWEEN '06/04/2010' AND '20/03/2017' GROUP BY LANDING_OUTCOME ORDER BY COUNT(LANDING_OUTCOME) DESC;
```

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

Landing_Outcome	COUNT(LANDING_OUTCOME)
Success	19
No attempt	9
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1

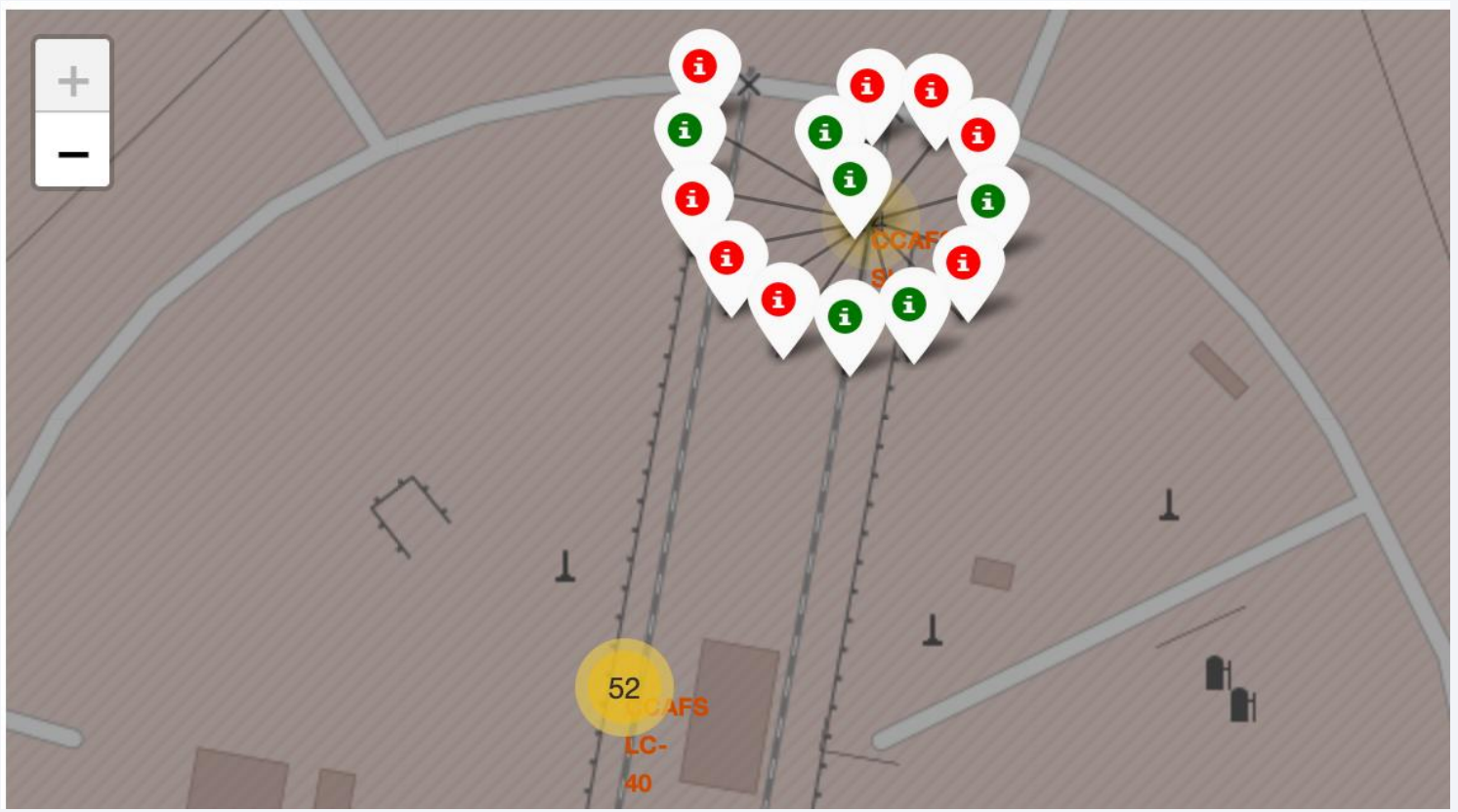
Between the two dates in the query, we can see that there have been more successes than failures


