IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Amber Maschino
17 December 2024

Outline

- EXECUTIVE SUMMARY
- INTRODUCTION
- METHODOLOGY
- RESULTS
- CONCLUSION
- APPENDIX

Executive Summary

Summary of methodologies

- Data Collection through APIs and Web Scraping
- Data Wrangling
- Exploratory Data Analysis with SQL and Data Visualization
- Interactive Visual Analytics with Folium and Plotly Dash
- Machine Learning Prediction

Results

- Recent launch’s first stage is more likely to return successfully.
- Even with more massive payloads, the first stage often returns successfully.
- VAFB-SLC launch site has no rockets launched for heavy payload mass (greater than 10,000 kg).
- Orbit Types: Success rates vary across different orbit types.
- Polar, LEO, and ISS orbits have higher successful landing rates for heavy payloads.
- The success rate has been increasing since 2013 up to 2020.

Introduction

Background:

As technology grows, we have moved closer to affordable space travel. Companies are looking to keep costs down to allow for them to increase their profits. Blue Origin is manufacturing reusable orbital rockets. SpaceX is keeping their manufacturing costs down by reusing the first stage of their rockets when able. Currently, they spend only $65 million on a Falcon9 launch where other companies are spending up to $165 million.

Questions to Answer:

- When does SpaceX reuse their first stage of the rocket?
- What criteria must be met to reuse the stage one?
- Can we predict when SpaceX will reuse their first stage?

Section 1

Methodology

Methodology

- Executive Summary
- Data collection methodology:
  - APIs
  - Web Scraping
- Perform data wrangling
  - Removing nulls by replacing them with mean values
  - Remove irrelevant data
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

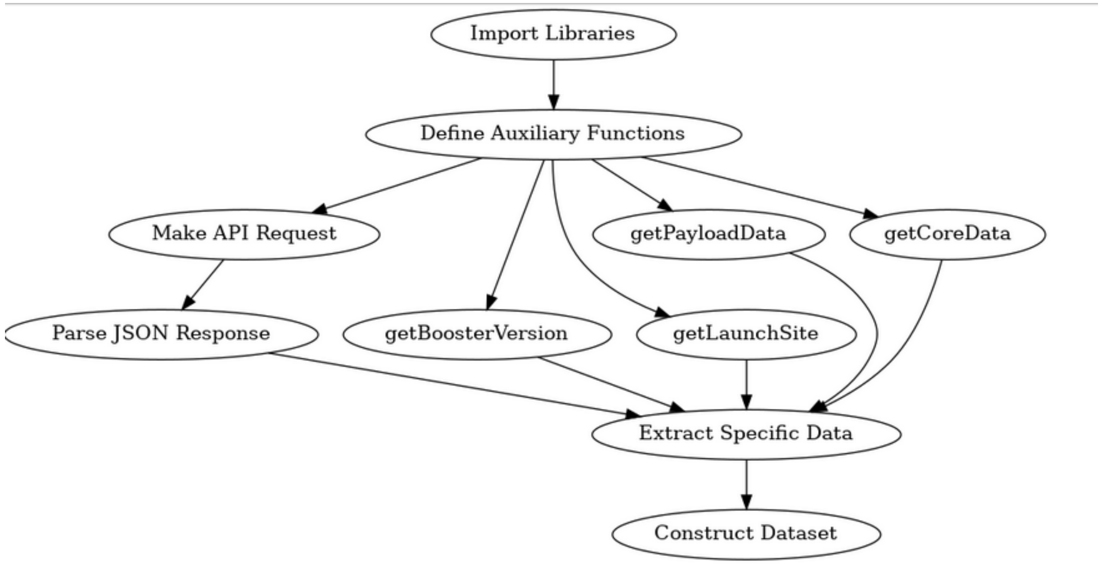
Data was collected using various methods:

- Using get requests to the SpaceX API
  - Used pandas to create a dataframe and then normalized the data with .json_normalize()
- By web scraping Falcon 9 data from the Wikipedia for launch records using BeautifulSoup
  - Parsed the table and converted it into a pandas dataframe

Data Collection – SpaceX API

Key Phrases:

