

Winning Space Race with Data Science

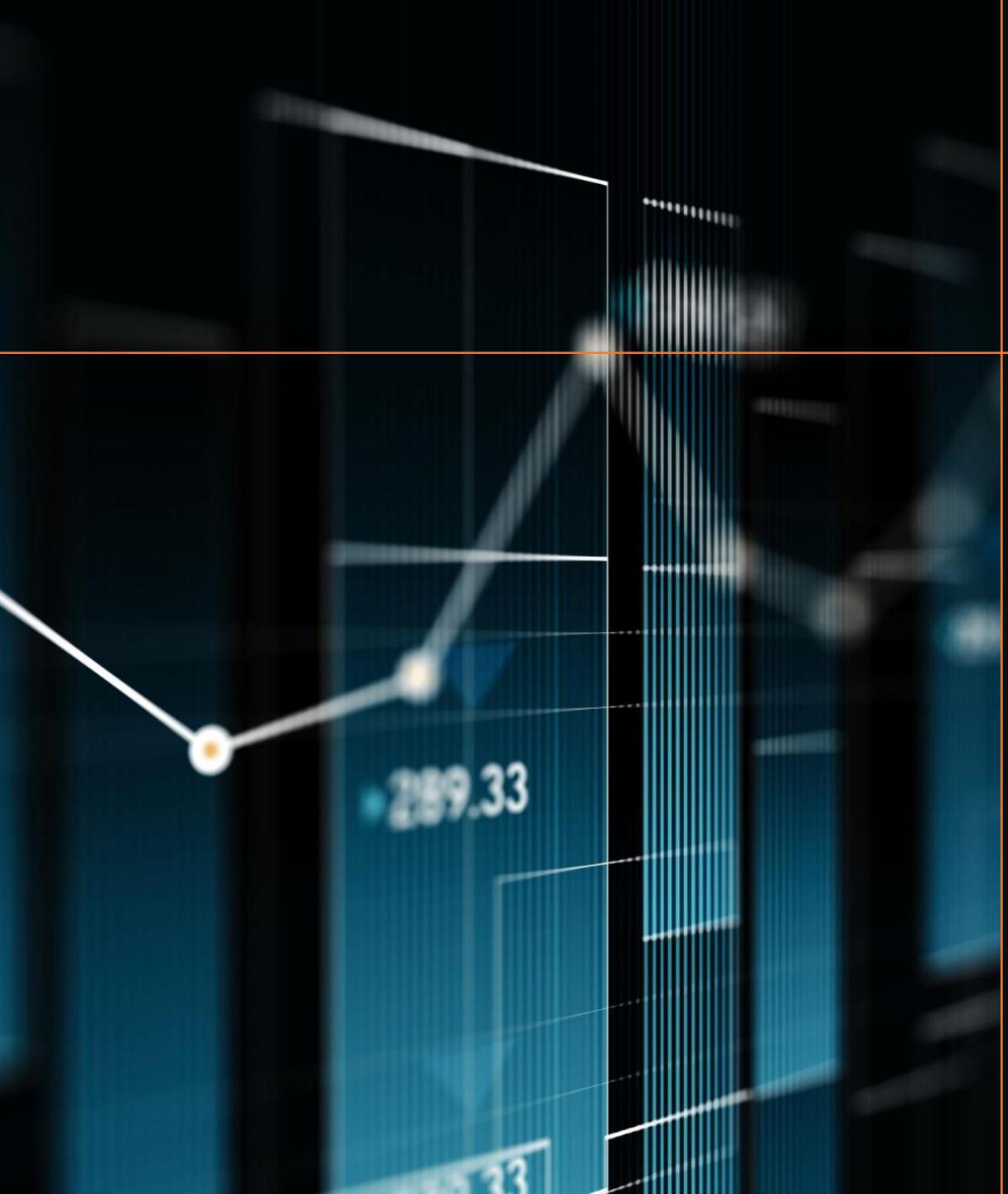
<Igor F>
<03/19/2023>



The left side of the slide features a dark blue background filled with numerous overlapping circles of various sizes and colors. The colors range from deep blues and purples to bright reds, yellows, and greens, creating a dynamic and modern visual texture.

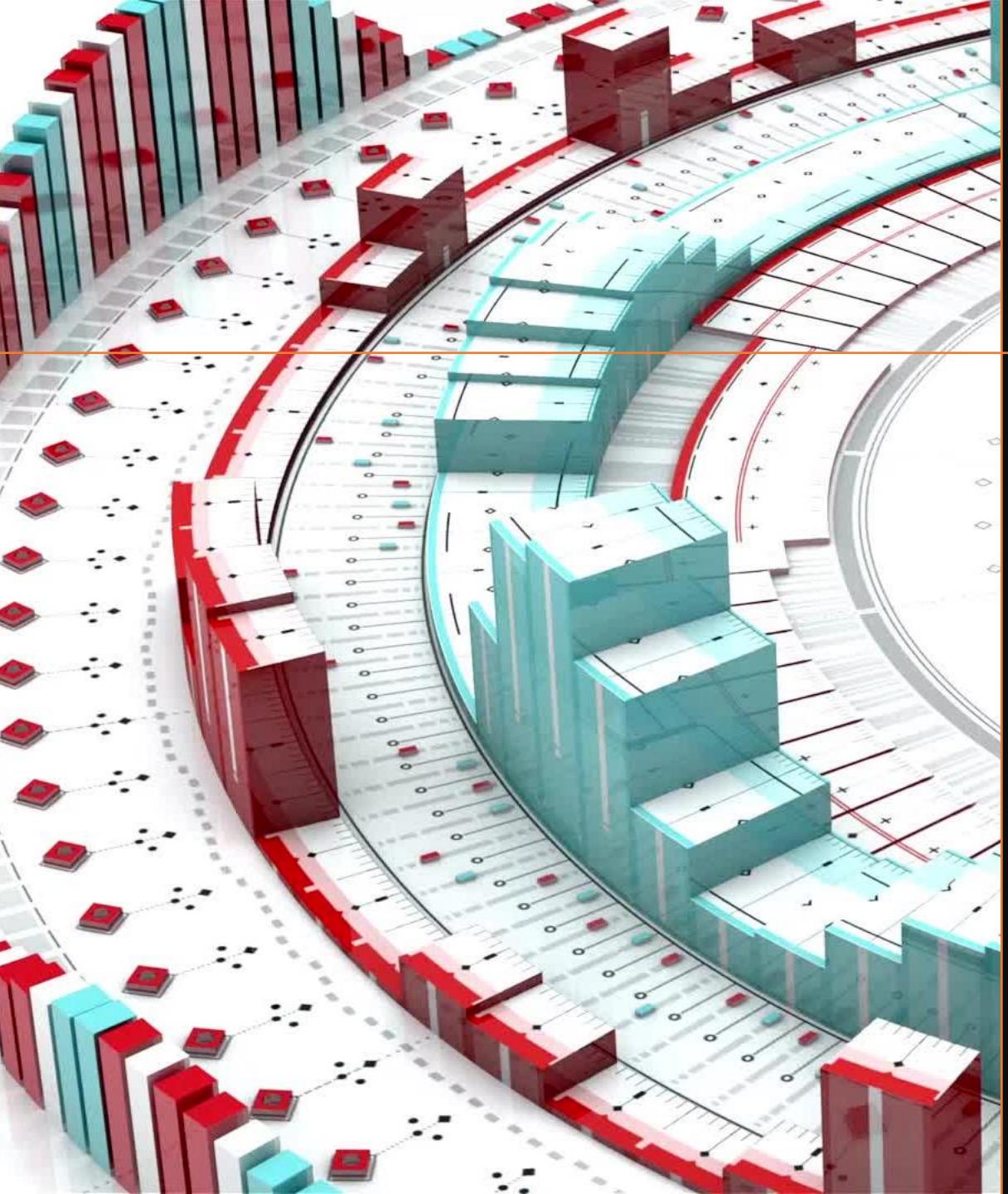
Outline

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



Executive Summary

- Summary of methodologies
 - Data collection utilizing an API
 - Data collection utilizing Web Scraping
 - Data wrangling
 - Exploratory Data Analysis using SQL
 - Exploratory data analysis with data visualization
 - Interactive visual analytics with Folium
 - Machine learning predictions
- Summary of all results
 - Exploratory data analysis results
 - Interactive analytics with screen shots

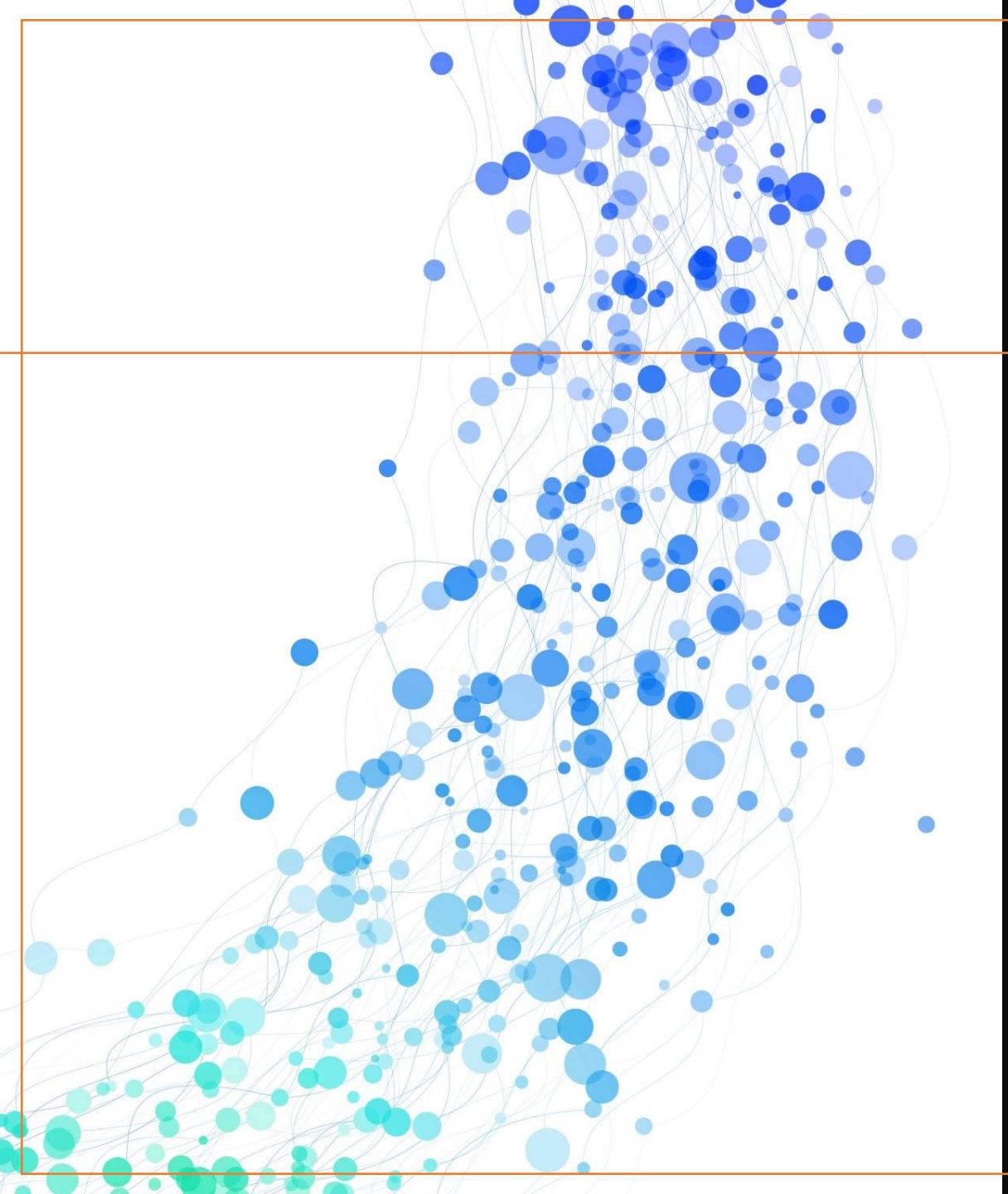


Introduction

- Project background and context
 - Space X 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 Space X 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 space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.
- Problems you want to find answers
 - What factors determine if the rocket will land successfully?
 - The interaction amongst various features that determine the success rate of a successful landing.
 - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

