IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ekram Alvi
5/15/2024

Outline

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

Executive Summary

- Summary of Methodologies:
  - Data Collection: Utilized the SpaceX REST API and web scraping to compile comprehensive launch data.
  - Data Wrangling: Standardized and cleaned data to ensure quality for analysis.
  - Exploratory Data Analysis (EDA): Employed visual and statistical methods to uncover patterns and insights.
  - Interactive Visualizations: Developed an interactive dashboard using Plotly Dash and Folium for dynamic data exploration.
  - Predictive Analysis: Built and optimized classification models using machine learning techniques to predict launch success.
- Summary of Results:
  - Data Insights: Revealed critical dependencies such as payload mass effects on launch success and variations in success rates across different launch sites.
  - Interactive Dashboard: Enabled users to filter and visualize data interactively, enhancing understanding of launch outcomes.
  - Model Performance: Decision Tree Classifier emerged as the best model with high accuracy, effectively predicting launch outcomes.

Introduction

Project Background and Context

- **SpaceX's Innovation:** SpaceX has revolutionized space travel by developing the reusable Falcon 9 rocket, drastically reducing costs compared to traditional, single-use rockets.
- **Economic Impact:** The ability to reuse the first stage of Falcon 9 significantly lowers the launch cost, making space travel more accessible and competitive.

Problems We Wanted Answers For

- **Predictive Analysis:** Can we accurately predict the successful landing of Falcon 9's first stage using data science methodologies?
- **Cost Implications:** How does the prediction of Falcon 9's first stage landing impact the financial modeling and competitive pricing of commercial space launches?

Section 1

Methodology

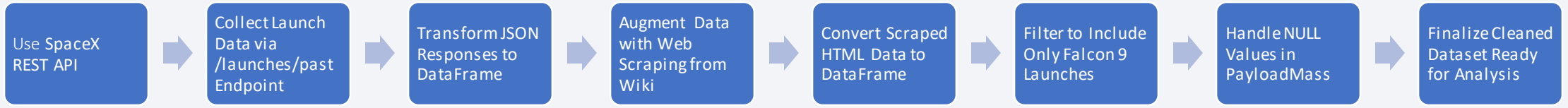
Methodology

Executive Summary

- Data collection methodology:
  - Utilized SpaceX API to systematically gather detailed Falcon 9 launch data to ensure a comprehensive and up-to-date dataset for analysis.
- Perform data wrangling
  - Performed data cleaning and preprocessing, including handling missing values and standardizing data formats, to ensure data quality and usability for predictive 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

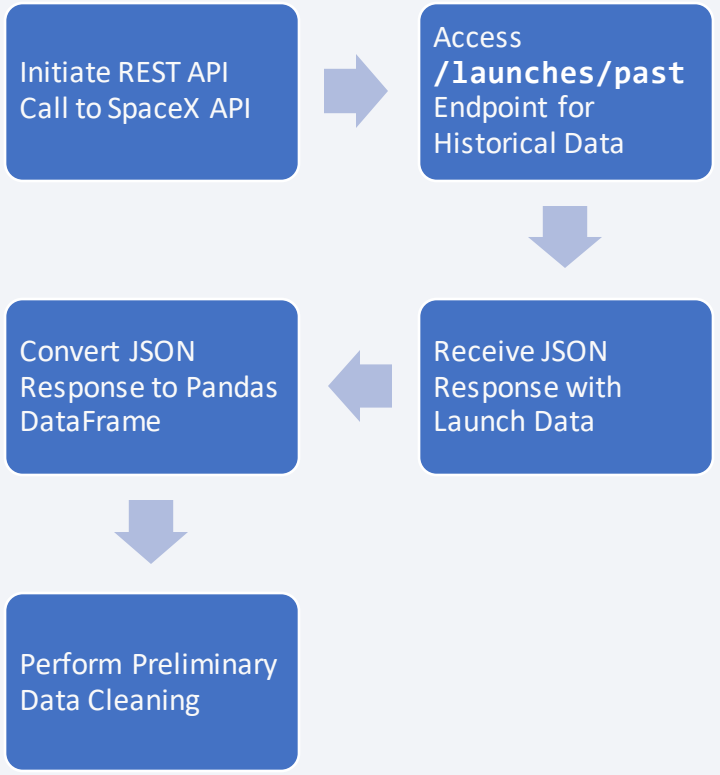
Data Collection

- **API Utilization**: Utilized the SpaceX REST API to systematically collect comprehensive launch data, including rocket specifications, payload details, and landing outcomes.
- **Data Transformation**: Employed methods to convert JSON responses from the API into structured Pandas data frames, facilitating analysis.
- **Web Scraping for Augmentation**: Augmented API data by scraping additional Falcon 9 launch information from Wiki pages using Python's BeautifulSoup package.
- **Focused Data Selection**: Implemented filtering to exclusively include Falcon 9 launches, discarding irrelevant Falcon 1 launch data.
- **Handling Data Imperfections**: Addressed NULL values, specifically in the PayloadMass column, by calculating and imputing the mean value, while accepting NULLs in the LandingPad column as indicative of no landing pad use.



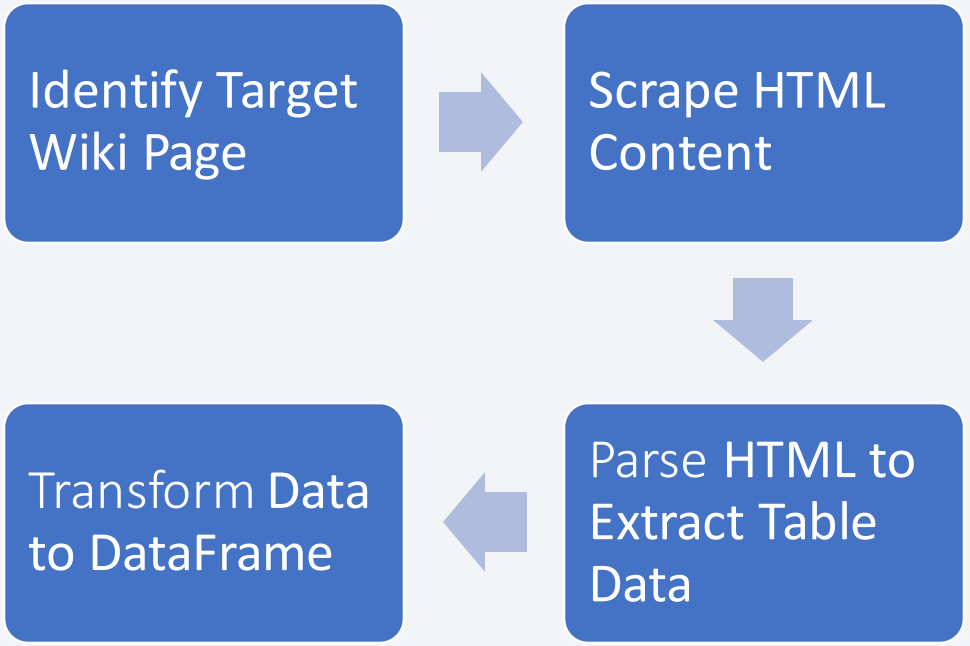
Data Collection – SpaceX API

- **SpaceX REST API Calls:** Utilized the SpaceX REST API to systematically gather comprehensive launch data, focusing on Falcon 9 missions.
- **Data Transformation and Cleaning:** Converted JSON responses into Pandas DataFrames and performed preliminary cleaning to ensure data quality for analysis
- https://github.com/generalmarke/ibmcapstoneproject/blob/5f4e9f2934617da618f89e7560989fbde4b9b557/jupyter-labs-spacex-data-collection-api.ipynb