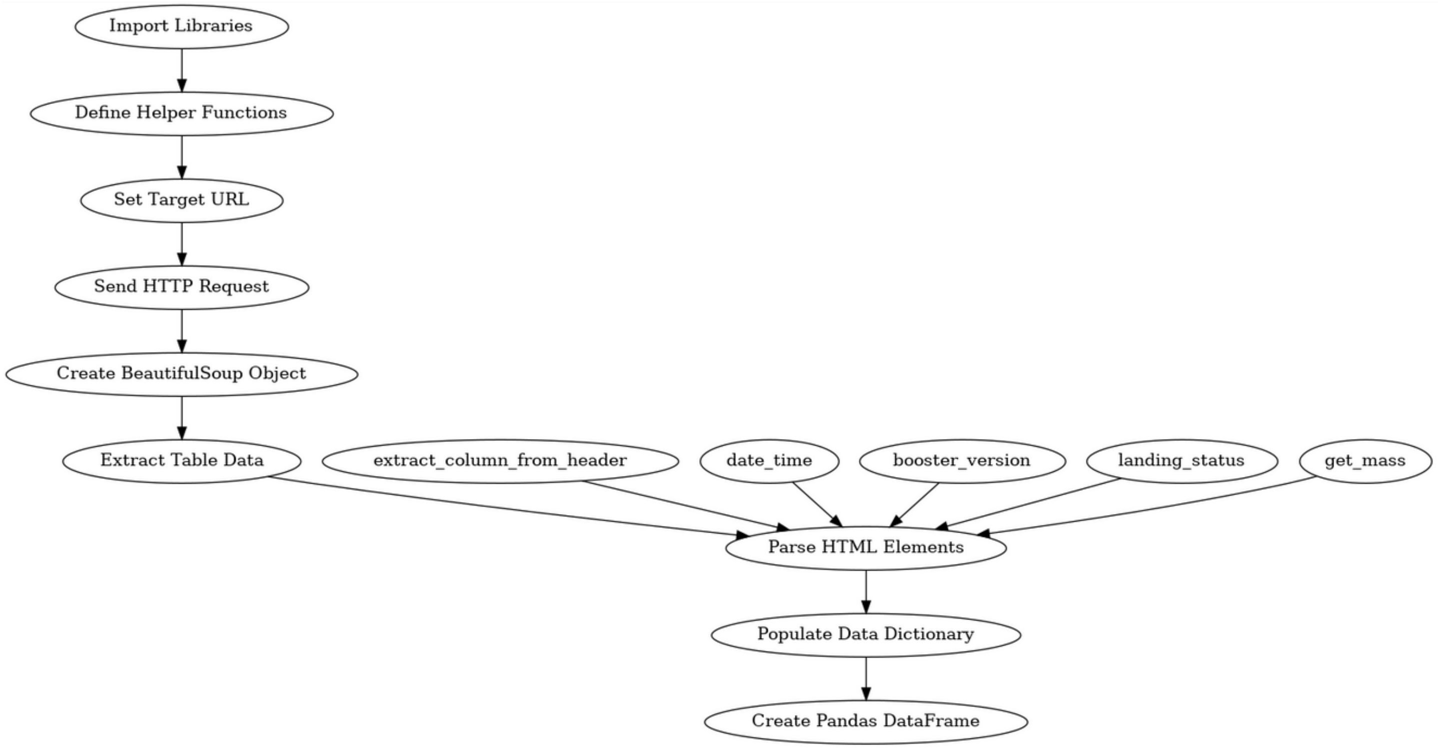
- Import Libraries
- Define Auxiliary Functions
- Make API Requests
- Parse JSON Response
- Extract Specific Data
- Construct Dataset



GitHub URL of the SpaceX API calls notebook:

https://github.com/AM-byte07/IBM_DS_Capstone/blob/main/ibm_ds_spacex-data-collection_lab.pdf

Data Collection - Scraping



Key Phrases:

- Import Required Libraries
- Define Helper Functions
- Set Target URL
- Send HTTP Request
- Create BeautifulSoup Object
- Extract Table Data
- Parse HTML Elements
- Populate Data Dictionary
- Create Pandas DataFrame