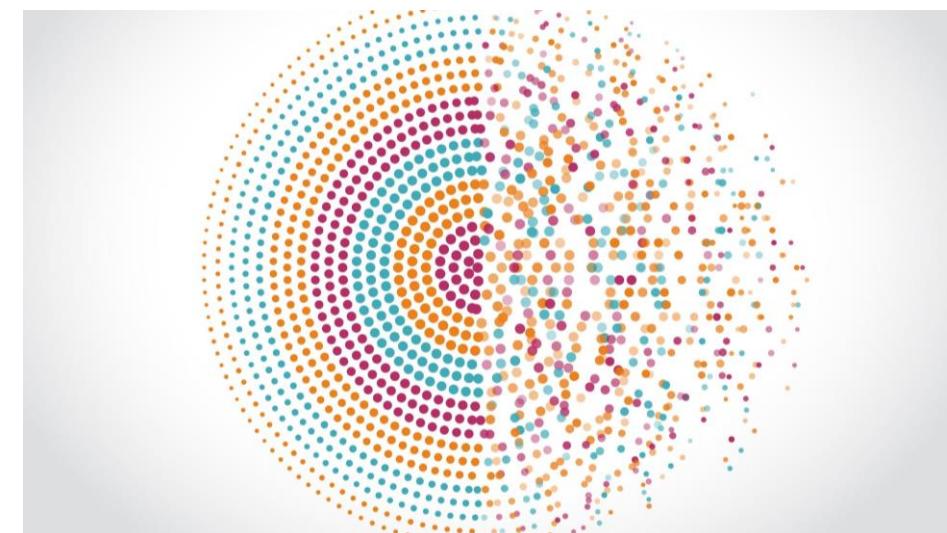
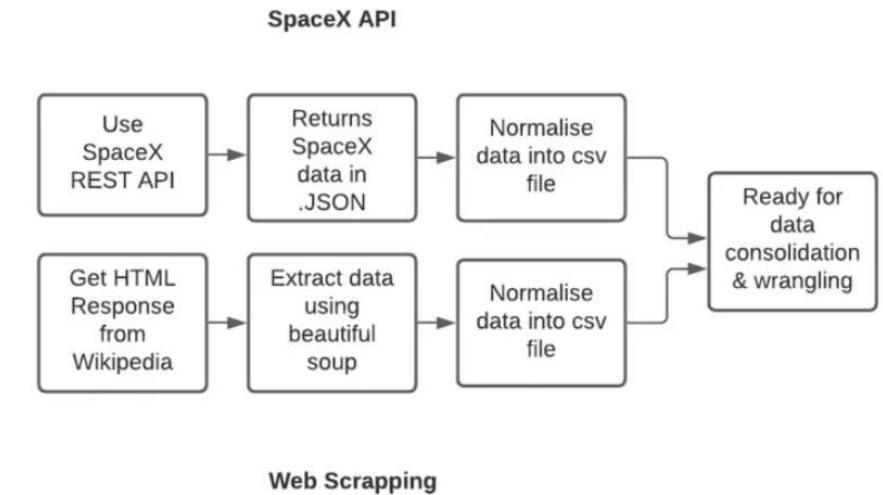


Methodology

- Executive Summary
- Data collection methodology:
 - Data was collected using Spacex API and Web Scraping from Wikipedia
- Perform data wrangling
 - Wrangle data by filtering the data, handling missing values and applying one hot encoding to prepare the data for analysis and modeling
- 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 to predict landing outcomes using classification models. Tune and evaluate models to find best model and parameters

Data Collection

- Describe how data sets were collected.
 - Data collection was done using get request to the SpaceX API.
 - Next, we decoded the response content as a Json using `.json()` function call and turn it into a pandas dataframe using `.json_normalize()`.
 - We then cleaned the data, checked for missing values and fill in missing values where necessary.
 - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
 - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.



Data Collection – SpaceX API

1. Get request for rocket launch data using API

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
In [7]: response = requests.get(spacex_url)
```

2. Use json_normalize method to convert json result to dataframe

```
In [12]: # Use json_normalize method to convert the json result into a dataframe  
  
# decode response content as json  
static_json_df = res.json()
```

```
In [13]: # apply json_normalize  
data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

```
In [30]: rows = data_falcon9['PayloadMass'].values.tolist()[0]  
  
df_rows = pd.DataFrame(rows)  
df_rows = df_rows.replace(np.nan, PayloadMass)  
  
data_falcon9['PayloadMass'][0] = df_rows.values  
data_falcon9
```

- Present your data collection with SpaceX REST calls using key phrases and flowcharts
- Request data from SpaceX API (rocket launch data)
- Decode response using .json() and convert to a dataframe using .json_normalize()
- Request information about the launches from SpaceX API using custom functions
- Create dictionary from the data
- Create dataframe from the dictionary
- Filter dataframe to contain only Falcon 9 launches
- Replace missing values of Payload Mass with calculated .mean()
- Export data to csv file
- Add the GitHub URL of the completed SpaceX API calls notebook (<https://github.com/iforte2/IBM-Capstone>), as an external reference and peer-review purpose

1. Apply HTTP Get method to request the Falcon 9 rocket launch page

```
In [4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

```
In [5]: # use requests.get() method with the provided static_url  
# assign the response to a object  
html_data = requests.get(static_url)  
html_data.status_code
```

```
Out[5]: 200
```

2. Create a BeautifulSoup object from the HTML response

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content  
soup = BeautifulSoup(html_data.text, 'html.parser')
```

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

```
In [7]: # Use soup.title attribute  
soup.title
```

```
Out[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3. Extract all column names from the HTML table header