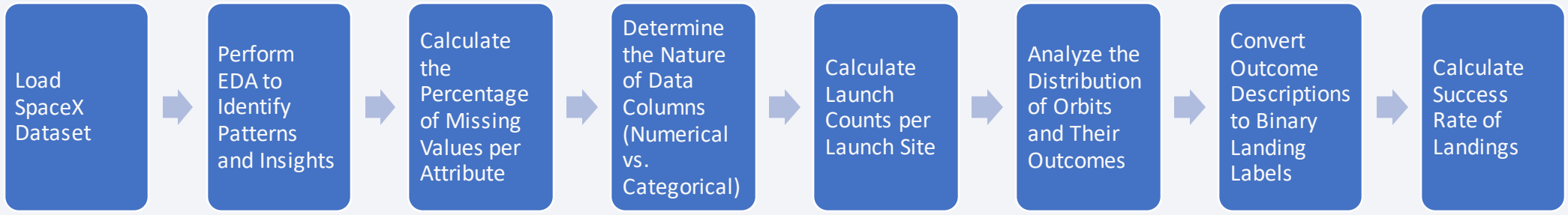
Data Collection - Scraping

- **Web Scraping for Data Augmentation:** Used Python's BeautifulSoup package to scrape Falcon 9 launch data from Wikipedia, complementing the API data.
- **Data Transformation:** Transformed the scraped HTML data into a structured Pandas DataFrame for analysis.
- https://github.com/generalmarke/ibmcapstoneproject/blob/5f4e9f2934617da618f89e7560989fbde4b9b557/jupyter-labs-webscraping.ipynb



Data Wrangling

- **Exploratory Data Analysis (EDA)**: Conducted EDA to uncover patterns in SpaceX launch data, focusing on outcomes related to Falcon 9 booster landings.
- **Label Determination**: Transformed various landing outcomes into a binary classification (1 for successful landings and 0 for unsuccessful attempts) to serve as training labels for supervised models.
- https://github.com/generalmarke/ibmcapstoneproject/blob/5f4e9f2934617da618f89e7560989fbde4b9b557/labs-jupyter-spacex-Data%20wrangling.ipynb



EDA with Data Visualization

- **Flight Number vs. Payload Mass:**
  - **Chart:** Scatter Plot with Outcome Overlay
  - **Insight:** Increases in Flight Number improve landing success; heavier payloads lower this rate.
- **Launch Site Success Rates:**
  - **Chart:** Bar Chart
  - **Insight:** Varying success rates (CCAFS LC-40 at 60%, KSC LC-39A and VAFB SLC 4E at 77%) suggest site-specific influences.
- **Flight Number vs. Launch Site:**
  - **Chart:** Categorical Plot
  - **Insight:** Demonstrates how experience and launch site correlate with success rates.
- **Payload vs. Launch Site:**
  - **Chart:** Scatter Plot
  - **Insight:** Absence of heavy payloads (>10,000kg) at VAFB SLC hints at launch capacity or mission constraints.

EDA with Data Visualization

- **Success Rate by Orbit Type:**
  - **Chart:** Bar Chart
  - **Insight:** Higher success in LEO, Polar, and ISS orbits, underscoring the significance of orbit type on landing outcomes.
- **FlightNumber vs. Orbit Type:**
  - **Chart:** Scatter Plot
  - **Insight:** Success in LEO correlates with flight count, unlike in GTO, revealing orbit-specific success factors.
- **Payload vs. Orbit Type:**
  - **Chart:** Scatter Plot
  - **Insight:** Heavy payloads fare better in Polar, LEO, and ISS orbits; mixed results in GTO orbit reflect the complexity of orbital dynamics.
- **Yearly Success Trend:**
  - **Chart:** Line Chart
  - **Insight:** Steady increase in success rate from 2013 to 2020 showcases SpaceX's progress and operational enhancements.
- https://github.com/generalmarke/ibmcapstoneproject/blob/5f4e9f2934617da618f89e7560989fbde4b9b557/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- **Task 1:** Identify all unique launch sites used in SpaceX missions.
- **Task 2:** Retrieve records for launch sites that start with 'CCA', displaying the first five.
- **Task 3:** Calculate the total payload mass for missions launched by NASA under the CRS program.
- **Task 4:** Determine the average payload mass carried by the Falcon 9 version 1.1 booster.
- **Task 5:** Find the date of the first successful landing on a ground pad.
- **Task 6:** List boosters that successfully landed on a drone ship with a payload mass between 4,000 and 6,000 kg.

EDA with SQL

- **Task 7:** Count the total number of missions with successful and failed outcomes.
- **Task 8:** Identify booster versions that have carried the maximum payload mass.
- **Task 9:** Display records from 2015 showing the month, failure outcomes on drone ships, booster versions, and launch sites.
- **Task 10:** Rank the frequency of different landing outcomes between June 4, 2010, and March 20, 2017, in descending order.
- https://github.com/generalmarke/ibmcapstoneproject/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

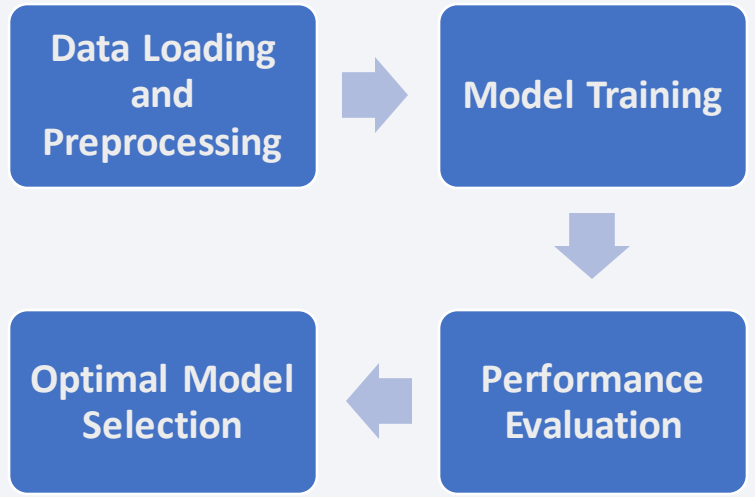
- **Markers:**
  - Launch sites with color-coded markers for success (green) and failure (red).
  - Additional markers for significant points like coastlines and NASA Johnson Space Center.
- **Circles:**
  - Highlighted circles for enhanced visibility at critical locations such as launch sites.
- **PolyLines:**
  - Lines illustrating distances from launch sites to nearby important features (coastlines, railways).
- **Marker Clusters:**
  - Grouped markers at the same coordinates to declutter the map and improve user navigation.
- **Purpose of Map Objects:**
  - Markers and Circles: Quick identification and status overview of launch sites and other key points.
  - PolyLines: Visual representation of geographical proximities relevant to launch logistics and safety.
  - Marker Clusters: Efficient management of multiple markers, simplifying the visual experience.
- https://github.com/generalmarke/ibmcapstoneproject/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- **Launch Site Filter:**
  - Dropdown Menu: Select between different launch sites or view all combined.
  - Purpose: Tailors the displayed data to specific sites or aggregated views.
- **Launch Success Visualization:**
  - Pie Chart: Displays success rates overall or by selected site.
  - Scatter Chart: Examines payload vs. success correlation within selected conditions.
- **Interactive Controls:**
  - Payload Slider: Filters scatter chart data by payload mass range.
  - Dynamic Updates: Visuals adjust in real-time based on user selections.
- https://github.com/generalmarke/ibmcapstoneproject/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- **Data Preparation:**
  - Standardized dataset features to normalize the scale.
  - Split data into training and test sets for model validation.
- **Model Training and Hyperparameter Tuning:**
  - Utilized GridSearchCV for hyperparameter optimization on multiple models: Logistic Regression, SVM, Decision Trees, and KNN.
  - Evaluated models based on cross-validated accuracy scores.
- **Model Evaluation:**
  - Assessed model performance using the test set.
  - Analyzed outcomes using confusion matrices to identify model strengths and weaknesses (e.g., false positives and false negatives).
- **Selection of Best Model:**
  - Compared the accuracy and performance of all models.
  - Identified the highest performing model based on test accuracy and generalization capabilities.
- https://github.com/generalmarke/ibmcapstoneproject/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