Section 3
Launch Sites
Proximities Analysis

Marked Launch Sites

- This screenshot, generated by Folium, shows the launch sites of the Falcon9. We can see that there is one in Los Angeles and the rest are in Florida

Color Labeled Launch Outcomes

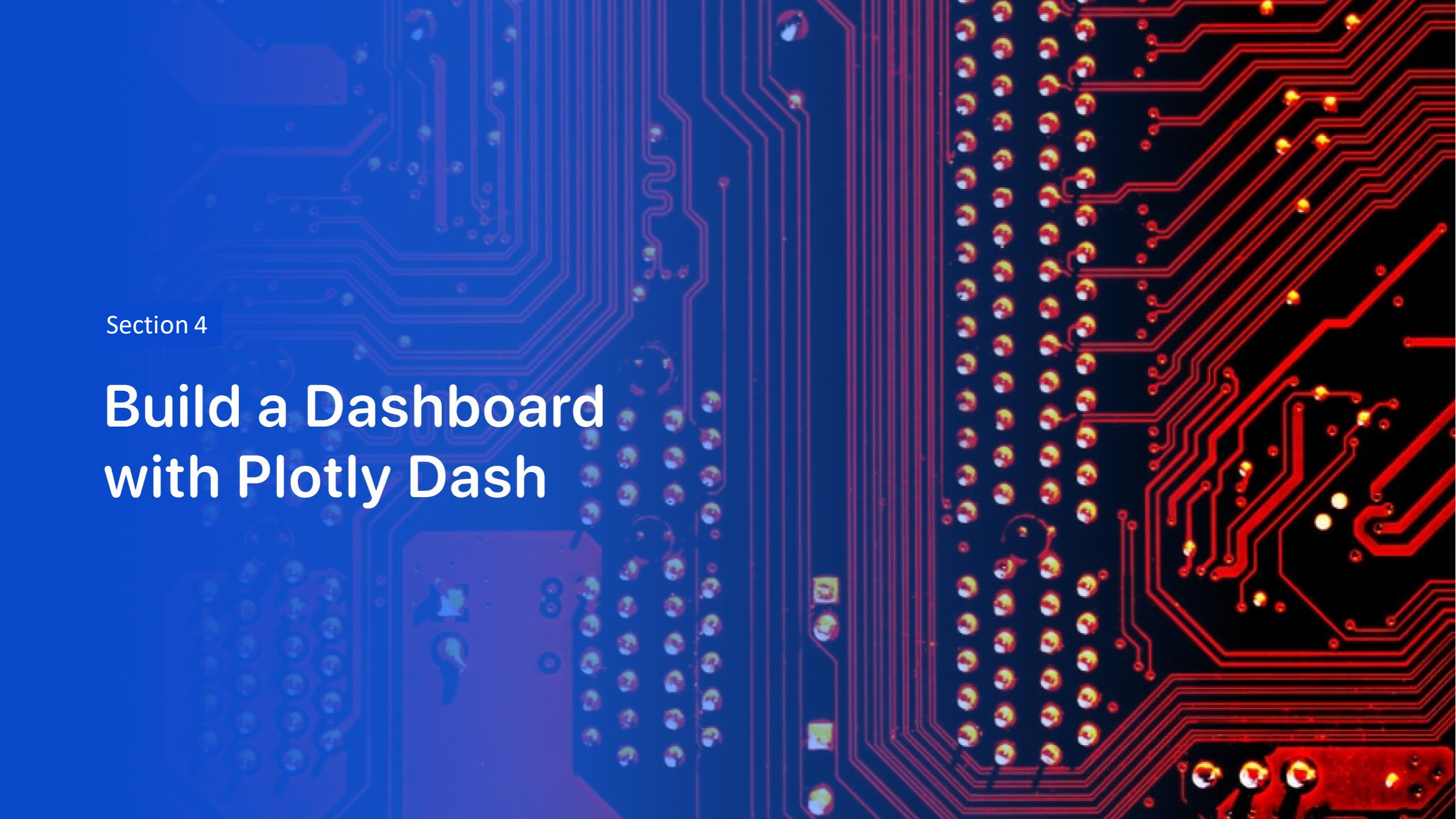


The color-labeled marker clusters make it easy to see which launch sites have high success rates