```
In [10]: 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  
  
element = soup.find_all('th')  
for row in range(len(element)):  
    try:  
        name = extract_column_from_header(element[row])  
        if (name is not None and len(name) > 0):  
            column_names.append(name)  
    except:  
        pass
```

4. Create a dataframe by parsing the launch HTML tables

5. Export data to csv

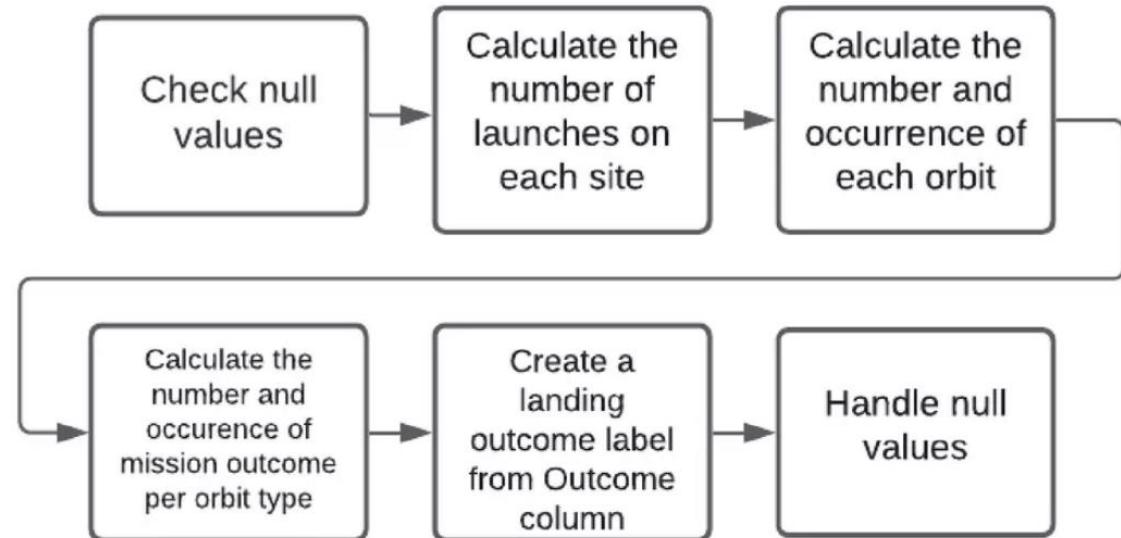
Data Collection - Scraping

- Request data (Falcon 9 launch data) from Wikipedia
- Create BeautifulSoup object from HTML response
- Extract column names from HTML table header
- Collect data from parsing HTML tables
- Create dictionary from the data
- Create dataframe from the dictionary
- Export data to csv file
- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose (<https://github.com/iforte2/IBM-Capstone>)

Data Wrangling

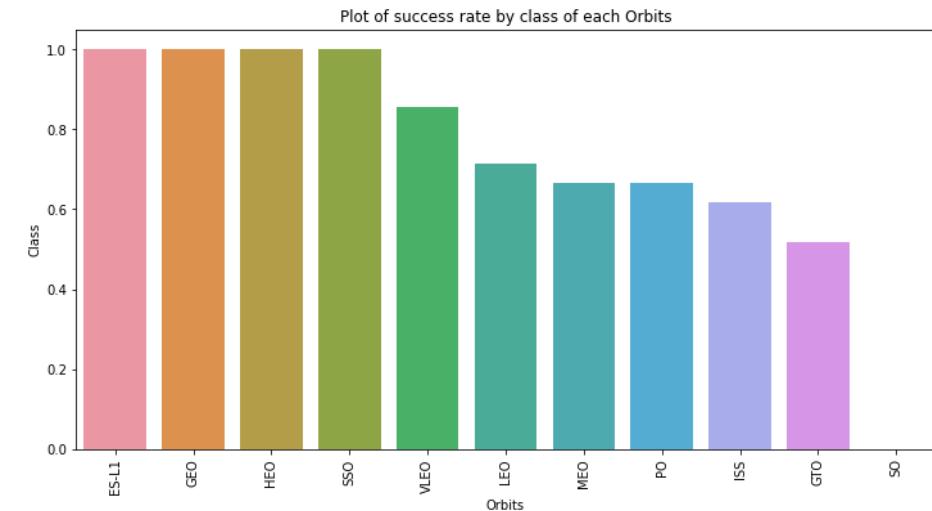
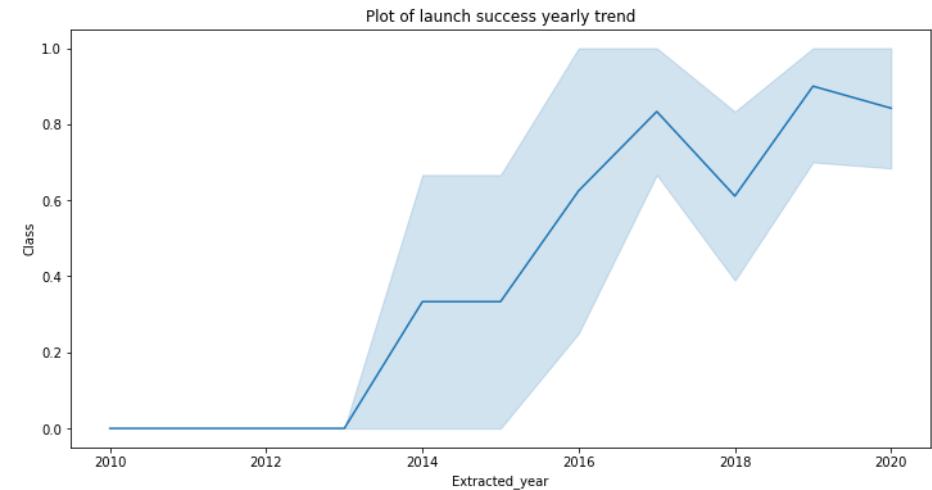
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- Add the GitHub URL of your completed data wrangling related notebooks, as an external reference and peer-review purpose

EDA analysis



EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.
- <https://github.com/iforte2/IBM-Capstone>

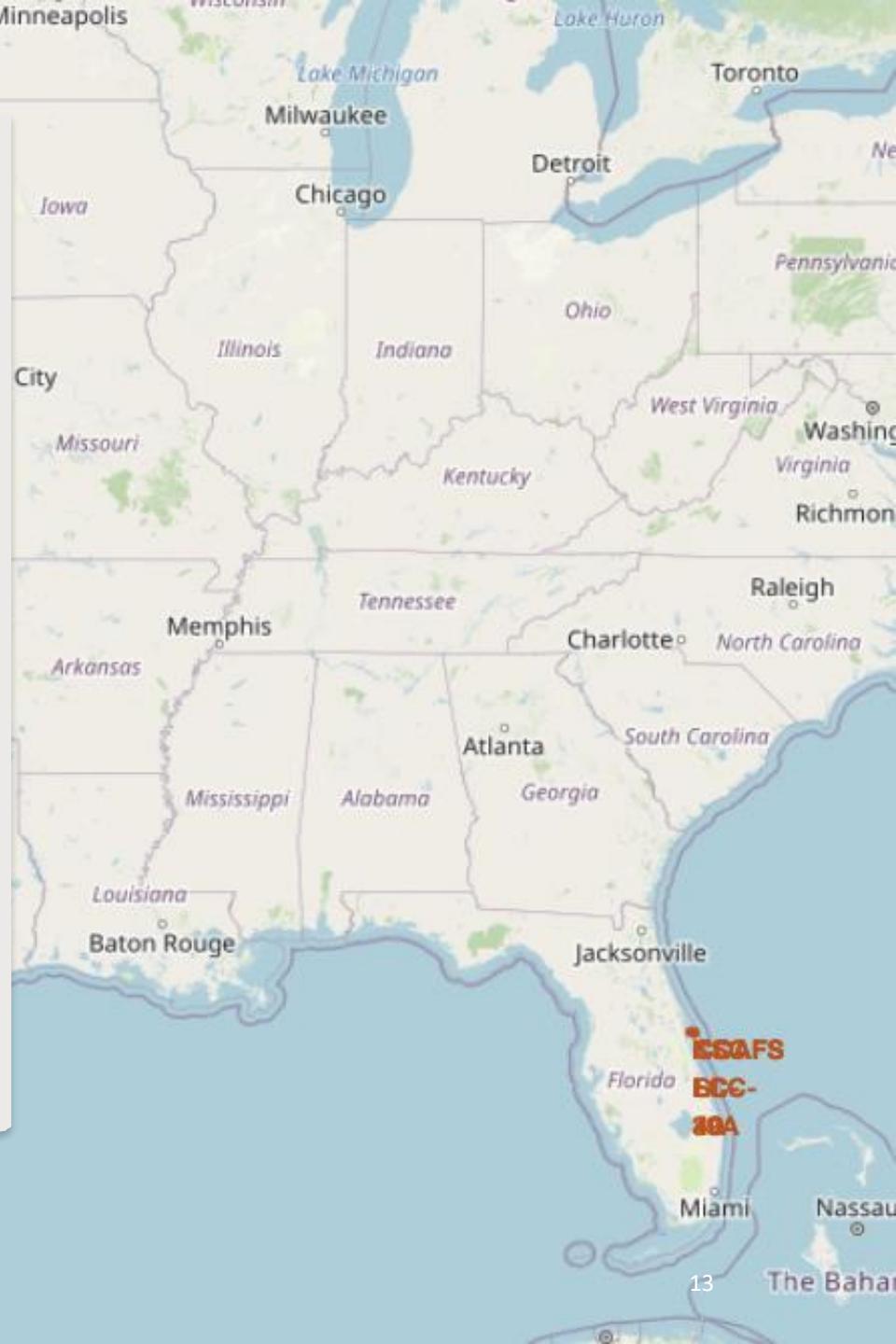


EDA with SQL

- SQL queries performed:
 - Displaying the names of the unique launch sites in the space mission
 - Displaying 5 records where launch sites begin with the string 'KSC'
 - Displaying the total payload mass carried by boosters launched by NASA (CRS)
 - Displaying average payload mass carried by booster version F9 v1.1
 - Listing the date where the successful landing outcome in drone ship was achieved
 - Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
 - Listing the total number of successful and failed mission outcomes
 - Listing the names of the booster versions which carried the maximum payload mass
 - Listing the records with month, successful landing outcomes in the ground pad, booster versions, and launch site months in year 2017
 - Ranking the count of successful landing outcomes between the date 06/04/2010 and 03/20/2017 in descending order
- <https://github.com/iforte2/IBM-Capstone>

Build an Interactive Map with Folium

- **Markers Indicating Launch Sites**
- Added **blue circle** at **NASA Johnson Space Center's coordinate** with a **popup label** showing its name using its latitude and longitude coordinates
- Added **red circles** at **all launch sites coordinates** with a **popup label** showing its name using its latitude and longitude coordinates