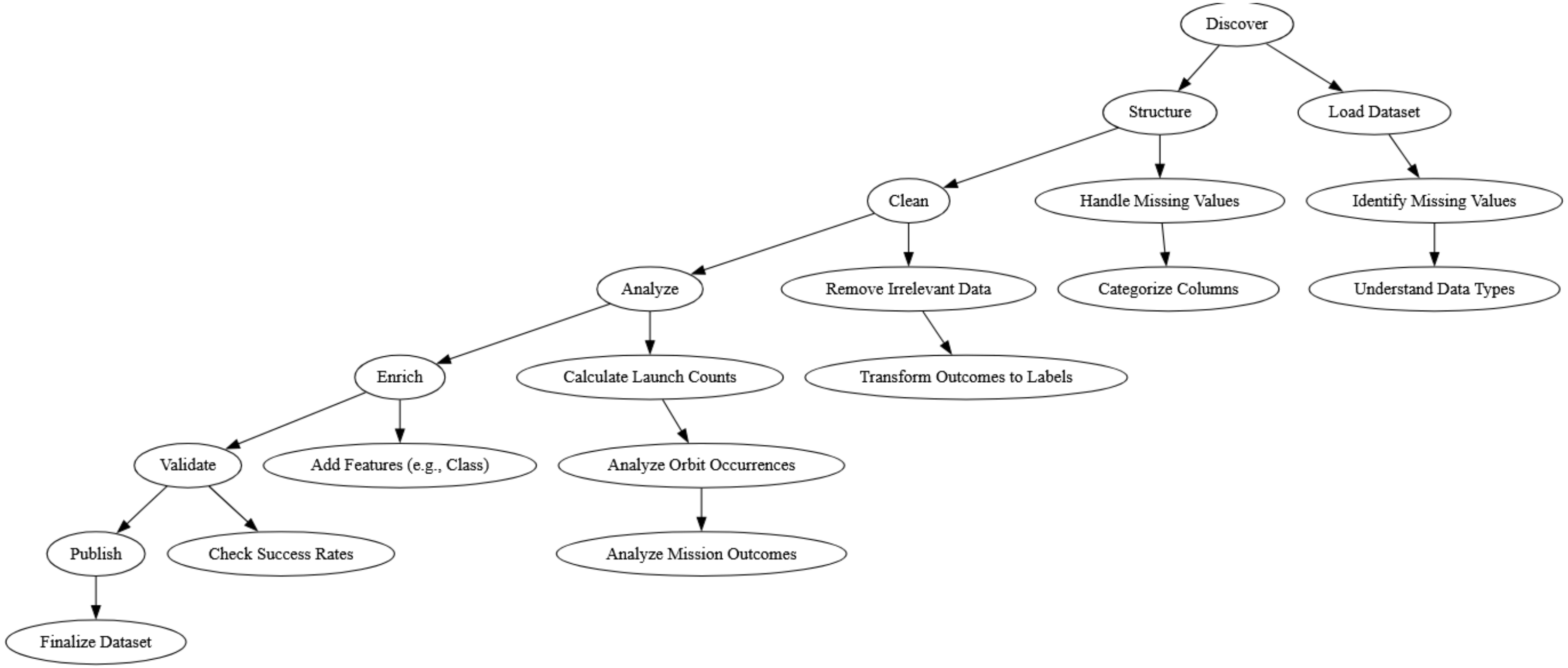
GitHub URL of the web scraping notebook:

- https://github.com/AM-byte07/IBM_DS_Capstone/blob/main/ibm_ds_webscraping_lab.pdf

Data Wrangling

Key Phrases:

- Discover
- Structure
- Clean
- Analyze
- Enrich
- Validate
- Publish



GitHub URL of data wrangling related notebooks:

https://github.com/AM-byte07/IBM_DS_Capstone/blob/main/ibm_ds_data%20wrangling_lab.pdf

EDA with Data Visualization

Scatterplot Flight Number Vs Launch Site: To visualize the distribution of launches across different sites over time.

- **Observation**: The plot reveals which launch sites are more frequently used and their success rates. Notably, there is a concentration of launches from specific sites, indicating operational preferences.

Scatterplot Launch Site vs Payload Mass: To explore the relationship between launch sites and the payload mass carried.

- **Observation**: The VAFB-SLC launch site shows no launches for heavy payloads (greater than 10,000 kg), indicating limitations in handling larger missions.

Bar Graph of Orbit vs Success Rate: To compare success rates across various orbit types.

- **Observation**: Higher success rates are observed for Polar, LEO, and ISS orbits with heavier payloads, while GTO orbits show a mix of successful and unsuccessful landings, making it difficult to distinguish outcomes.

Scatter Point Chart of Flight Number vs Orbit: To examine how flight numbers correlate with different orbit types.

- **Observation**: In LEO orbits, success appears to correlate positively with flight numbers; however, GTO orbits do not exhibit a clear relationship between flight numbers and success rates.

EDA with Data Visualization Cont.

Scatter Point Chart of Payload Mass vs Orbit: To visualize how payload mass varies with different orbit types.