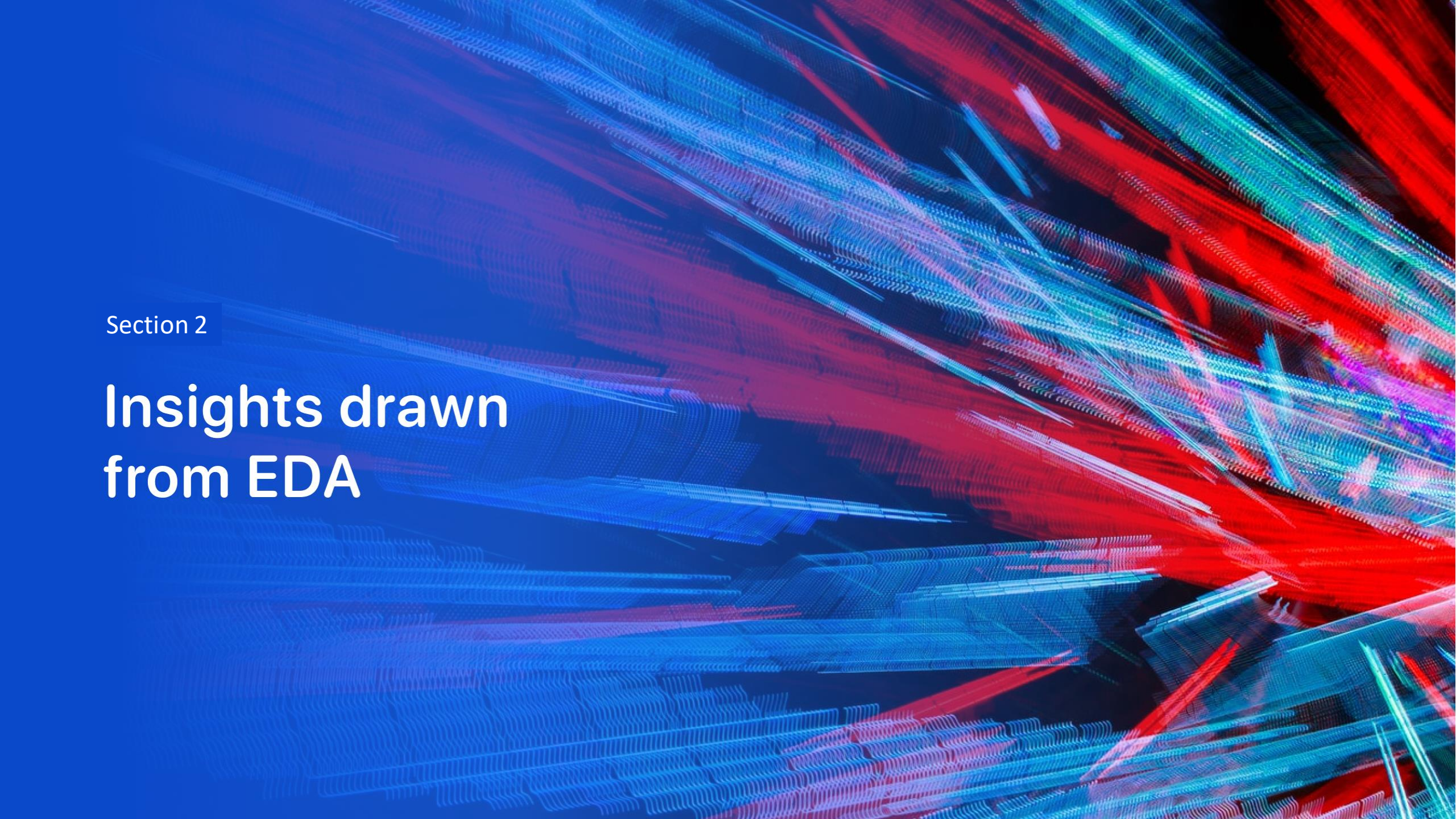


Results

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

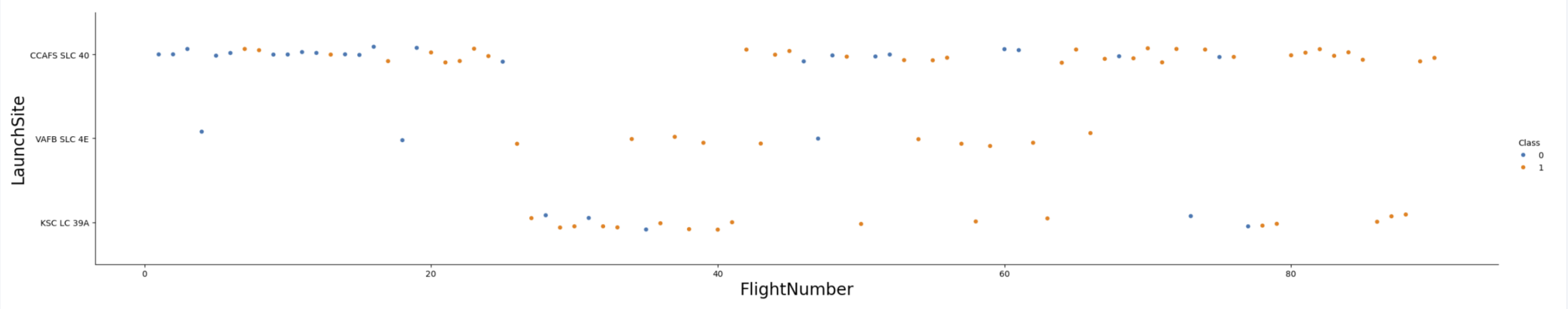
Section 2

Insights drawn from EDA



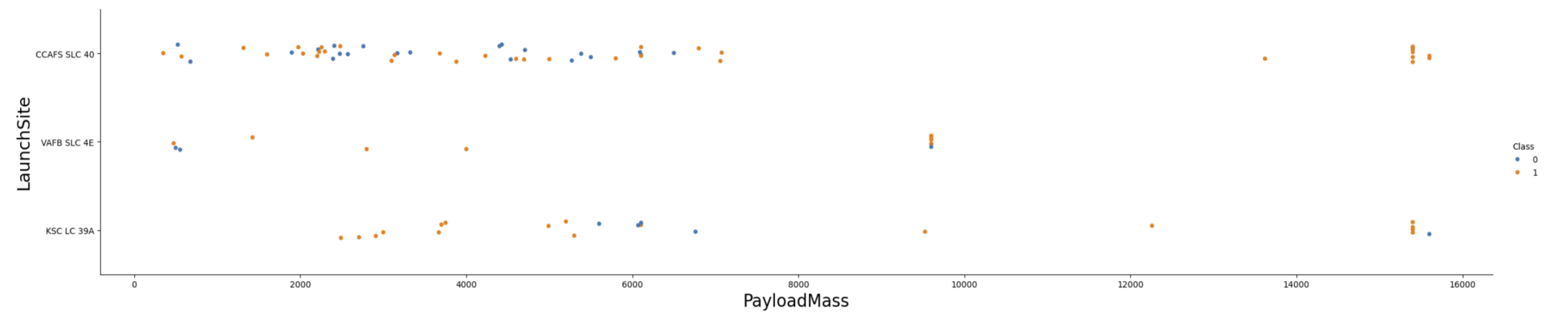
Flight Number vs. Launch Site

- Flight Number in the x-axis. Launch Site on the y-axis. Orange dots indicate successful launches while the blue indicate failed launches.
- As flight number increased, there is an observed increase in success rate in each launch site.