- **Map with Folium**
- **Colored Markers of Launch Outcomes**
- Added **colored markers** of **successful(green)** and **unsuccessful(red)** **launches** at each launch site to show which launch sites have high success rates
- **Distances Between a Launch Site to Proximities**
- Added **colored lines** to show distance between launch site **CCAFS SLC-40** and its proximity to the **nearest coastline, railway, highway, and city**

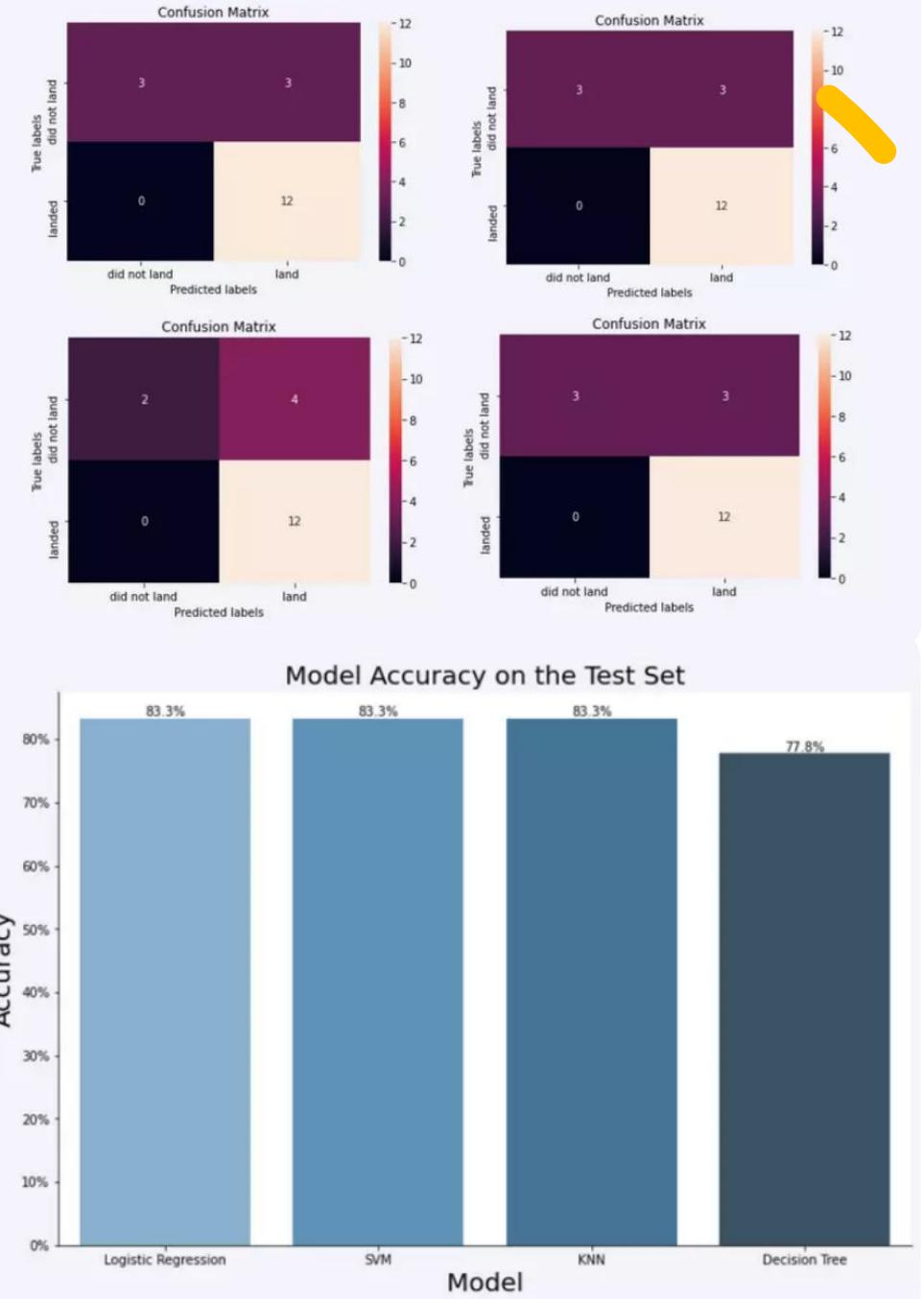


Build a Dashboard with Plotly Dash



- **Dropdown List with Launch Sites**
- Allow user to select all launch sites or a certain launch site
- **Dashboard with Plotly Dash**
- **Slider of Payload Mass Range**
- Allow user to select payload mass range
- **Pie Chart Showing Successful Launches**
- Allow user to see successful and unsuccessful launches as a percent of the total
- **Scatter Chart Showing Payload Mass vs. Success Rate by Booster Version**
- Allow user to see the correlation between Payload and Launch Success
- <https://github.com/iforte2/IBM-Capstone>





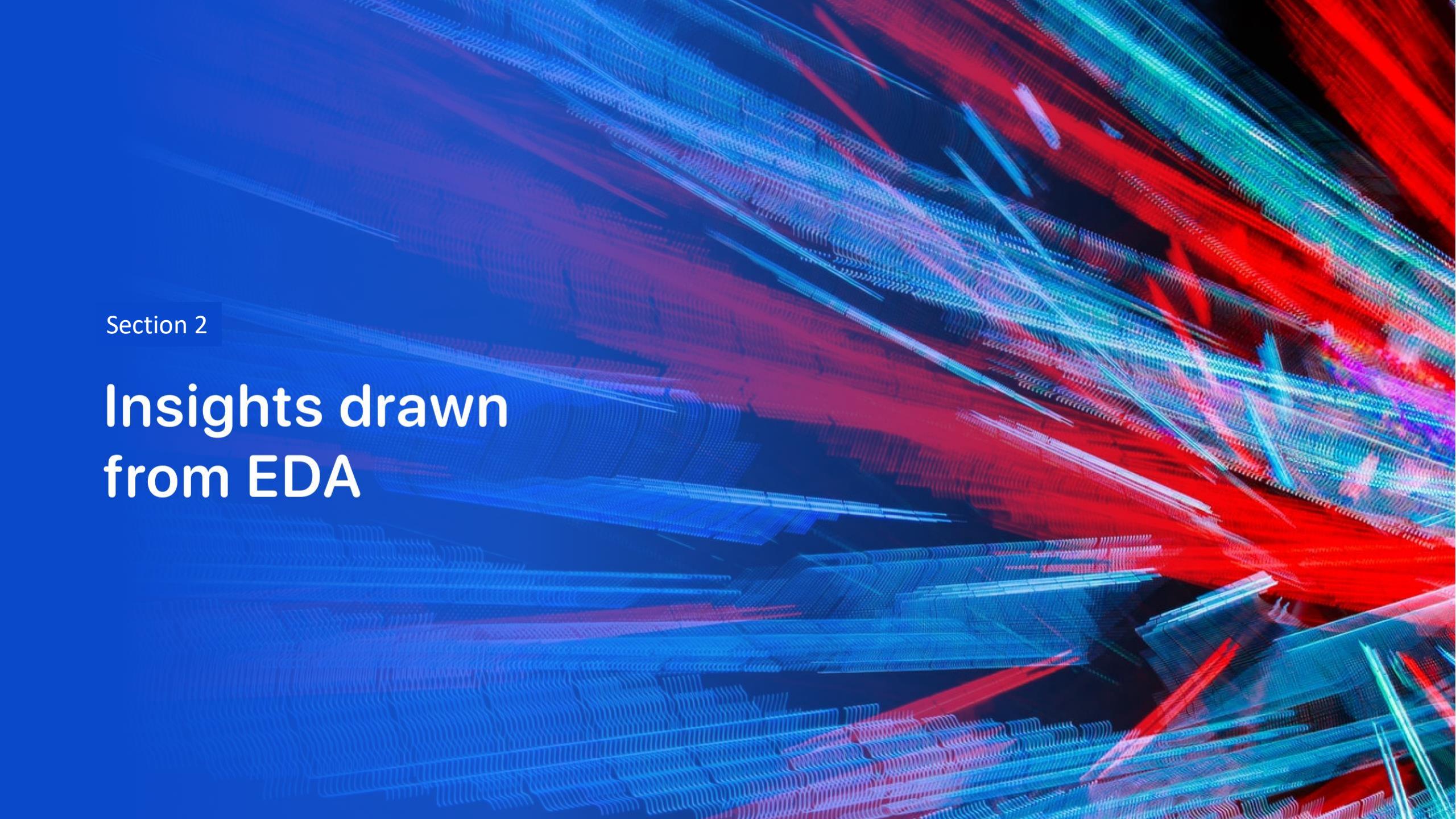
Predictive Analysis (Classification)

- **Charts**
- **Create NumPy array from the Class column**
- **Standardize** the data with StandardScaler. Fit and transform the data.
- **Split** the data using train_test_split
- **Create** a GridSearchCV object with cv=10 for parameter optimization
- **Apply** GridSearchCV on different algorithms: logistic regression (LogisticRegression()), support vector machine (SVC()), decision tree (DecisionTreeClassifier()), K-Nearest Neighbor (KNeighborsClassifier())
- **Calculate** accuracy on the test data using .score() for all models
- **Assess** the confusion matrix for all models
- **Identify** the best model using Jaccard_Score, F1_Score and Accuracy
- The SVM, KNN, and Logistic Regression model achieved the highest accuracy at 83.3%, while the SVM performs the best in terms of Area under the curve *AUC) at 0.958
- <https://github.com/iforte2/IBM-Capstone>



Results

- The SVM, KNN, and Logistic regression models are the best in terms of precision accuracy for this dataset
- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional, and they will eventually perfect the launches
- KSC LC 39A had the most successful launches from all the sites
- Orbit GEO, HEO, SSO, ES L1 has the best success rate
- **Visual Analytics**
- Most launch sites are near the equator, and all are close to the coast