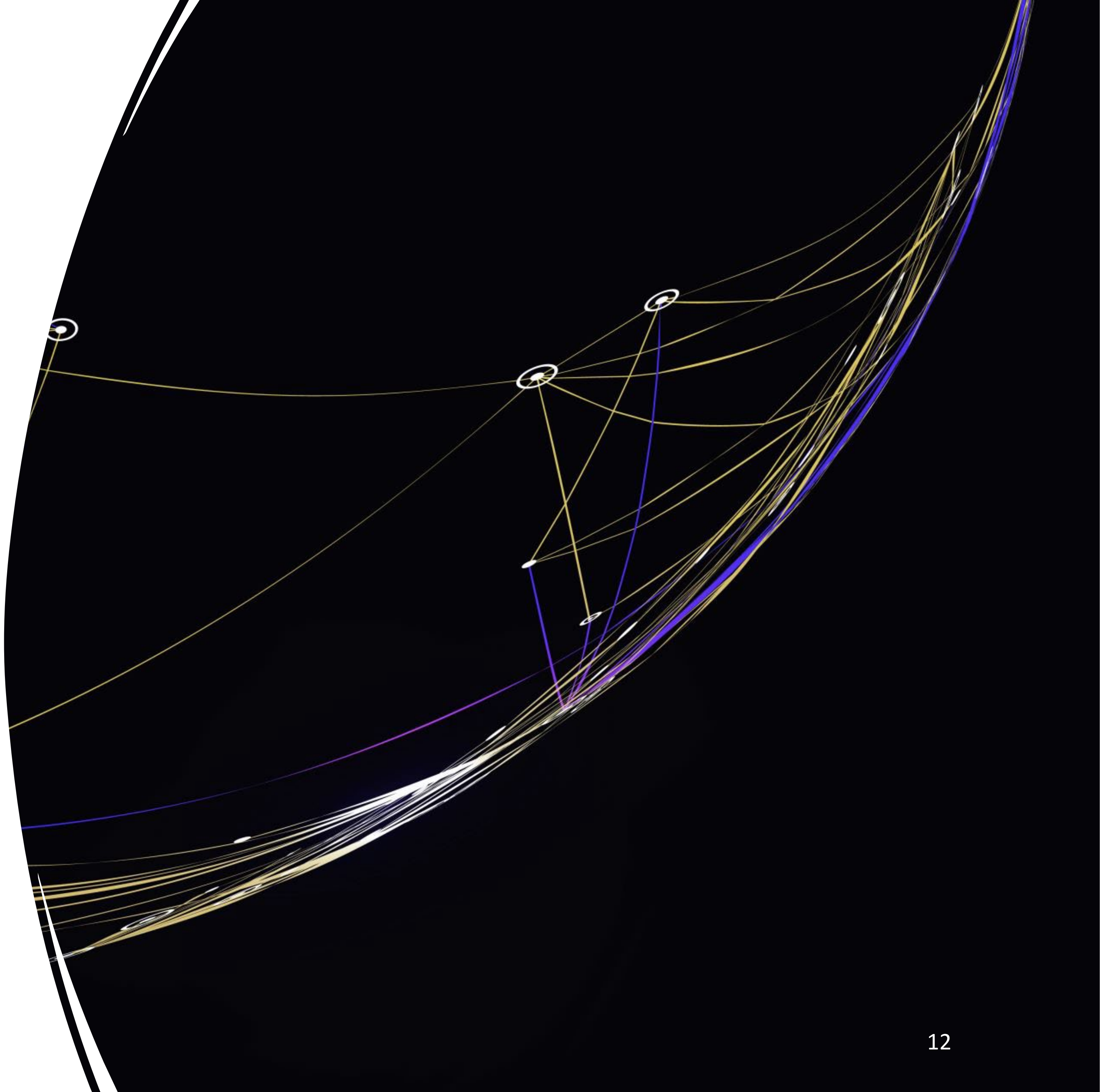
- **Observation**: The scatter plot highlights trends in payload distribution across orbit types, emphasizing that heavier payloads are more successfully launched in specific orbits.

Line Chart of the Year vs Success Rate: To illustrate trends in average launch success rates over the years.

- **Observation**: The trend shows a consistent increase in success rates from 2013 to 2020, indicating improvements in technology and operational efficiency over time.

GitHub URL of the EDA with data visualization notebook:

https://github.com/AM-byte07/IBM_DS_Capstone/blob/main/ibm_ds_eda.pdf



EDA with SQL

SQL Queries with Summary:

Unique Launch Sites:

- **Summary**: This query retrieves distinct launch sites where SpaceX missions have occurred, highlighting locations such as CCAFS LC-40, VAFB SLC-4E, and KSC LC-39A. Understanding these sites is crucial for analyzing operational patterns and geographical preferences in SpaceX launches.

Launch Sites Starting with 'CCA’

- **Summary**: The query filters launch sites that start with "CCA," showing repeated entries for CCAFS LC-40. This indicates that this site has been a significant launch point for SpaceX, which may reflect its operational capacity and frequency of use.

Total Payload Mass for NASA (CRS)

- **Summary**: This query calculates the cumulative payload mass for missions involving NASA's Commercial Resupply Services (CRS), totaling 45,596 kg. This information is essential for understanding NASA's reliance on SpaceX for cargo deliveries to the International Space Station.

EDA with SQL Cont.