- Green labels indicate success, while red labels indicate failure

Distance to Coast

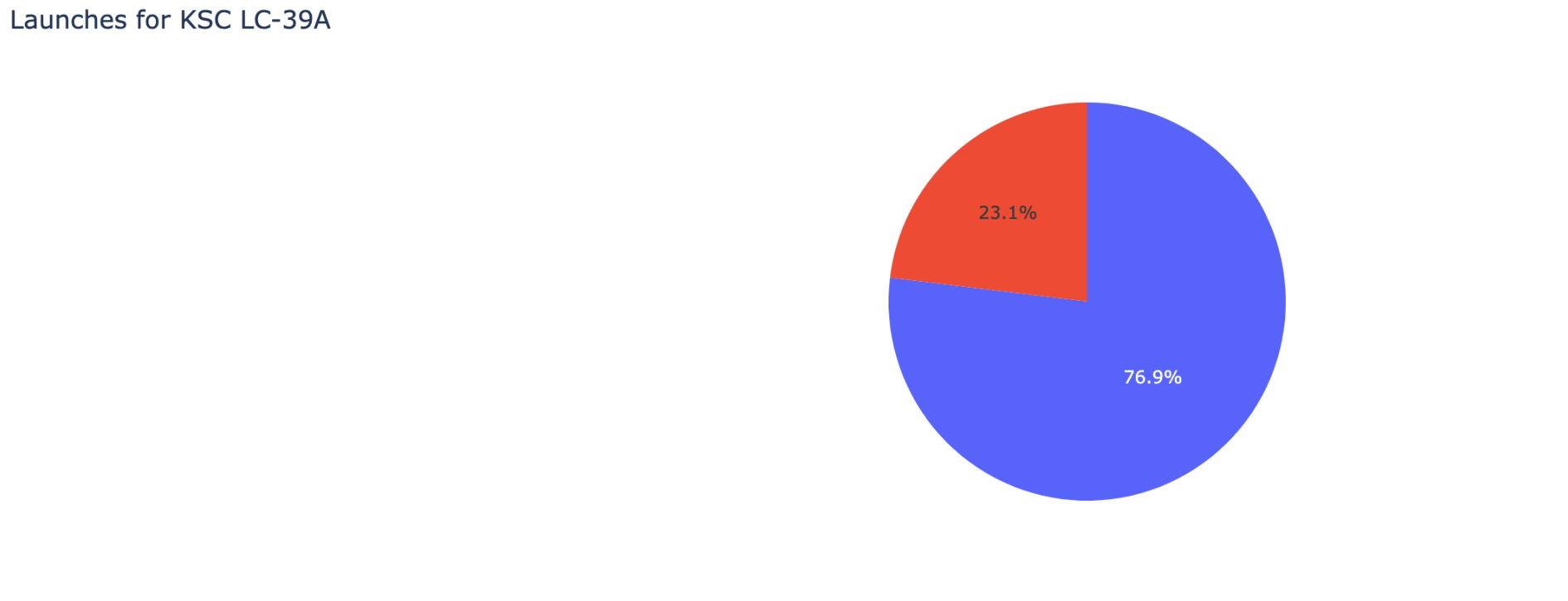
This map shows the distance from the launch site to the coastline


Section 4
Build a Dashboard with Plotly Dash

Launch Success Counts (All sites)

From this pie chart we can see the different launch sites and how they account for the total number of successful launches