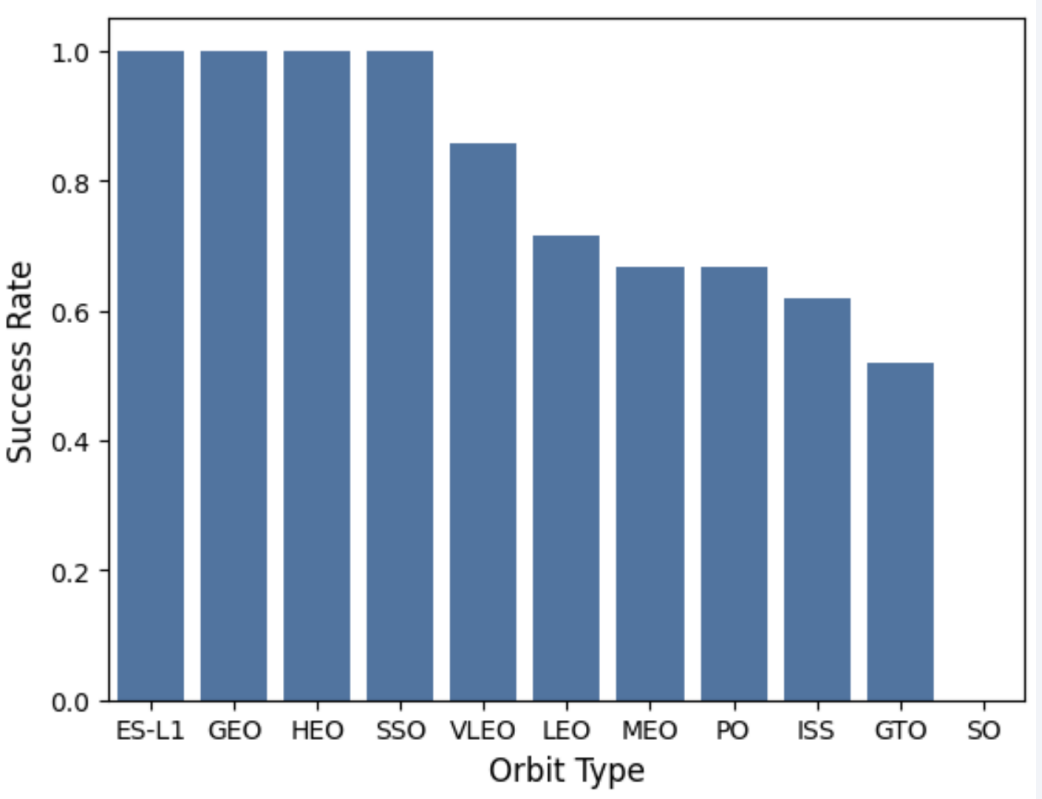
Payload vs. Launch Site

- There isn't a noticeable correlation between Payload & Launch Site.
- Launch Site VAFB SLC 4E had no Payload Mass greater than 10000.
- Almost all launches greater than 9000 have been successful.



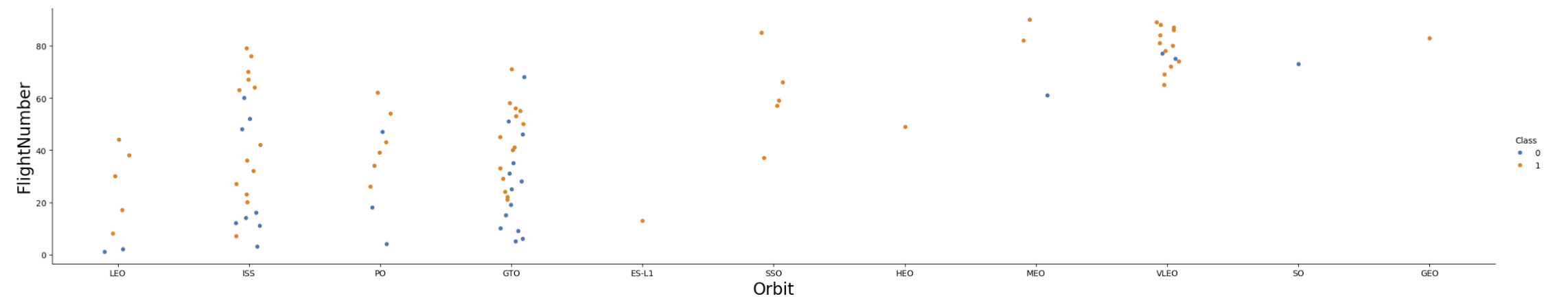
Success Rate vs. Orbit Type

- The x-axis indicates the Orbit Type. The Success Rate is the y-axis.
- Orbit types ES-L1, GEO, HEO & SSO all had a 100% success rate.
- Orbit SO had a 0% success rate.



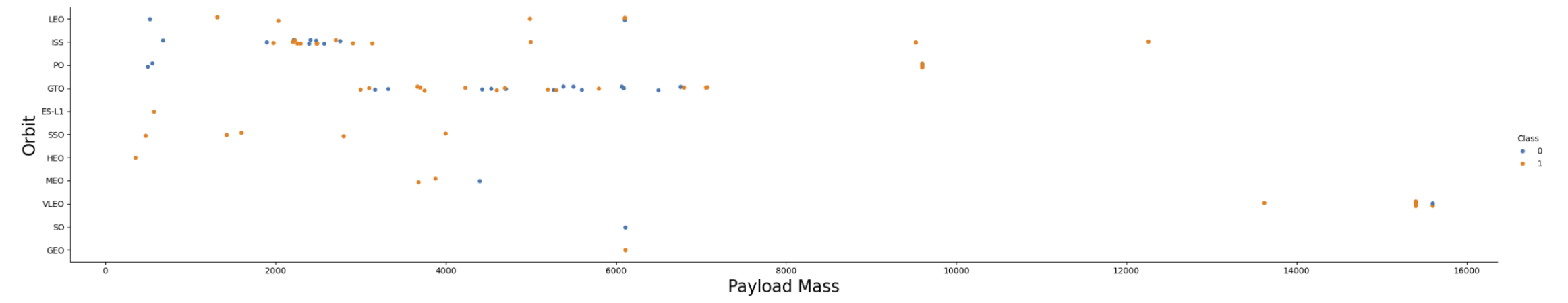
Flight Number vs. Orbit Type

- For Orbit LEO, as Flight Number increased, the success rate increased. There seems to be no other correlations observable.