SQL Queries with Summary:

Average Payload Mass for Booster Version F9 v1.1

- **Summary**: The average payload mass for the F9 v1.1 booster is approximately 2,534.67 kg. This metric helps assess the performance and capacity of specific booster versions in carrying payloads.

First Successful Landing Date on Ground Pad

- **Summary**: The query identifies multiple dates when successful landings occurred on ground pads, starting from December 22, 2015. Tracking these milestones is vital for evaluating SpaceX's progress in landing technology.

Boosters with Successful Drone Ship Landings

- **Summary**: This query targets specific booster versions that achieved successful landings on drone ships while carrying moderate payloads. Identifying these boosters can provide insights into operational success rates under varying payload conditions.

Total Successful and Failed Mission Outcomes

- **Summary**: The results show a breakdown of landing outcomes, with notable successes (38 total) and various failure types (e.g., controlled ocean landings). This data is critical for assessing overall mission reliability and performance.

EDA with SQL Cont.

SQL Queries with Summary:

Booster Versions with Maximum Payload Mass

- **Summary**: The query identifies multiple booster versions capable of carrying maximum payloads, indicating their significance in heavy-lift missions. Understanding which boosters achieve these feats can inform future mission planning.

Monthly Failure Outcomes in Drone Ship Landings (2015)

- **Summary**: The results highlight specific months where failures occurred during drone ship landings, providing insights into operational challenges faced during those periods.

Ranking Landing Outcomes Between Specific Dates

- **Summary**: This query ranks various landing outcomes by frequency within a specified date range, revealing trends in mission success and failure rates over time.

GitHub URL of the EDA with SQL notebook:

https://github.com/AM-byte07/IBM_DS_Capstone/blob/main/ibm_ds-eda-sql-lab.pdf

Build an Interactive Map with Folium

Markers for Launch Sites (Object: folium.Marker)

- **Purpose**: Each launch site was marked with a distinct colored marker indicating its location on the map. The markers were labeled with the name of the launch site using DivIcon to provide clear identification.
  Reason: This allows users to easily locate and identify each launch site, facilitating an understanding of SpaceX's operational geography.

Circles Around Launch Sites (Object: folium.Circle)

- **Purpose**: A colored circle (with a radius of 1000 meters) was drawn around each launch site.
  Reason: Circles visually represent the area surrounding each launch site, helping to illustrate proximity and potential operational zones. Different colors were used for each site to enhance visual differentiation.

Markers for Launch Outcomes (Object: folium.Marker)

- **Purpose**: Markers were added for each launch record, colored green for successful launches and red for failed launches.
  Reason: This visual distinction allows users to quickly assess the success rates at various launch sites, highlighting patterns in performance.

Marker Clusters (Object: MarkerCluster)

- **Purpose**: A marker cluster was created to group markers representing multiple launches at the same location.
  Reason: This helps manage map clutter when multiple launches occur at the same site, allowing for a cleaner and more organized visual representation.

Build an Interactive Map with Folium Cont.

Mouse Position Plugin (Object: MousePosition)

- **Purpose**: This plugin displays latitude and longitude coordinates of the mouse pointer on the map.
  Reason: It aids users in identifying specific geographical points of interest (e.g., railways, highways) as they explore the map.

Distance Calculation Markers (Objects: folium.Marker & DivIcon)

- **Purpose**: Markers were added for key proximity points (like coastlines or cities), along with distance labels indicating how far these points are from the launch sites.
  Reason: This provides valuable context regarding how close launch sites are to important infrastructure, which can impact operational logistics.

Polylines Between Points (Object: folium.PolyLine)

- **Purpose**: Lines were drawn between launch sites and their proximities (e.g., coastline, city) to visually represent distances.
  Reason: Polylines help illustrate direct relationships between locations, making it easier to understand spatial dynamics and distances involved.

GitHub URL of the interactive map with Folium map:

https://github.com/AM-byte07/IBM_DS_Capstone/blob/main/ibm_ds_interactive_map.pdf

Build a Dashboard with Plotly Dash

Dropdown Menu (Component: dcc.Dropdown)

- **Purpose**: Allows users to select a specific launch site or view data for all sites
  Interaction: Updates both the pie chart and scatter plot based on selection

Pie Chart (Component: dcc.Graph)

- **Purpose**: Visualizes the success rate of launches
  Interaction: Updates based on the selected launch site from the dropdown

Range Slider (Component: dcc.RangeSlider)

- **Purpose**: Enables selection of payload mass range (0-10,000 kg)
  Interaction: Filters data for the scatter plot based on selected range

Scatter Plot (Component: dcc.Graph)

- **Purpose**: Shows correlation between payload mass and launch success
  Interaction: Updates based on both dropdown selection and range slider input