- Launch sites are far enough away from anything a failed launch can damage (city, highway, railway), while still close enough to bring people and material to support launch activities
- **Predictive Analytics**
- Decision Tree model is the best predictive model for the dataset

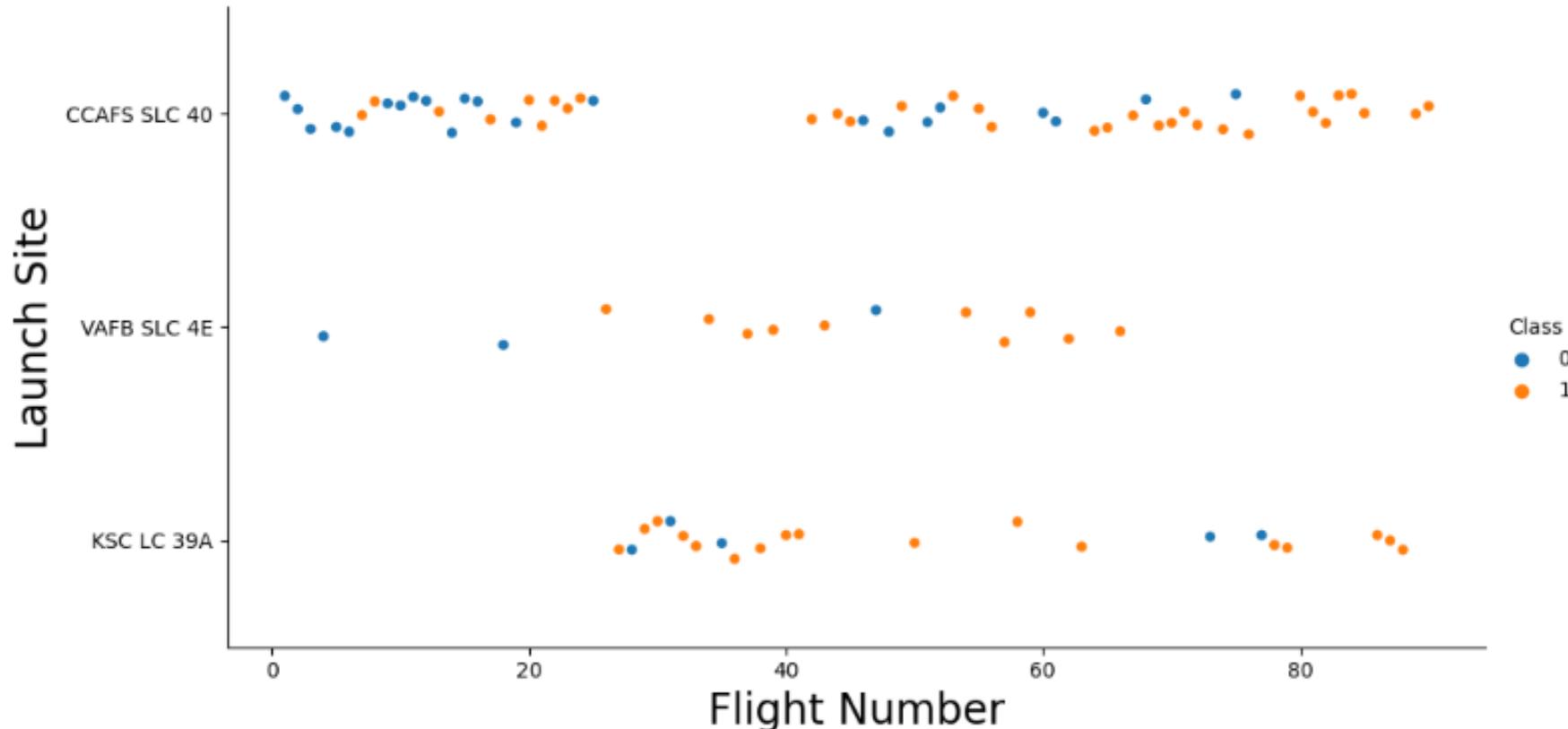
The background of the slide features a complex, abstract digital visualization. It consists of numerous thin, glowing lines that create a sense of depth and motion. The lines are primarily blue and red, with some green and purple highlights. They form a grid-like structure that curves and twists across the frame, resembling a three-dimensional space or a network of data points. The overall effect is futuristic and dynamic.

Section 2

Insights drawn from EDA

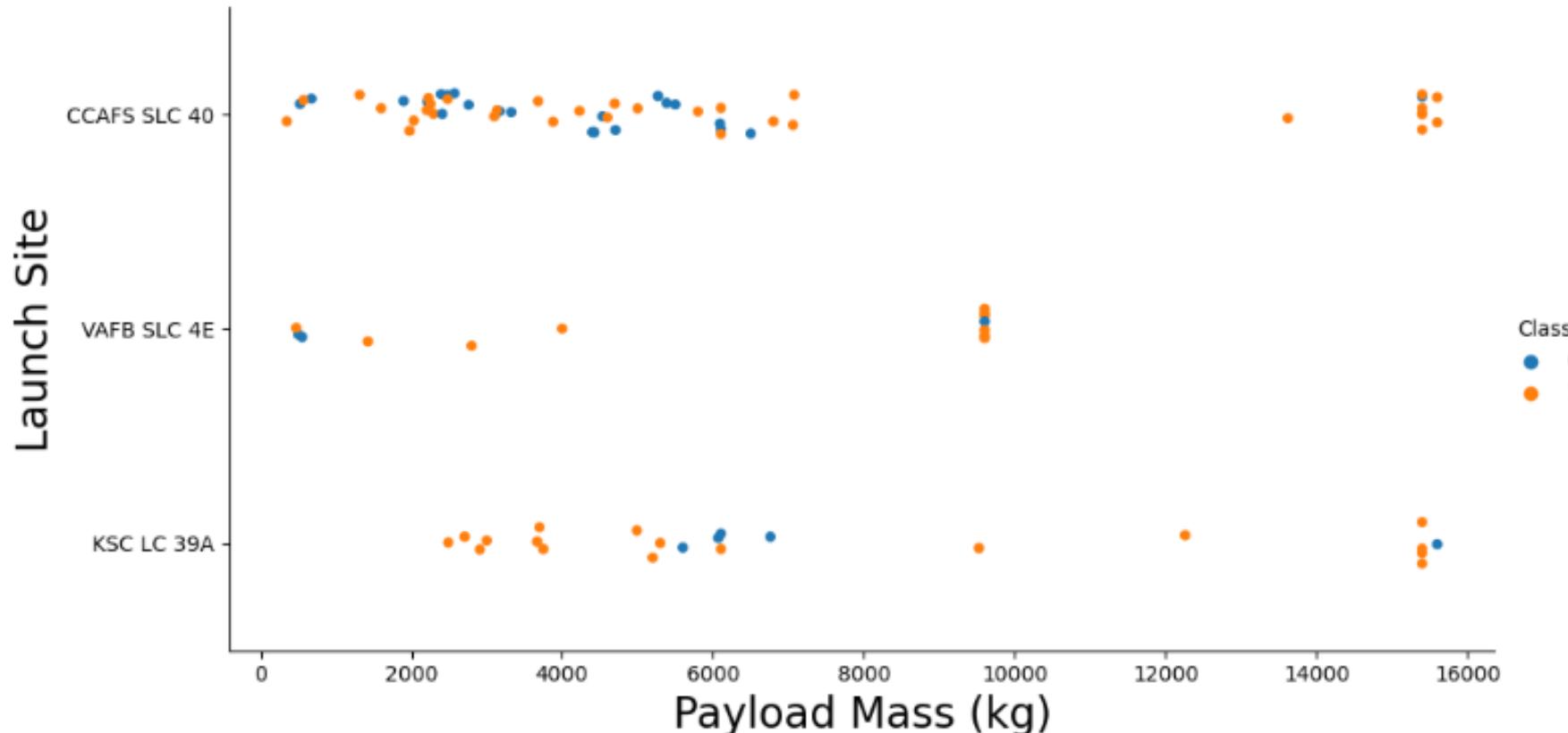
Flight Number vs. Launch Site

- Exploratory Data Analysis
- Earlier flights had a lower success rate (blue = fail)
- Later flights had a higher success rate (orange = success)
- Around half of launches were from CCAFS SLC 40 launch site
- VAFB SLC 4E and KSC LC 39A have higher success rates
- We can infer that new launches have a higher success rate



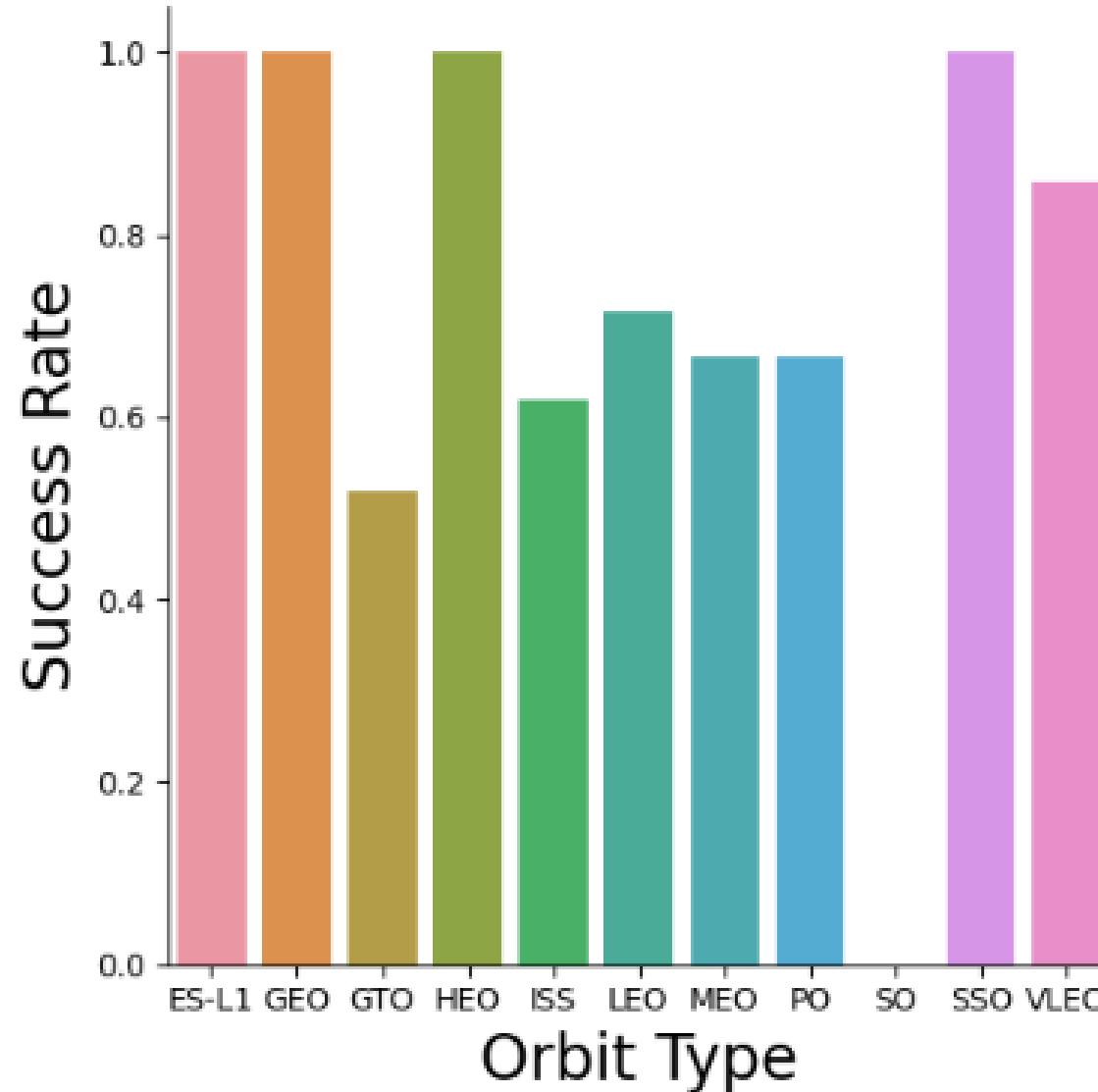
Payload vs. Launch Site

- **Exploratory Data Analysis**
- Typically, the **higher** the payload mass (kg), the **higher** the success rate
- Most launches with a payload greater than 7,000 kg were successful
- KSC LC 39A has a 100% success rate for launches less than 5,500 kg
- VAFB SKC 4E has not launched anything greater than ~10,000 kg



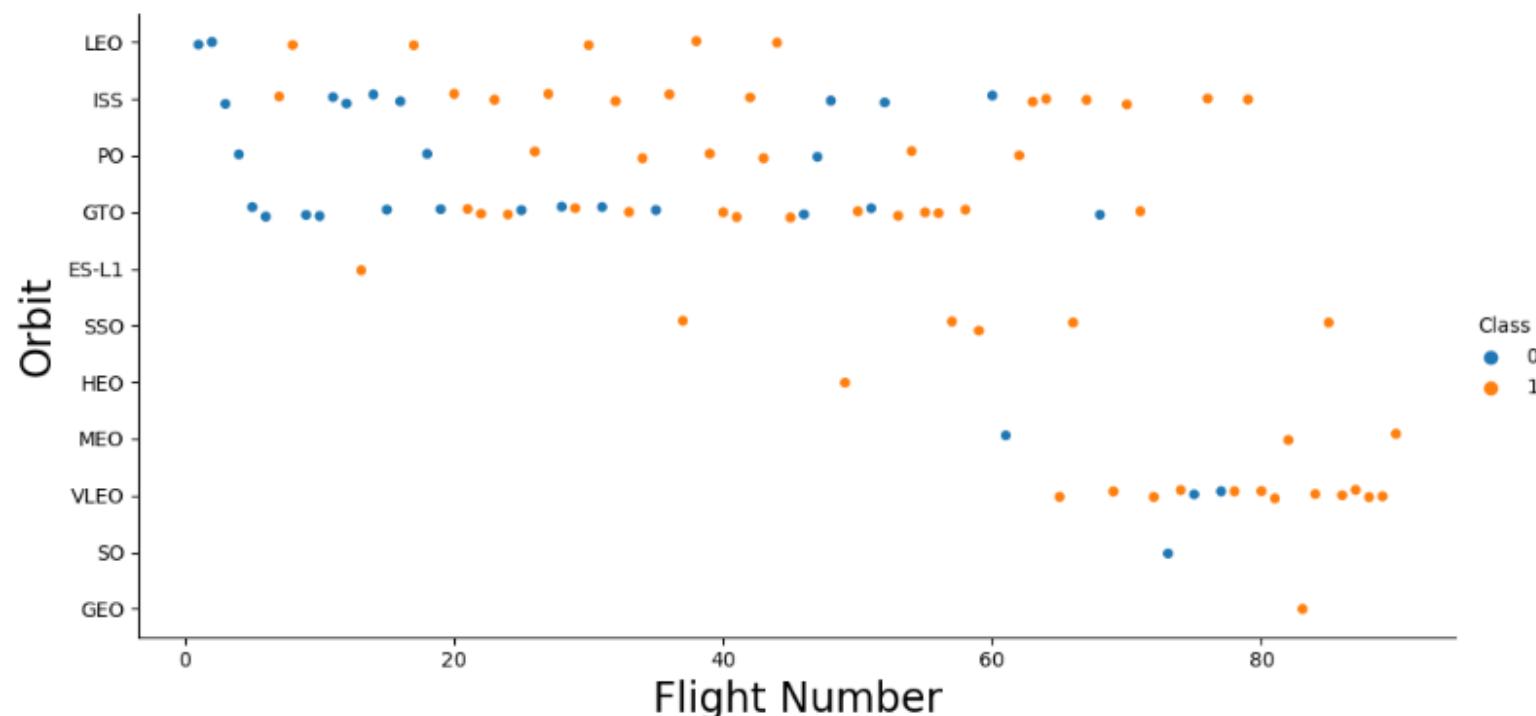
Success Rate vs. Orbit Type

-
- **Exploratory Data Analysis**
 - **100% Success Rate:** ES-L1, GEO, HEO and SSO
 - **50%-80% Success Rate:** GTO, ISS, LEO, MEO, PO
 - **0% Success Rate:** SO



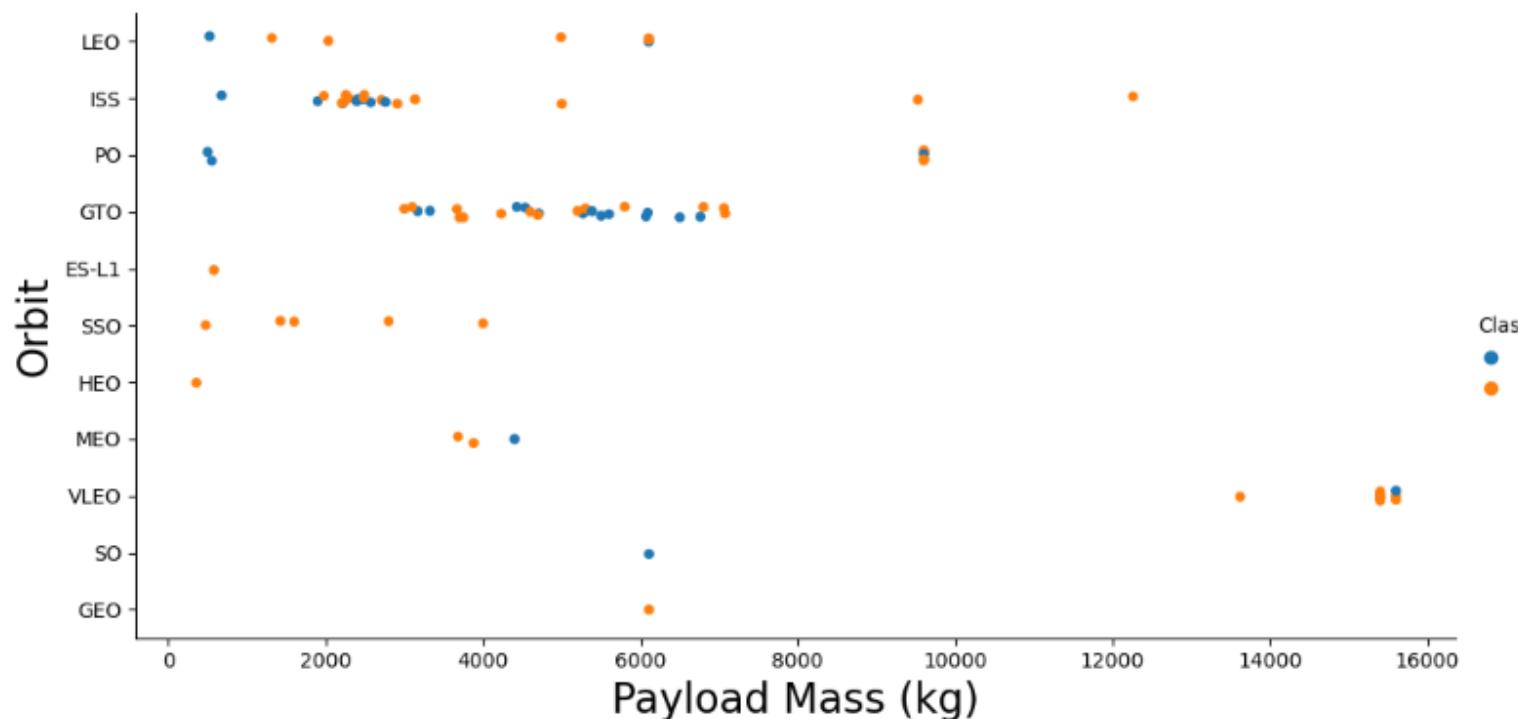
Flight Number vs. Orbit Type

- The success rate typically increases with the number of flights for each orbit
- This relationship is highly apparent for the LEO orbit
- The GTO orbit, however, does not follow this trend



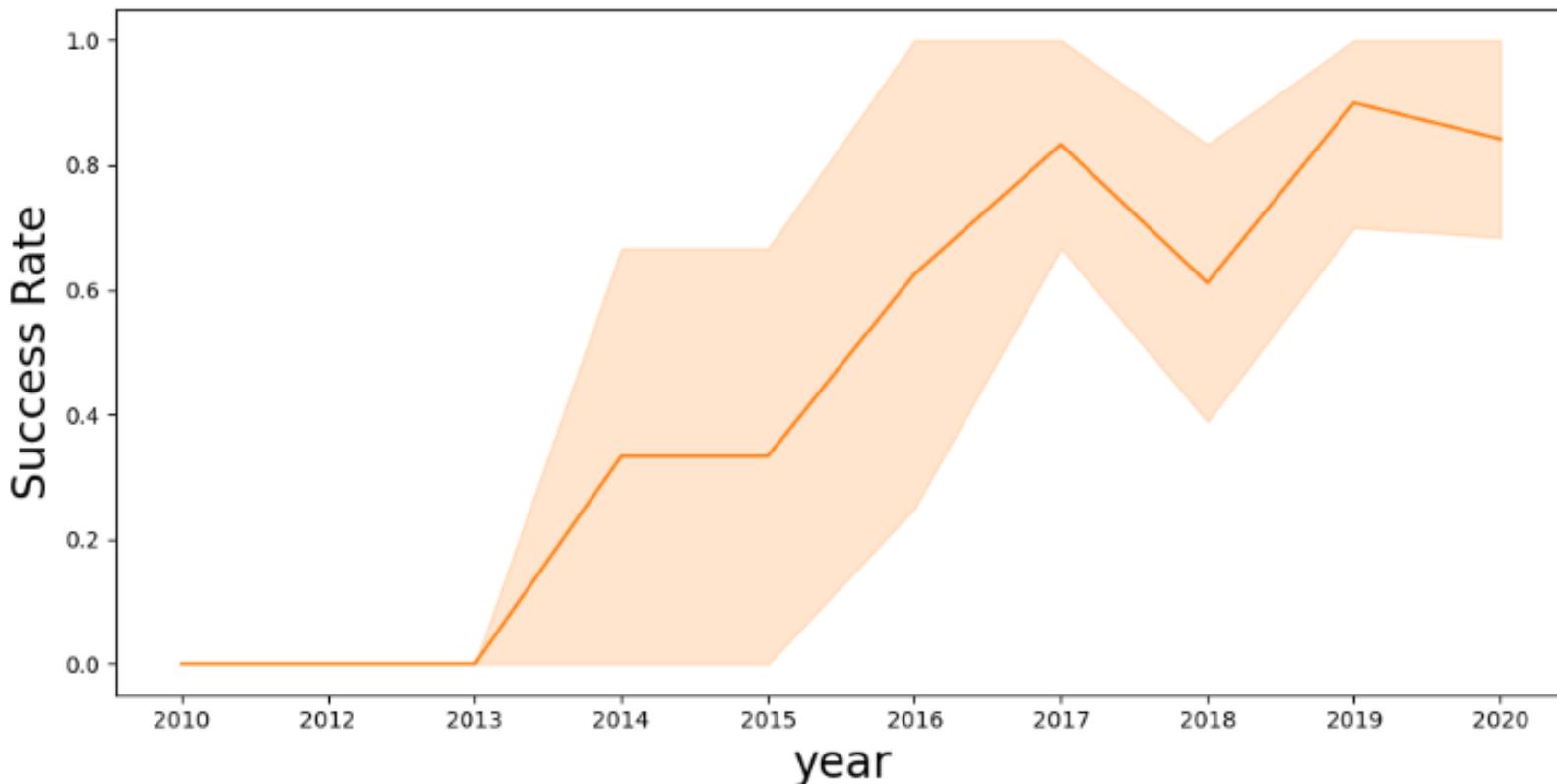
Payload vs. Orbit Type

- Heavy payloads are better with LEO, ISS and PO orbits
- The GTO orbit has mixed success with heavier payloads



Launch Success Yearly Trend

- The success rate improved from 2013-2017 and 2018-2019
- The success rate decreased from 2017-2018 and from 2019-2020
- Overall, the success rate has improved since 2013



All Launch Site Names

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

In [10]:

```
task_1 = ...  
        SELECT DISTINCT LaunchSite  
        FROM SpaceX  
...  
create_pandas_df(task_1, database=conn)
```

Out[10]:

launchsite

0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

- **Launch Site Names**
- CCAFS LC-40
- CCAFS SLC-40
- KSC LC-39A
- VAFB SLC-4E

Launch Site Names Begin with 'CCA'

- We used the query above to display 5 records where launch sites begin with 'CCA'

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

In [11]: task_2 = """
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        """
create_pandas_df(task_2, database=conn)

Out[11]:
```

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 45596 using the query below

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

In [12]:

```
task_3 = """
    SELECT SUM(PayloadMassKG) AS Total_PayloadMass
    FROM SpaceX
    WHERE Customer LIKE 'NASA (CRS)'
    """
create_pandas_df(task_3, database=conn)
```

Out[12]:

total_payloadmass

0	45596
---	-------

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

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

```
In [13]: task_4 = """
    SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
    FROM SpaceX
    WHERE BoosterVersion = 'F9 v1.1'
    """
create_pandas_df(task_4, database=conn)
```

Out[13]: avg_payloadmass

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

```
In [14]: task_5 = ...  
        SELECT MIN(Date) AS FirstSuccessfull_landing_date  
        FROM SpaceX  
        WHERE LandingOutcome LIKE 'Success (ground pad)'  
        ...  
create_pandas_df(task_5, database=conn)
```

```
Out[14]: firstsuccessfull_landing_date  
0 2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- Booster mass greater than 4,000 but less than 6,000
- JSCAT-14, JSCAT-16, SES-10, SES-11 / EchoStar 105

In [15]:

```
task_6 = ''  
        SELECT BoosterVersion  
        FROM SpaceX  
        WHERE LandingOutcome = 'Success (drone ship)'  
              AND PayloadMassKG > 4000  
              AND PayloadMassKG < 6000  
        ...  
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- **Total Number of Successful and Failed Mission Outcomes**
- 1 Failure in Flight
- 99 Success
- 1 Success (payload status unclear)

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

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

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

Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

```
%sql SELECT BOOSTER_VERSION \
FROM SPACEXTBL \
WHERE PAYLOAD_MASS_KG = (SELECT MAX(PAYLOAD_MASS_KG) FROM SPACEXTBL);

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

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

- Showing month, date, booster version, launch site and landing outcome

```
%sql SELECT substr(Date,4,2) as month, DATE, BOOSTER_VERSION, LAUNCH_SITE, [Landing _Outcome] \
FROM SPACEXTBL \
where [Landing _Outcome] = 'Failure (drone ship)' and substr(Date,7,4)='2015';
```

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

month	Date	Booster_Version	Launch_Site	Landing _Outcome
01	10-01-2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	14-04-2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