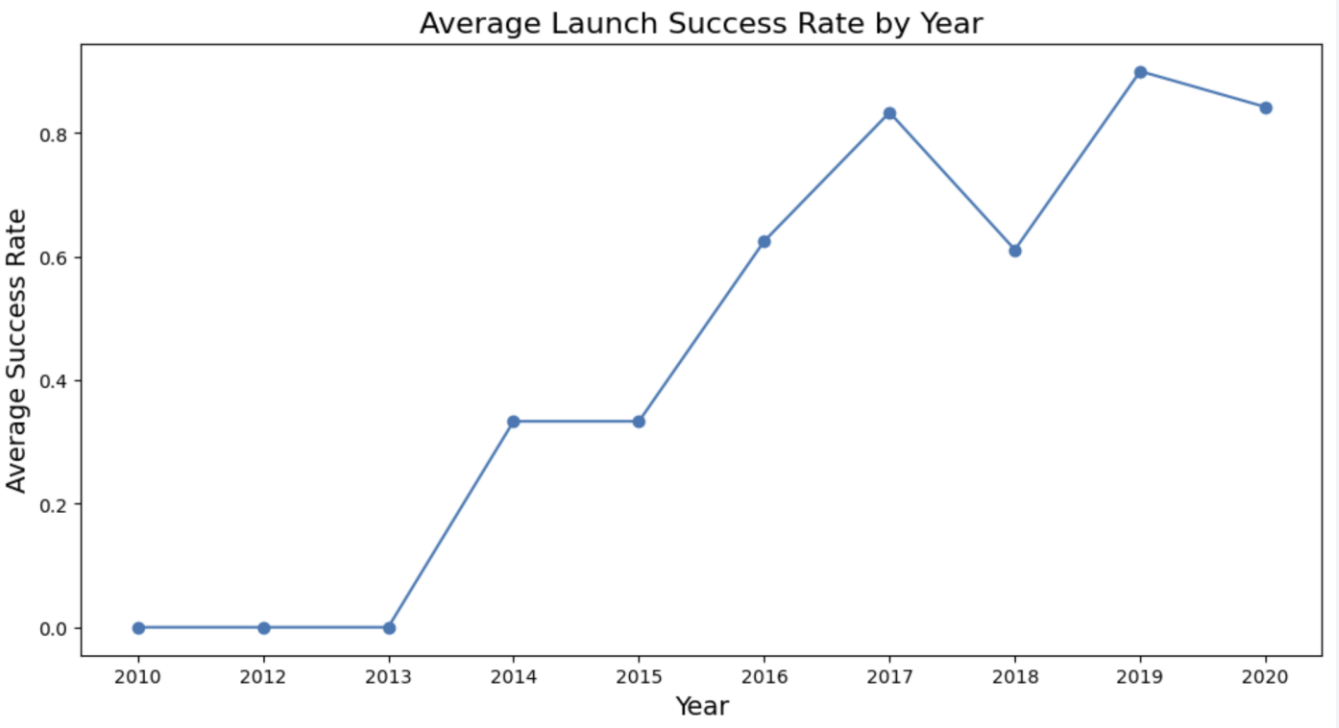
Payload vs. Orbit Type

- There seems to be a correlation between payload mass and success rate for orbits ISS & PO.
- There is slight correlation for LEO. The other orbits don't have any conclusive correlations.



Launch Success Yearly Trend

- The success rate has overall been increasing from 2013 to 2020.
- There was a 0% success rate between 2010-2013.
- The max success rate was in 2019.



All Launch Site Names

- Using the "distinct' query, presented are the unique launch sites in the dataset.

```
In [13]: %sql select distinct Launch_Site from SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

Out[13]:

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

Launch Site Names Begin with 'CCA'

- Presented five records from the data set where the Launch site name begins with "CCA".
- The "like 'CCA%'" allows to selectively query.
- The "limit 5" shows only 5 records.

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

In [15]:
```
%sql select * from SPACEXTABLE where Launch_Site like 'CCA%' limit 5
```

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

Out[15]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcc
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parach
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 (parach
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No atte
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No atte
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No atte

Total Payload Mass

- Calculate the total payload carried by boosters from NASA using the 'SUM' function on the "Payload_Mass__KG_" column.
- The "where Customer = 'NASA (CRS)'" allows to select from where the boosters were launched by NASA.

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

In [21]:
```
%sql select SUM(PAYLOAD_MASS__KG_) as 'Total Payload Mass (KG)' from SPACEXTABLE where Customer = 'NASA (CRS)'
```

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

Out[21]:

Total Payload Mass (KG)
45596

Average Payload Mass by F9 v1.1

- Use the "AVG" function to get the average of the "Payload_Mass__KG_" for the Booster_Version that is like "F9 v1.1"

Query: %sql select AVG(PAYLOAD_MASS__KG_) as 'Average Payload Carried by F9 v1.1 (KG)' from SPACEXTABLE where Booster_Version like 'F9 v1.1%'

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

In [22]:
```
%sql select AVG(PAYLOAD_MASS__KG_) as 'Average Payload Carried by F9 v1.1 (KG)' from SPACEXTABLE where Booster_Version like
```

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

Out[22]:

Average Payload Carried by F9 v1.1 (KG)
2534.6666666666665

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad by using the "MIN" function and the "Where" function to query where "Landing_Outcome" will be "Success (ground pad)".

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

Hint:Use min function

In [43]:
```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)'
```

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

Out[43]:

MIN(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Find the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 using the "where" function to specify range and the "like 'Success (drone ship)' to get the successful drone ship landings

```
Query: %sql select Booster_Version,PAYLOAD_MASS__KG_,Landing_Outcome from SPACEXTABLE where PAYLOAD_MASS__KG_ > 4000 and
PAYLOAD_MASS__KG_ < 6000 and Landing_Outcome like 'Success (drone ship)' order by PAYLOAD_MASS__KG_
```

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

In [39]:

```
%sql select Booster_Version,PAYLOAD_MASS__KG_,Landing_Outcome from SPACEXTABLE where PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MA
and Landing_Outcome like 'Success (drone ship)' order by PAYLOAD_MASS__KG_
```

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

Out[39]:

Booster_Version	PAYLOAD_MASS__KG_	Landing_Outcome
F9 FT B1026	4600	Success (drone ship)
F9 FT B1022	4696	Success (drone ship)
F9 FT B1031.2	5200	Success (drone ship)
F9 FT B1021.2	5300	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes using the "Count" function on the "Mission_Outcome" column.
- 100 overall mission successes and one failure.

List the total number of successful and failure mission outcomes

In [15]:
```
%sql select Mission_Outcome, count(*) as "Count" from SPACEXTABLE group by Mission_Outcome;
```

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

Out[15]:

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

Boosters Carried Maximum Payload

- Query of the names of the booster which have carried the maximum payload mass using a "MAX" statement to find the maximum payload in "Payload_Mass_KG_". Then using the "Where" statement, find any payload mass that match that maximum.

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