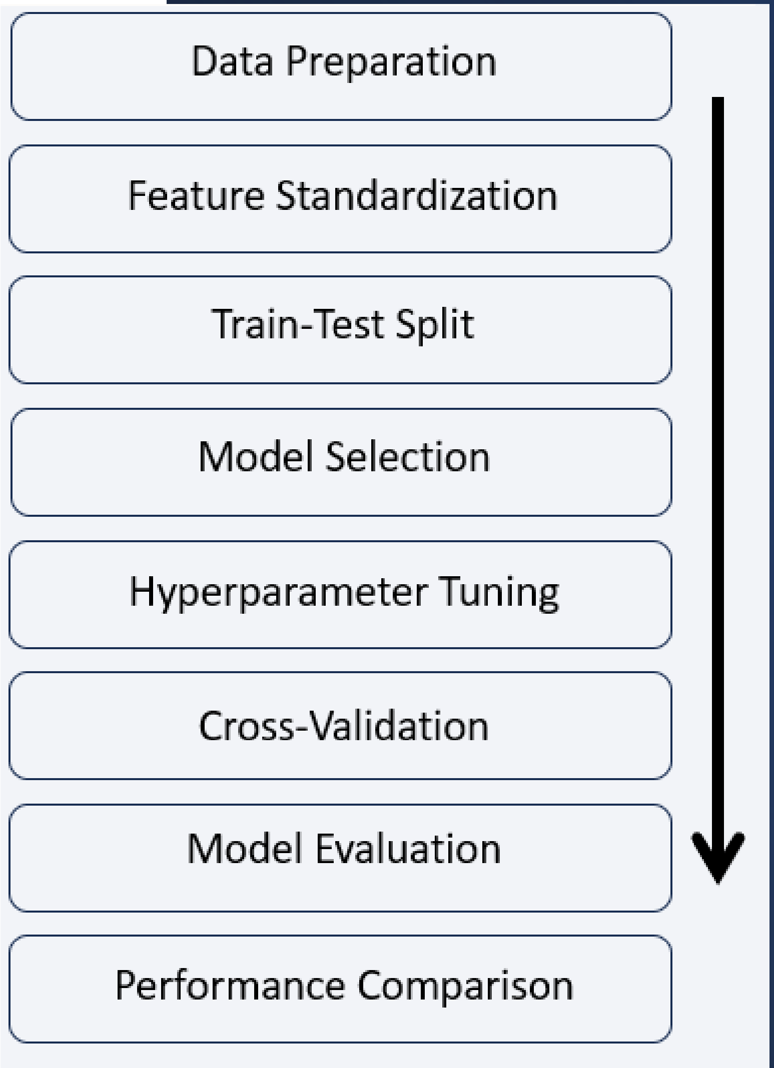
GitHub URL of the Plotly Dash lab:

https://github.com/AM-byte07/IBM_DS_Capstone/blob/main/ibm_ds_plotly_dash.pdf

Predictive Analysis (Classification)

<u>Key Phrases:</u>

- Data Preparation
- Feature Standardization
- Train-Test Split
- Model Selection
- Hyperparameter Tuning
- Cross-Validation
- Model Evaluation
- Performance Comparison



GitHub URL of the predictive analysis lab:
https://github.com/AM-byte07/IBM_DS_Capstone/blob/main/ibm_ds_predictive_analysis_lab.pdf

Predictive Analysis (Classification) Cont.

Summary of model development process:

- **<u>Data Preparation:</u>** Loaded and preprocessed the SpaceX launch data.
- **<u>Feature Standardization:</u>** Applied StandardScaler to normalize features.
- **<u>Train-Test Split:</u>** Divided data into 80% training and 20% testing sets.
- **<u>Model Selection:</u>** Chose four classification algorithms - Logistic Regression, Support Vector Machine, Decision Tree, and K-Nearest Neighbors.
- **<u>Hyperparameter Tuning</u>**: Used GridSearchCV to find optimal hyperparameters for each model.
- **<u>Cross-Validation:</u>** Employed 10-fold cross-validation during hyperparameter tuning.
- **<u>Model Evaluation:</u>** Assessed each model's performance using accuracy scores and confusion matrices.
- **<u>Performance Comparison:</u>**
  - Logistic Regression: 83.33% accuracy
    Support Vector Machine: 83.33% accuracy
    Decision Tree: 91.79% accuracy (best performance)
    K-Nearest Neighbors: 83.33% accuracy