```
%sql SELECT [Landing _Outcome], count(*) as count_outcomes \
FROM SPACEXTBL \
WHERE DATE between '04-06-2010' and '20-03-2017' group by [Landing _Outcome] order by count_outcomes DESC;
```

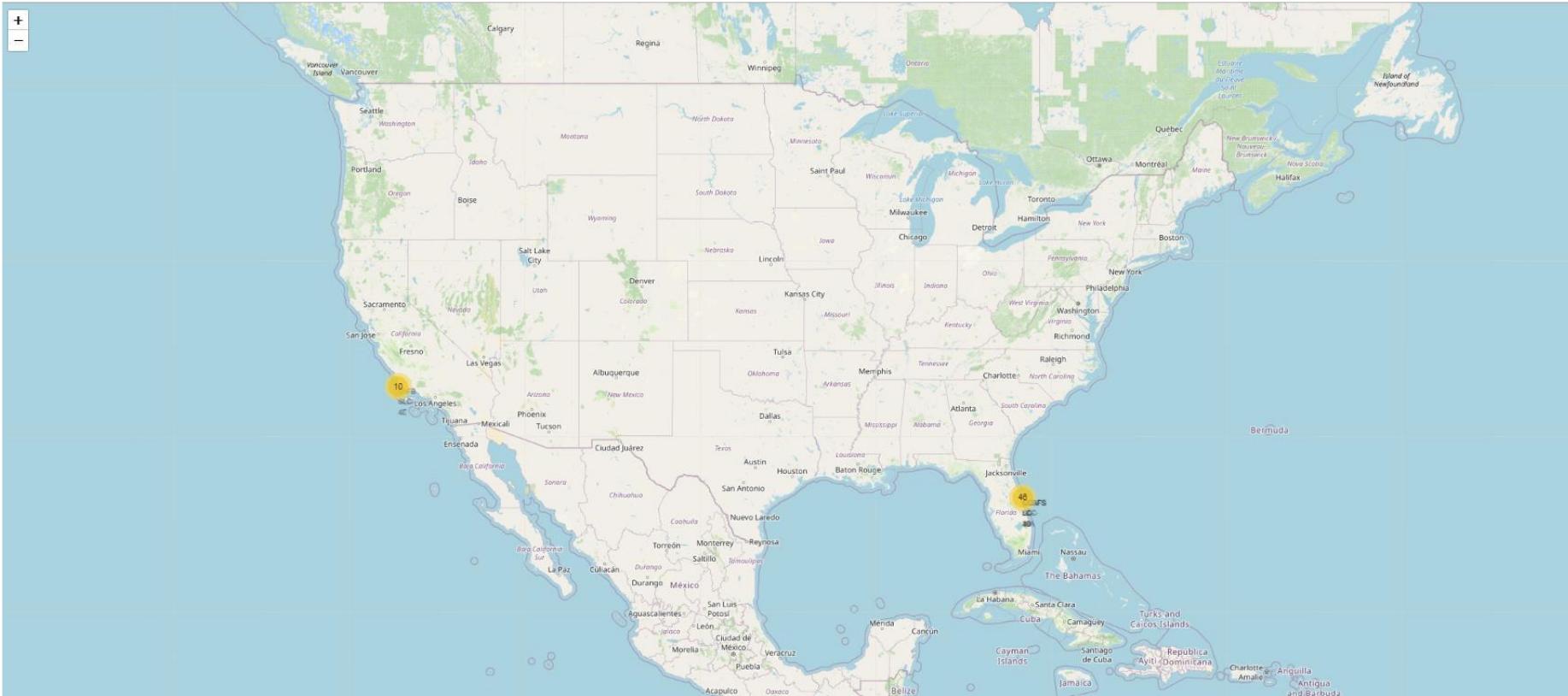
```
* sqlite:///my_data1.db
Done.
```

Landing _Outcome	count_outcomes
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

The background of the slide is a photograph taken from space at night. It shows the curvature of the Earth's horizon against a dark blue sky. Numerous glowing yellow and white points represent city lights, concentrated in coastal and urban areas. In the upper right quadrant, there are bright green and yellow bands of light, likely the Aurora Borealis or Australis. The overall atmosphere is dark and mysterious.

Section 3

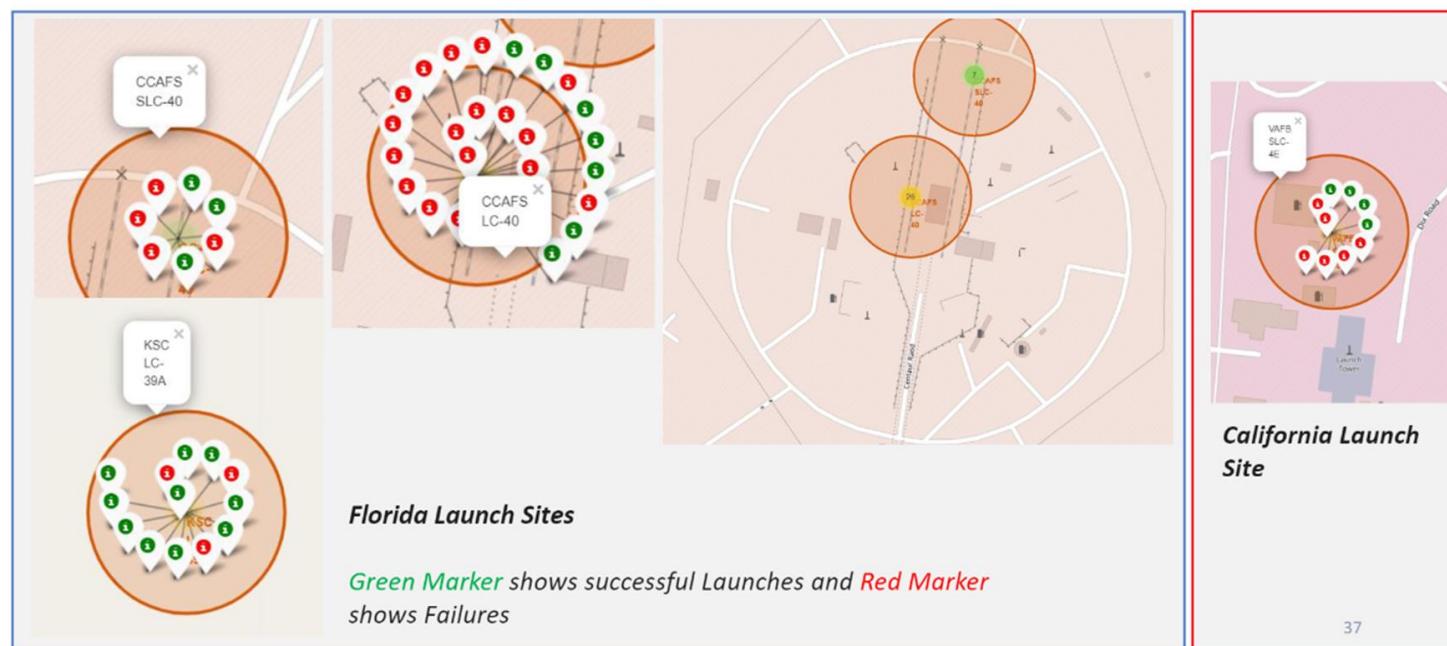
Launch Sites Proximities Analysis



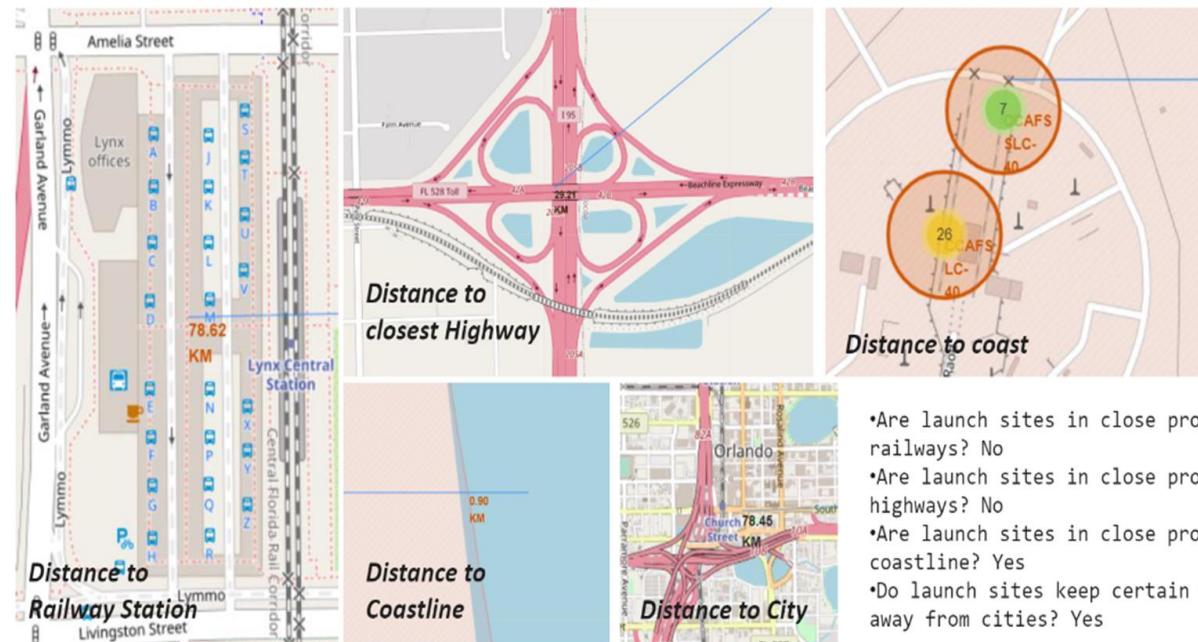
<Folium Map Screenshot 1>

- With Markers
- Near Equator: the closer the launch site to the equator, the easier it is to launch to equatorial orbit, and the more help you get from Earth's rotation for a prograde orbit. Rockets launched from sites near the equator get an additional natural boost due to the rotational speed of earth that helps save the cost of putting in extra fuel and boosters.

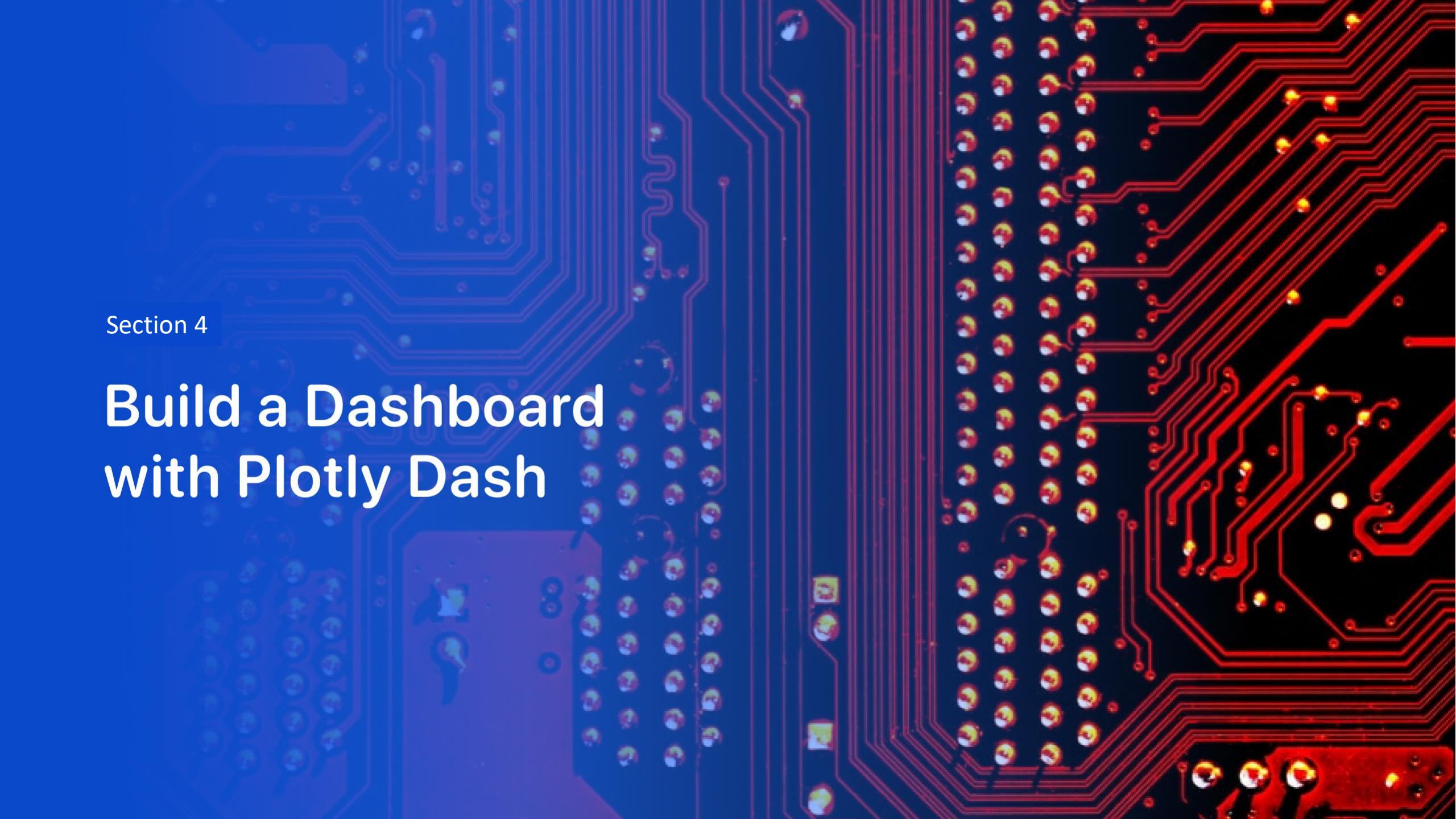
<Folium Map Screenshot 2>



<Folium Map Screenshot 3>



- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

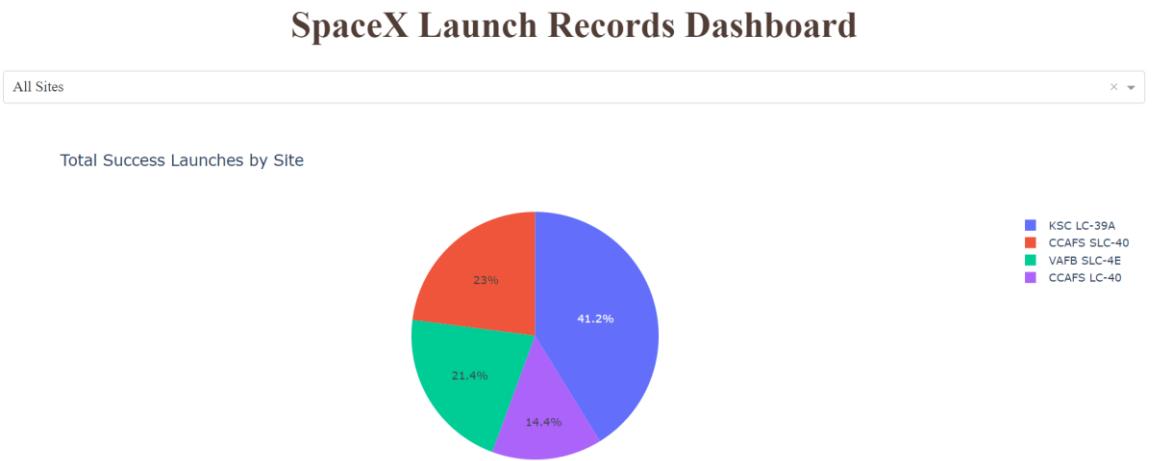


Section 4

Build a Dashboard with Plotly Dash

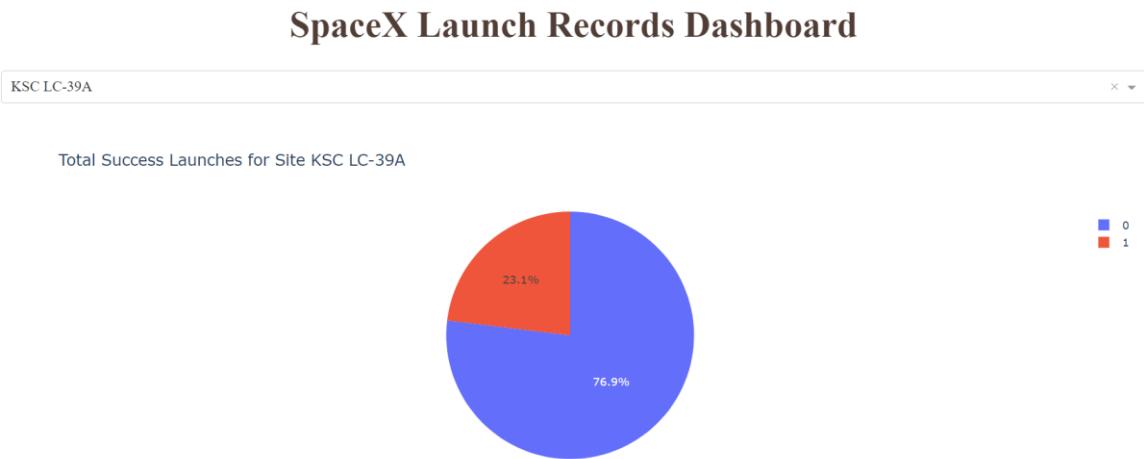
<Dashboard Screenshot 1>

- Success as Percent of Total
- **KSC LC-39A has the most successful launches amongst launch sites (41.2%)**



<Dashboard Screenshot 2>

- Success as Percent of Total
- KSC LC-39A has the highest success rate amongst launch sites (**76.9%**)
- 10 successful launches and 3 failed launches



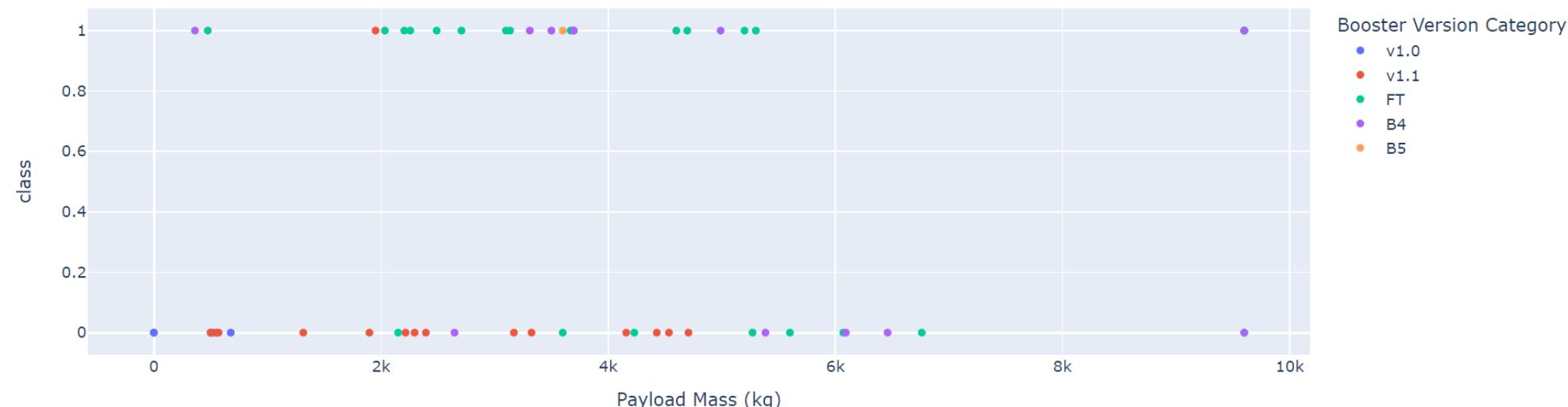
<Dashboard Screenshot 3>

- By Booster Version
- Payloads between 2,000 kg and 5,000 kg have the **highest success rate**
- 1 indicating successful outcome and 0 indicating an unsuccessful outcome

Payload range (Kg):



Correlation Between Payload and Success for All Sites



The background of the slide features a dynamic, abstract design. It consists of several thick, curved lines that transition from a bright yellow at the top right to a deep blue at the bottom left. These lines create a sense of motion and depth, resembling a tunnel or a stylized landscape. The overall effect is modern and professional.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- **Accuracy**
- All the models performed at about the same level and had the same scores and accuracy. This is likely due to the small dataset. The **Decision Tree model** slightly outperformed the rest when looking at `.best_score_`
- `.best_score_` is the average of all cv folds for a single combination of the parameters