Successful and Failed Launches (KSC LAC-39A)



The launch site KSC LAC-39A has the highest rate of success out of all the launch sites

Booster Versions and Success Rate based on Mass

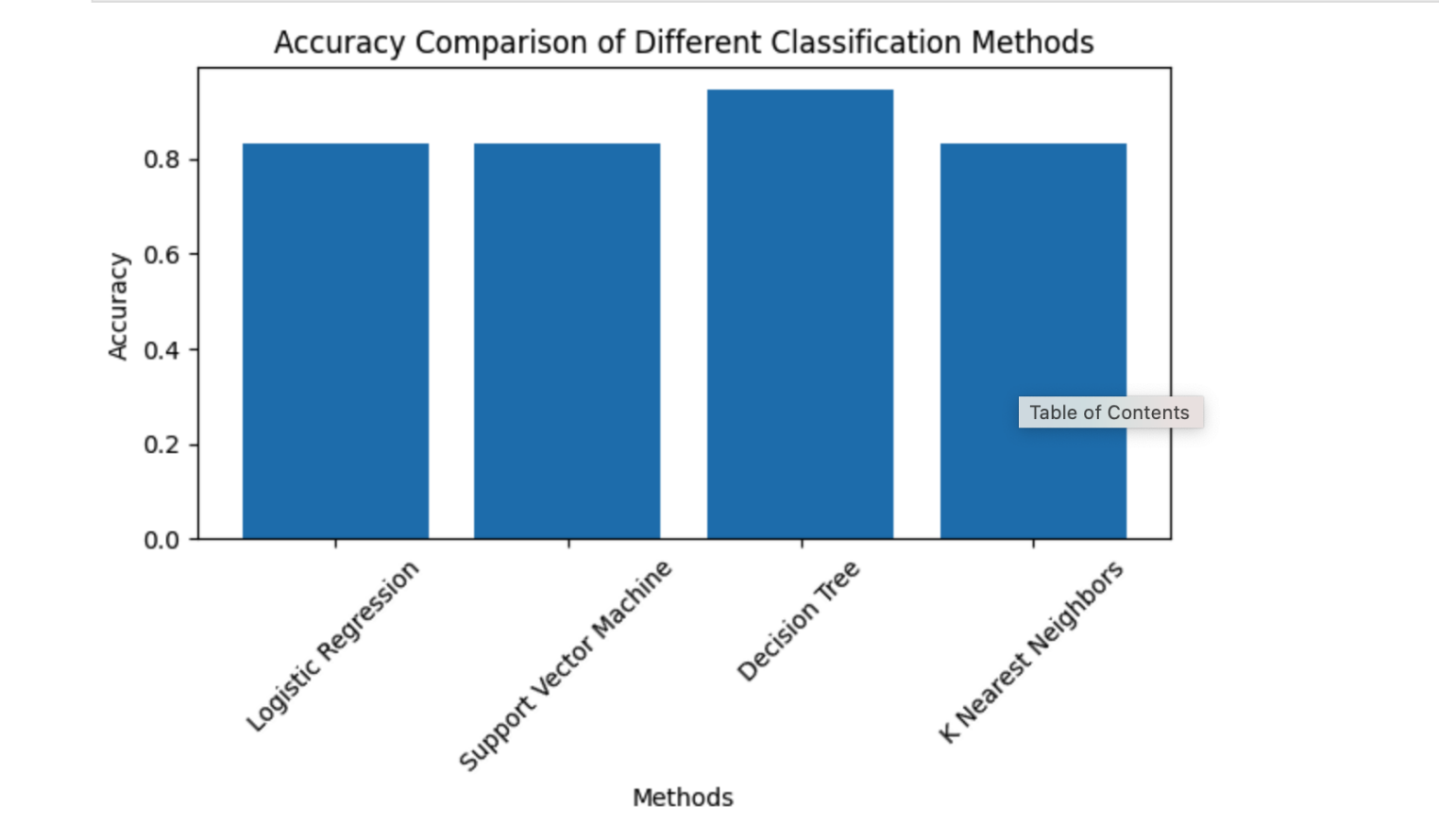
This shows the different successes for the booster version category based on a certain range of mass.