Results

EXPLORATORY DATA ANALYSIS RESULTS

INTERACTIVE ANALYTICS DEMO IN SCREENSHOTS

PREDICTIVE ANALYSIS RESULTS

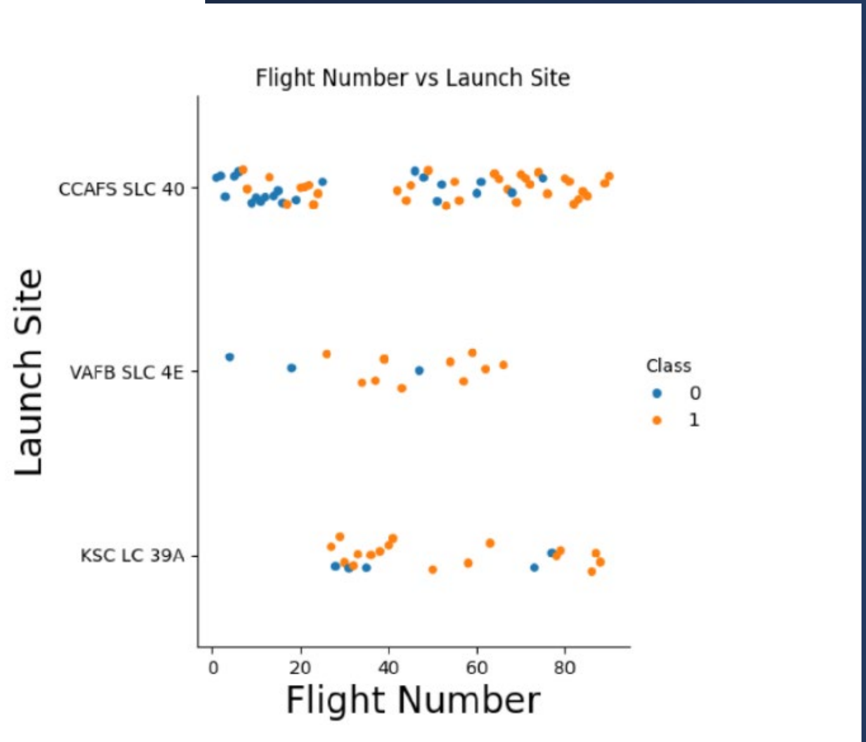
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

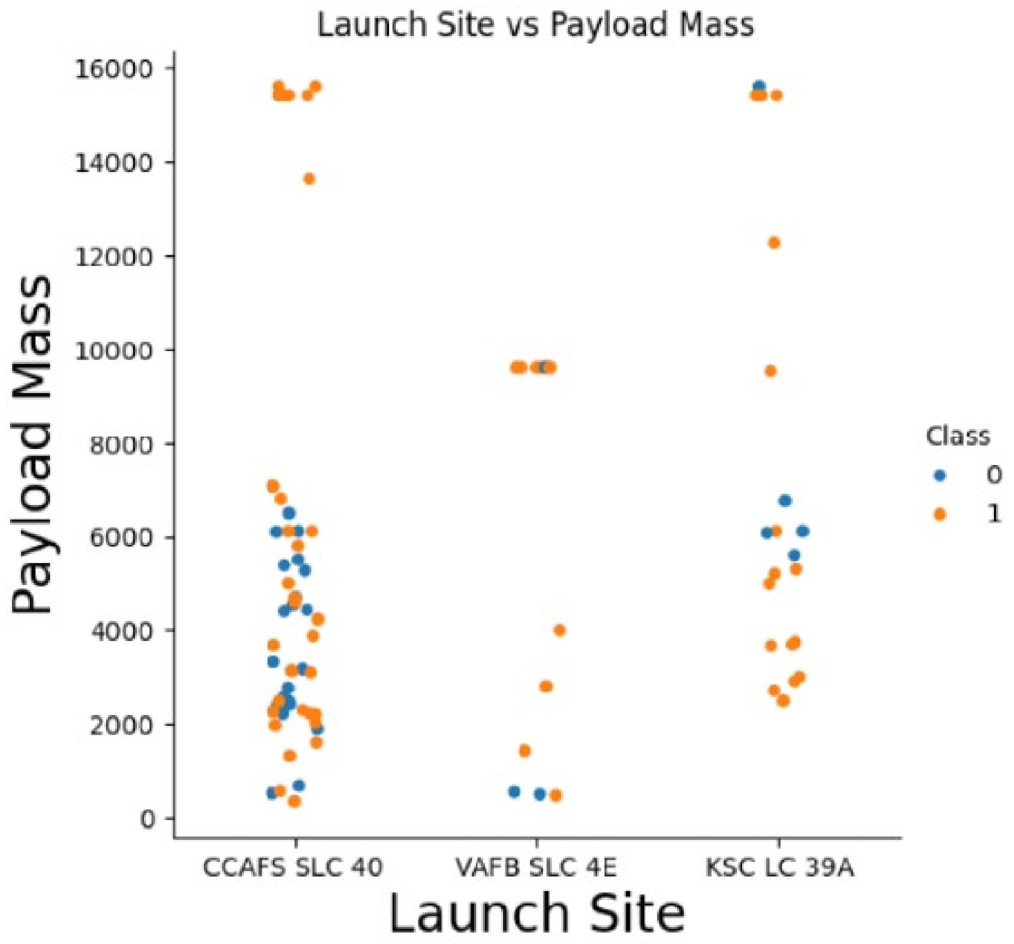
- Specific locations are used for consecutive launches as seen on the chart.