```
models = {'KNeighbors':knn_cv.best_score_,  
          'DecisionTree':tree_cv.best_score_,  
          'LogisticRegression':logreg_cv.best_score_,  
          'SupportVector': svm_cv.best_score_}  
  
bestalgorithm = max(models, key=models.get)  
print('Best model is', bestalgorithm, 'with a score of', models[bestalgorithm])  
if bestalgorithm == 'DecisionTree':  
    print('Best params is :', tree_cv.best_params_)  
if bestalgorithm == 'KNeighbors':  
    print('Best params is :', knn_cv.best_params_)  
if bestalgorithm == 'LogisticRegression':  
    print('Best params is :', logreg_cv.best_params_)  
if bestalgorithm == 'SupportVector':  
    print('Best params is :', svm_cv.best_params_)  
  
Best model is DecisionTree with a score of 0.8732142857142856  
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

	LogReg	SVM	Tree	KNN
Jaccard_Score	0.800000	0.800000	0.800000	0.800000
F1_Score	0.888889	0.888889	0.888889	0.888889
Accuracy	0.833333	0.833333	0.833333	0.833333

Confusion Matrix

A confusion matrix summarizes the performance of a classification algorithm

- All the confusion matrices were identical
- The fact that there are false positives (Type 1 error) is not good
- Confusion Matrix Outputs: 12 True positive

3 True negative

3 False positive

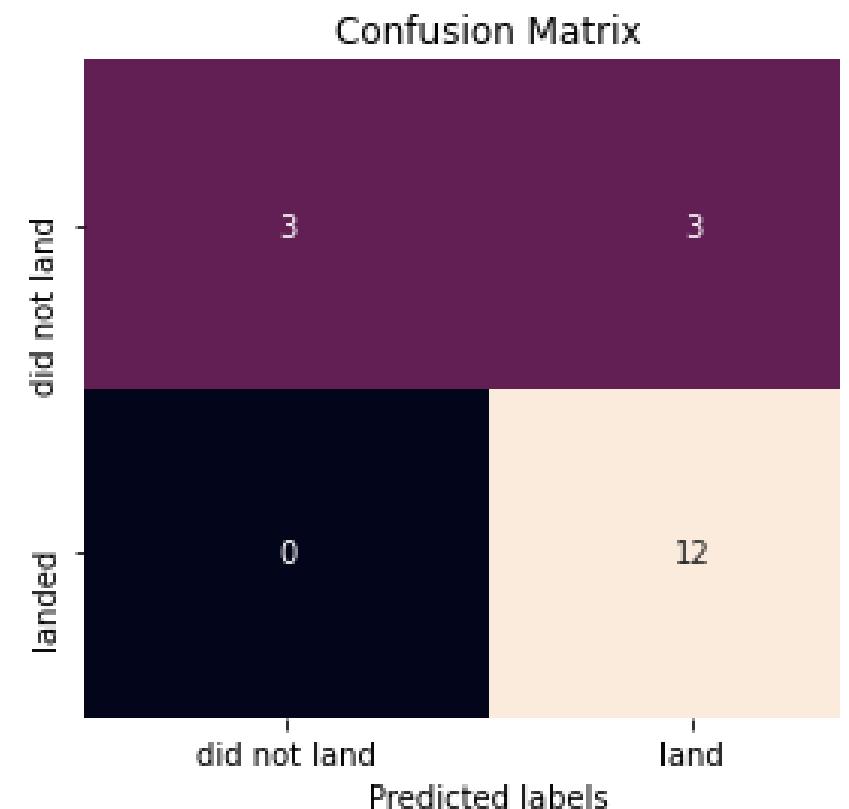
0 False Negative

- **Precision**= $TP / (TP + FP)$ $12 / 15 = .80$

- **Recall**= $TP / (TP + FN)$ $12 / 12 = 1$

- **F1 Score**= $2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$ $2 * (.8 * 1) / (.8 + 1) = .89$

- **Accuracy**= $(TP + TN) / (TP + TN + FP + FN)$ $.833$



Conclusions

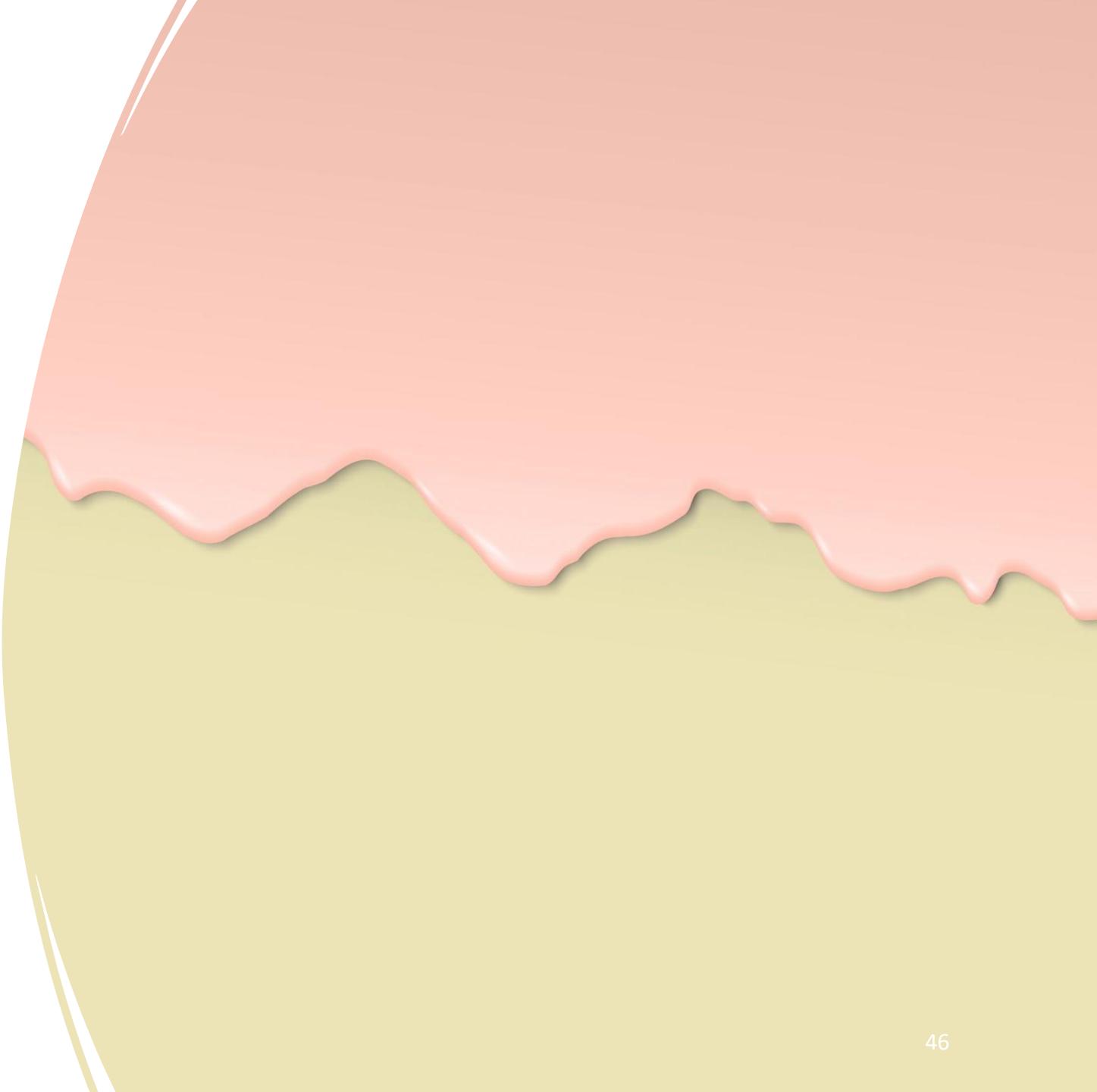
Research

- **Model Performance:** The models performed similarly on the test set with the decision tree model slightly outperforming
- **Equator:** Most of the launch sites are near the equator for an additional natural boost -due to the rotational speed of earth -which helps save the cost of putting in extra fuel and boosters
- **Coast:** All the launch sites are close to the coast
- **Launch Success:** Increases over time
- **KSC LC-39A:** Has the highest success rate among launch sites. Has a 100% success rate for launches less than 5,500 kg
- **Orbits:** ES-L1, GEO, HEO, and SSO have a 100% success rate
- **Payload Mass:** Across all launch sites, the higher the payload mass (kg), the higher the success rate

Appendix

Things to Consider

- **Dataset:** A larger dataset will help build on the predictive analytics results to help understand if the findings can be generalizable to a larger data set
- **Feature Analysis / PCA:** Additional feature analysis or principal component analysis should be conducted to see if it can help improve accuracy
- **XGBoost:** Is a powerful model which was not utilized in this study. It would be interesting to see if it outperforms the other classification models



Thank you!