As we can see, the Booster Version category FT has a high number of successes with a payload mass between 2000kg and 10,000kg.

Section 5

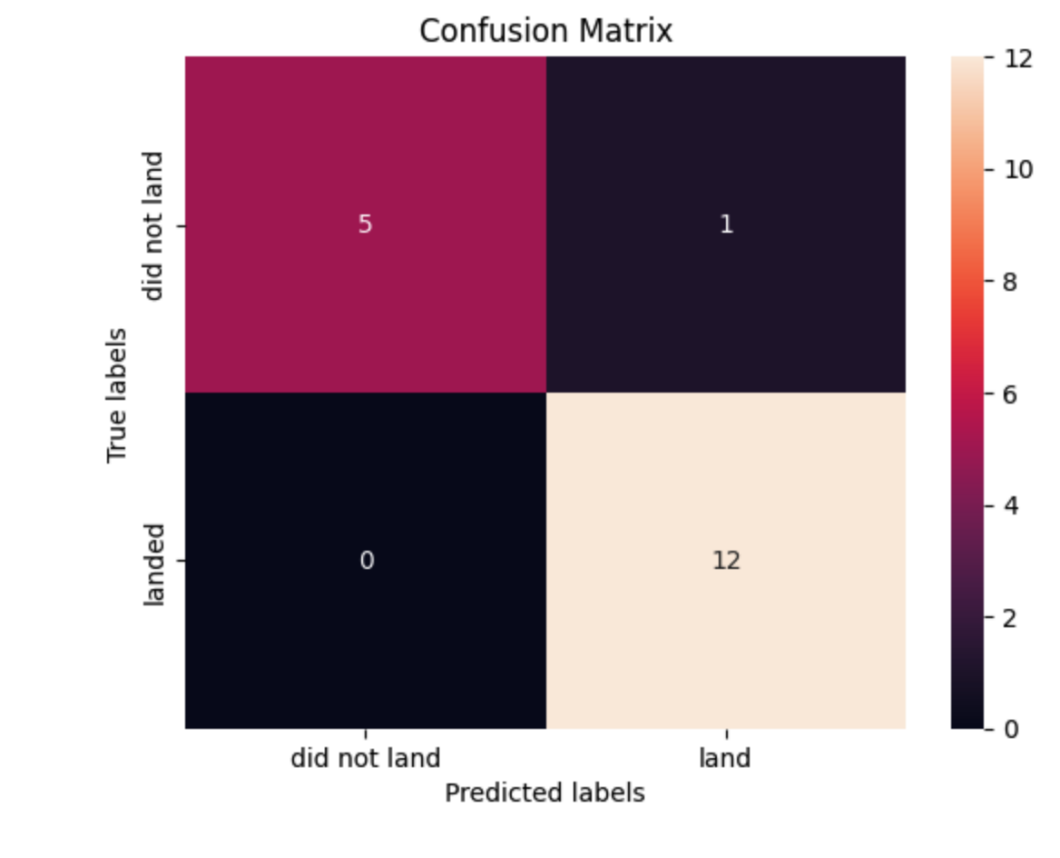
Predictive Analysis (Classification)

Classification Accuracy



This bar chart plots the success rates of the algorithms used. As we can see, the Decision Tree has the highest accuracy

Confusion Matrix



The confusion matrix shows that only one sample was incorrectly labeled (top right.)

Conclusions

- In conclusion, KC9 demonstrated the most successful launch sites, and had the highest success to failure ratio compared to the other sites
- Exploratory data analysis with SQL and data visualization helps us to identify patterns and relationships in the data which
- The decision tree algorithm had the highest classification accuracy, producing only one false positive and correctly labeling all of the other test samples

Thank you!