```
In [16]: %sql select Booster_Version,PAYLOAD_MASS__KG_ from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select MAX(PAYLOAD_MASS__KG_) from
```

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

Out[16]:

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

Query: `%sql select Booster_Version,PAYLOAD_MASS__KG_ from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select MAX(PAYLOAD_MASS__KG_) from SPACEXTABLE)`

2015 Launch Records

- Query for a list of the failed "landing_outcomes" in drone ship, their booster versions, and launch site names for in year 2015. Done using "Where" statement to specify "Failure (drone ship)" and the "Substr" function to derive the month from the date.

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 [17]:
```
%sql select substr(Date, 6, 2) as Month, Date, Booster_Version, Launch_Site, Landing_Outcome \
from SPACEXTABLE where Landing_Outcome like 'Failure (drone ship)' and substr(Date, 1, 4) like '2015';
```

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

Out[17]:

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

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

- Query for 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 using the "where" statement to specify date range and the "count", "group" and "order" function to get the count of the landing outcomes.

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 [18]:
```
%sql select Landing_Outcome, count(Landing_Outcome) as Outcome_Count from SPACEXTABLE \
where Date >= '2010-06-04' and Date <= '2017-03-20' group by Landing_Outcome order by Outcome_Count desc;
```

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

Out[18]:

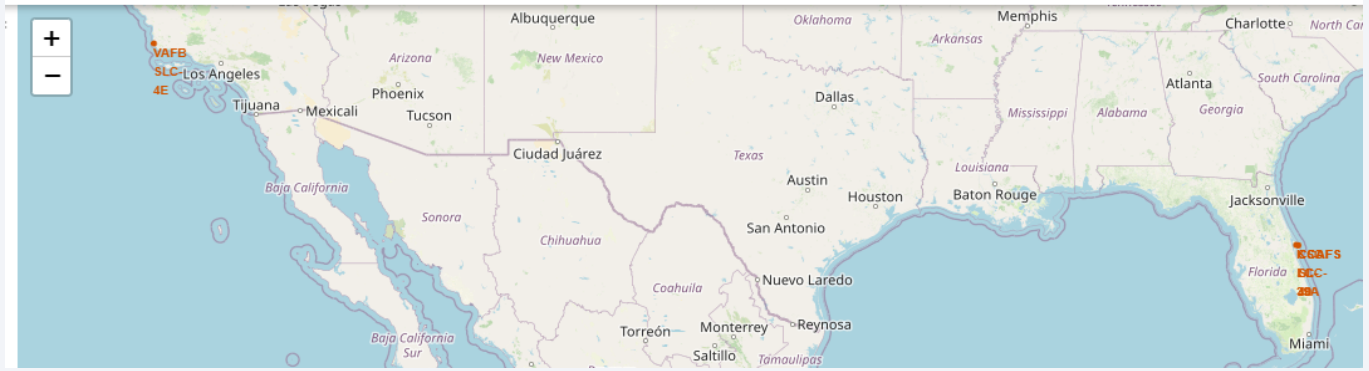
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1


Section 3
Launch Sites
Proximities Analysis

Folium Map of All Launch Sites

- This folium map is an interactive map that has all the launch sites marked.
- The marker on the map contains the launch site name and then a circle surrounding it.

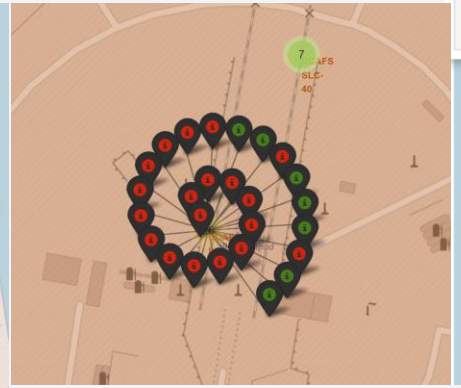
```
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as a popup label

for index, row in launch_sites_df.iterrows():
    # Extract the latitude and longitude
    site_coords = [row['Lat'], row['Long']]
    site_name = row['Launch Site']

    site_circle = folium.Circle(site_coords, radius=1000, color='#d35400', fill=True).add_child(folium.Popup(site_name))
    site_map.add_child(site_circle)

    site_marker = folium.Marker(
        site_coords,
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#d35400;"><b>{}</b></div>'.format(site_name),
            )
        )
    site_map.add_child(site_marker)

site_map
```

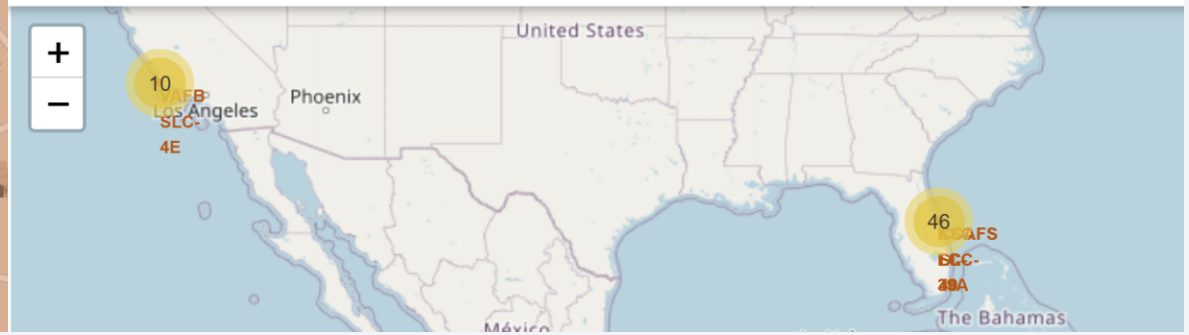


Folium Map of Successful/Failed Launches

- This map shows all the successful/failed launches marked on the map for each site.

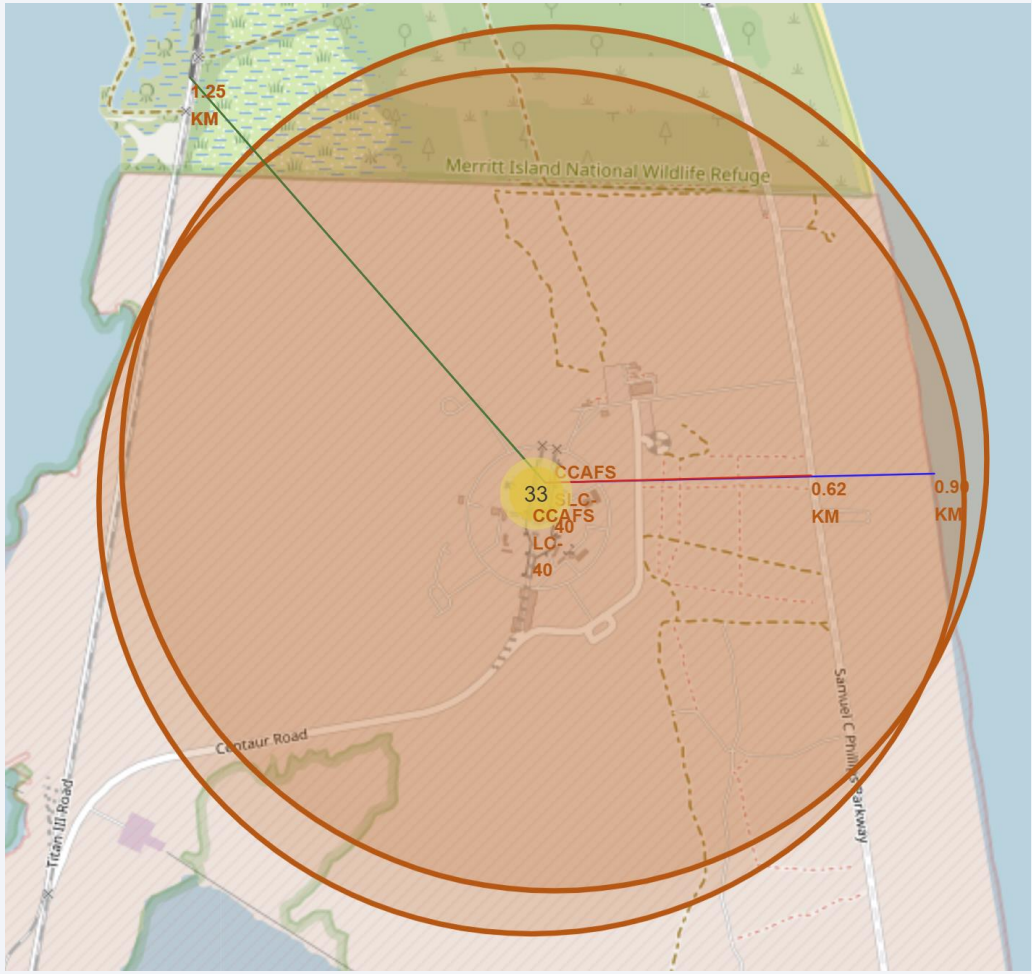
```
# Add marker_cluster to current site_map
site_map.add_child(marker_cluster)

# for each row in spacex_df data frame
# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch was successed or failed,
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
for index, record in spacex_df.iterrows():
    # TODO: Create and add a Marker cluster to the site map
    # marker = folium.Marker(...)
    marker = folium.Marker(
        [record['Lat'], record['Long']],
        icon=folium.Icon(color='black', icon_color=record['marker_color']),
        popup=record['Launch Site']
    )
    marker_cluster.add_child(marker)

site_map
```

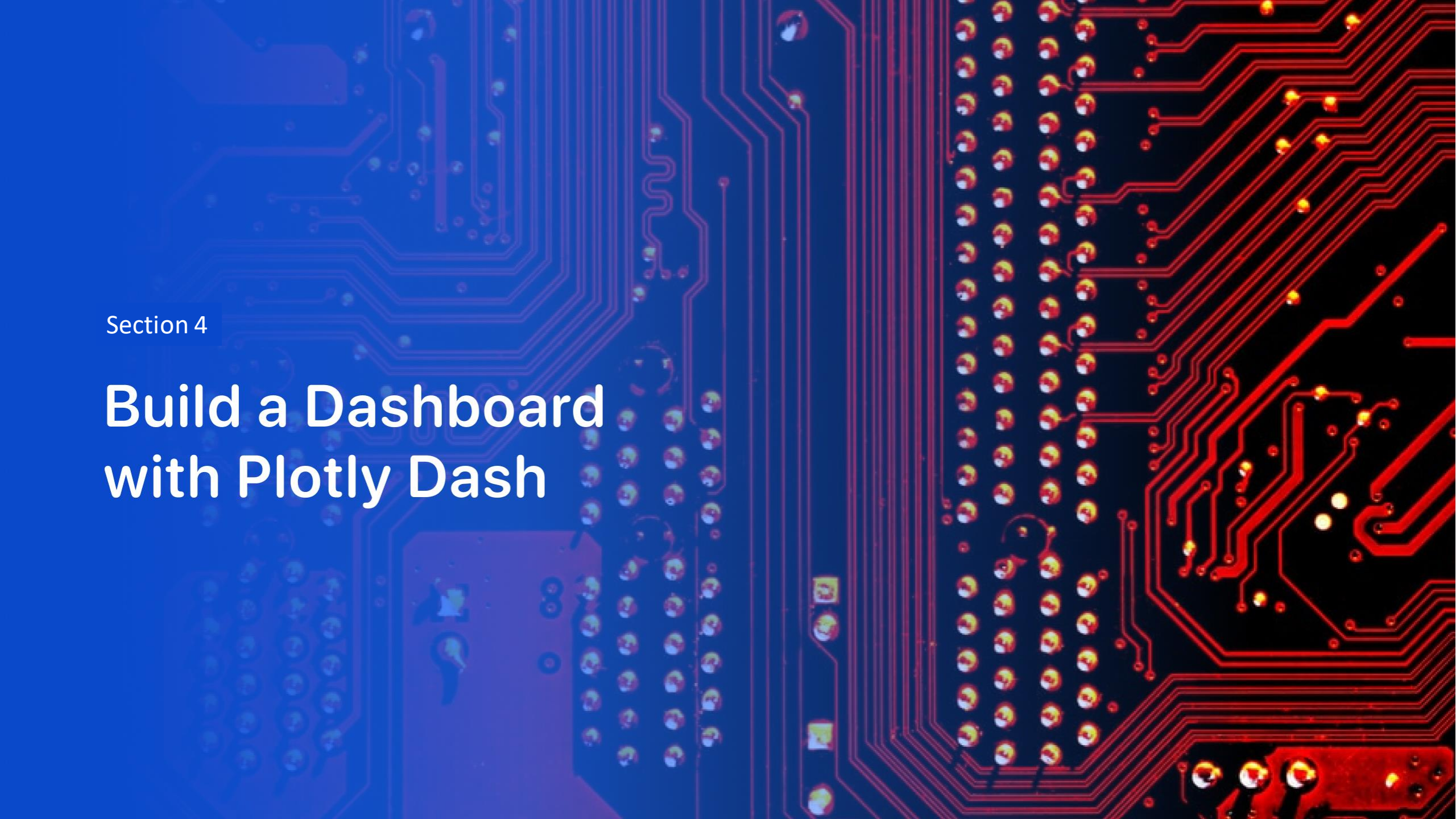




Folium Map with Launch Sites and Distance Markers

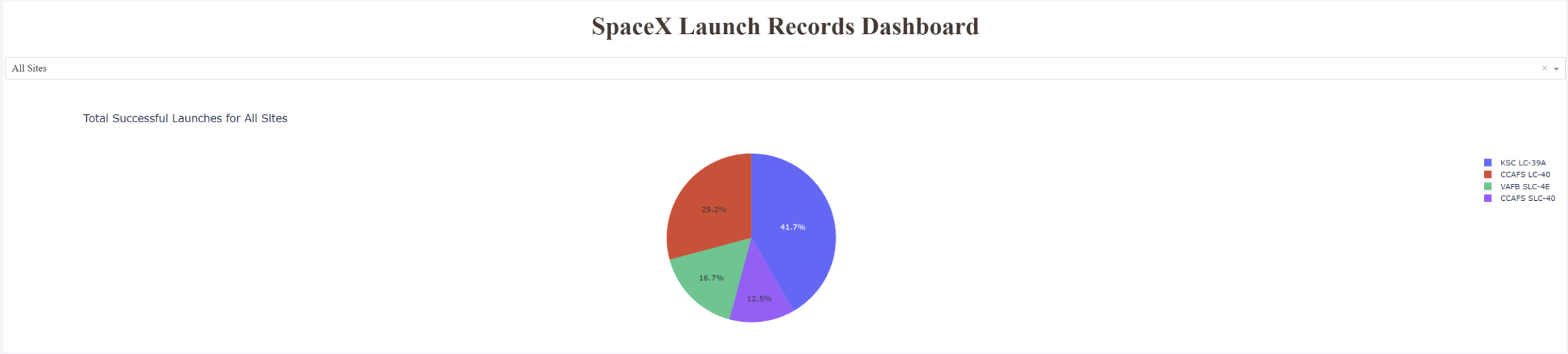
- Show the launch sites (CCAFS SLC-40) with the distance to nearby places of interest.
- Red Line: Highway (0.62KM)
- Blue Line: Coastline (0.90KM)
- Green Line: Railway (1.25KM)




Section 4
Build a Dashboard with Plotly Dash

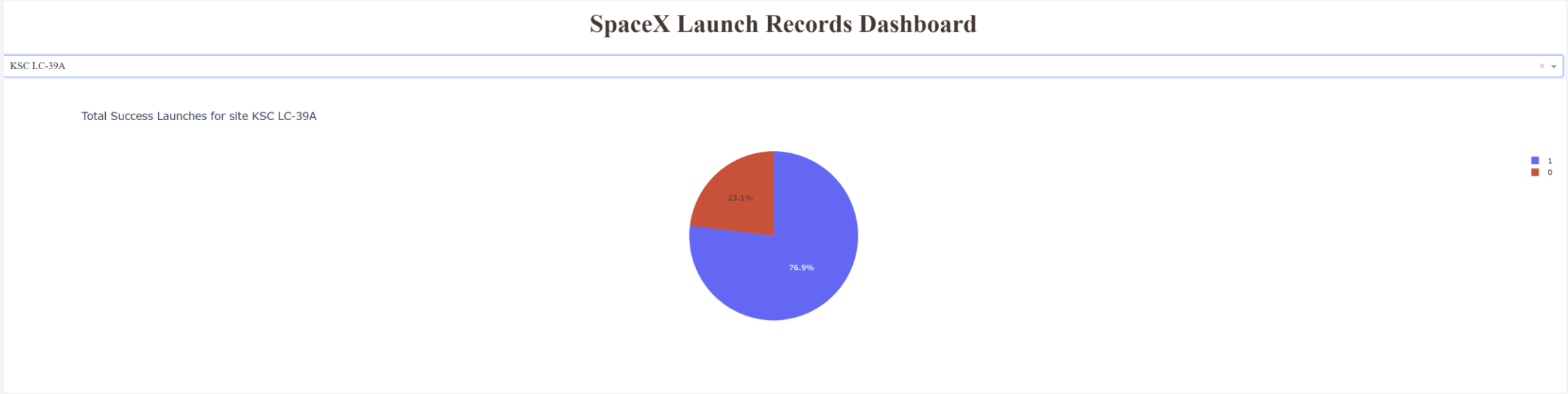
Launch Records Dashboard – Successful Launches

- Pie chart of launch success count for all launch sites
- KSC LC-39A had the greatest percentage of successful launches while CCAFS SLC-40 had the least.



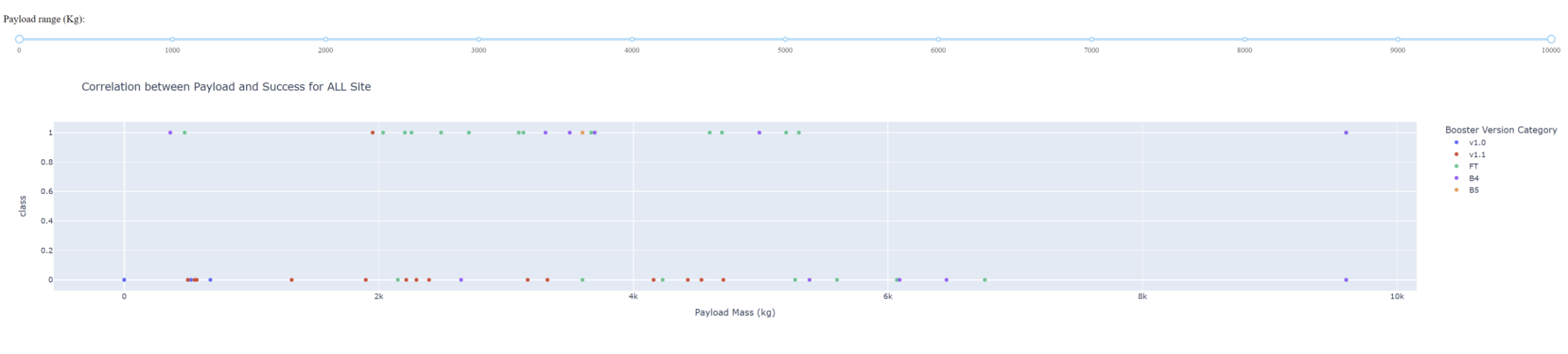
Launch Records Dashboard – Successful Launches

- Pie chart for the launch site with highest launch success ratio – KSC LC-39A



Launch Records Dashboard

- Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider
- The most successes are in the 2K to 6K range but have a similar number of failures. Booster version FT had more successes than the other boosters.



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier was the best model with the highest accuracy when compared to logistic regression, SVM, and k-nearest neighbor.

```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

```
tree_cv = GridSearchCV(tree, parameters, cv=10)
tree_cv.fit(X_train, Y_train)
```

```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

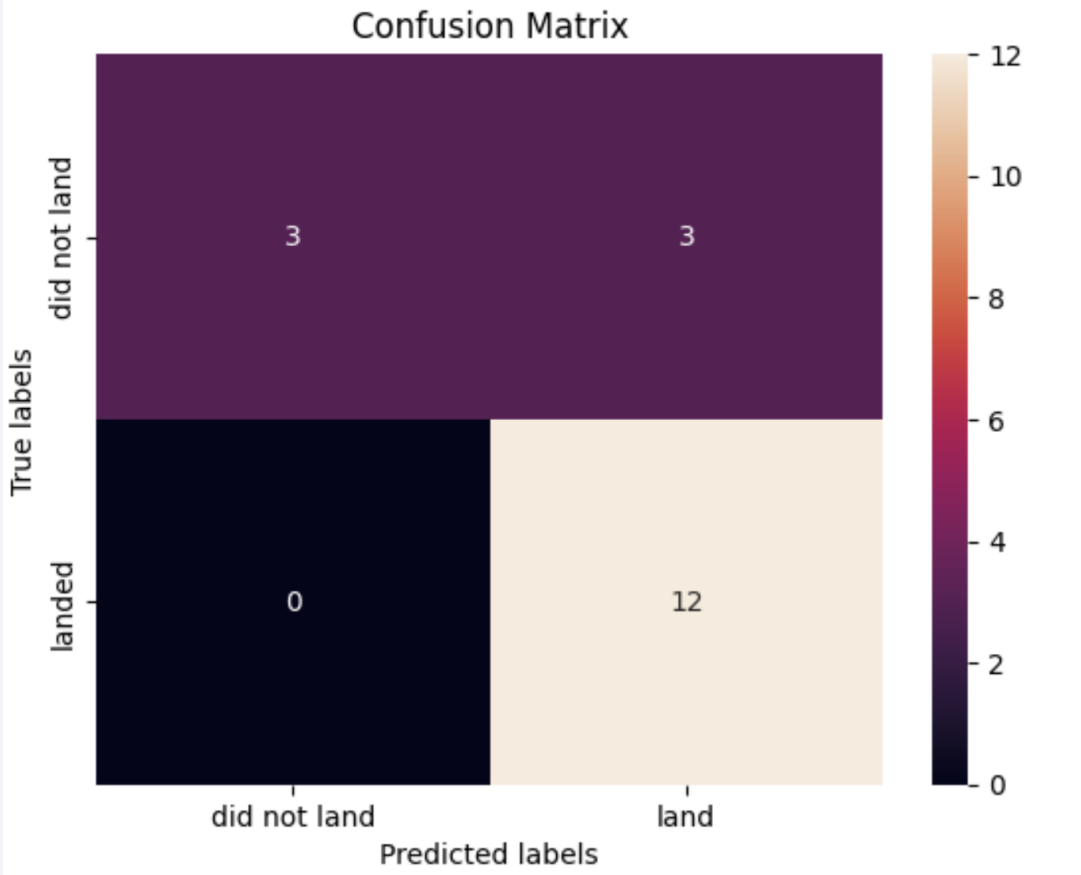
```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf':
1, 'min_samples_split': 2, 'splitter': 'random'}
accuracy : 0.8875
```

Confusion Matrix

- The confusion matrix shows how the model performed for the predicting the test data set. It shows what was correctly predicted and that what was false positives or negatives.

We can plot the confusion matrix

```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- There was a positive correlation between flight number and the success rate of the launches.
- The launch success rate had overall increased between 2013 and 2020.
- Orbit types ES-L1, GEO, HEO & SSO all had a 100% success rate.
- KSC LC-39A had the greatest percentage of successful launches while CCAFS SLC-40 had the least.
- The decision tree classifier had the highest accuracy out of the classification models.

Appendix

- All Python code, SQL queries, charts, Notebook outputs, and data sets are on this repository: https://github.com/generalmarke/ibmcapstoneproject/tree/main

Thank you!