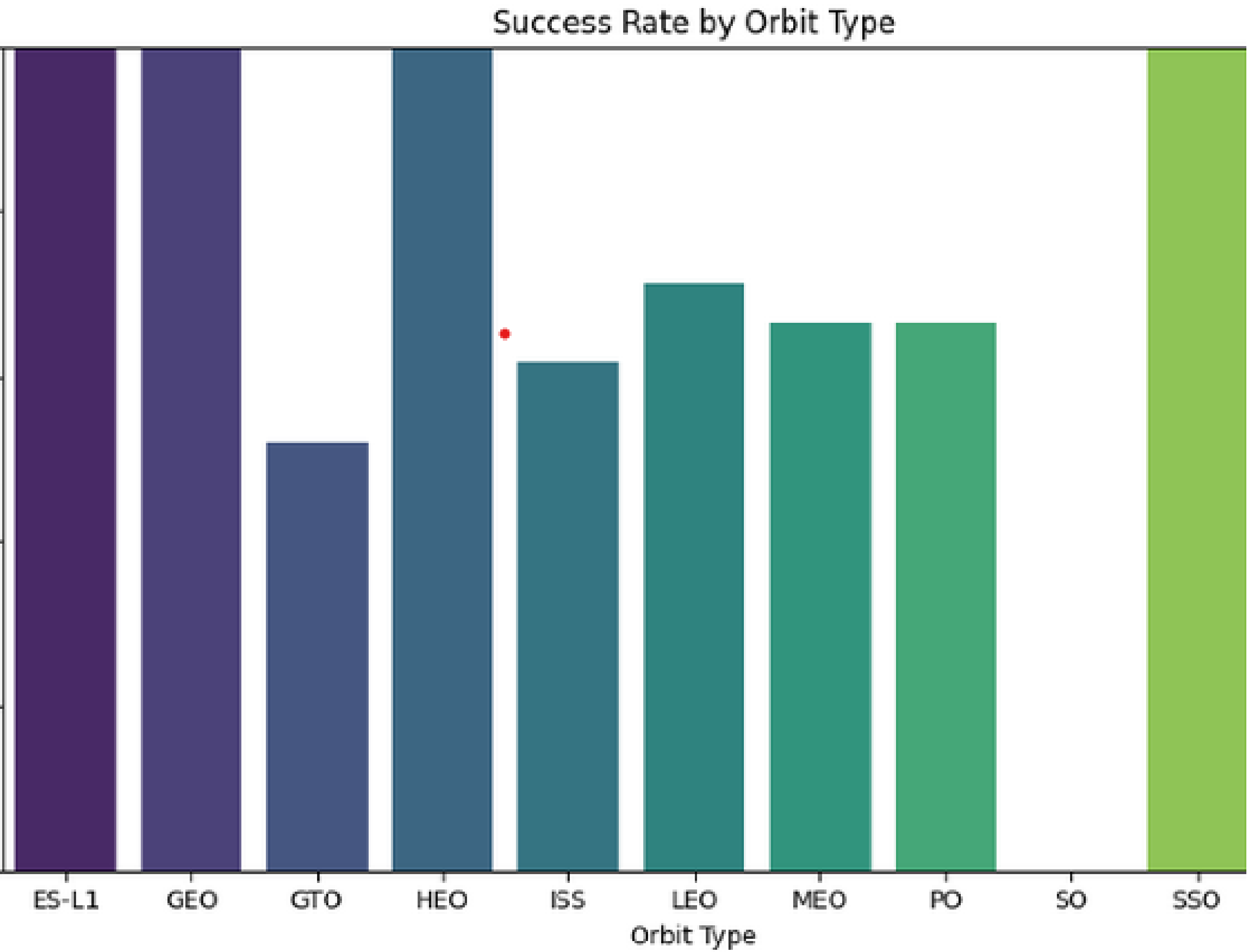
- SpaceX is seen to normally only use one location at a time and only rarely bounces between locations



Payload vs. Launch Site

- CCAFS SLC40 and KSC LC 39A have more launches than VAFB SLC 4E.
- VAFB SLC 4E has more successful outcomes compared to the other sites.
- Payloads over 8000Kg have a higher success rates with landing outcomes





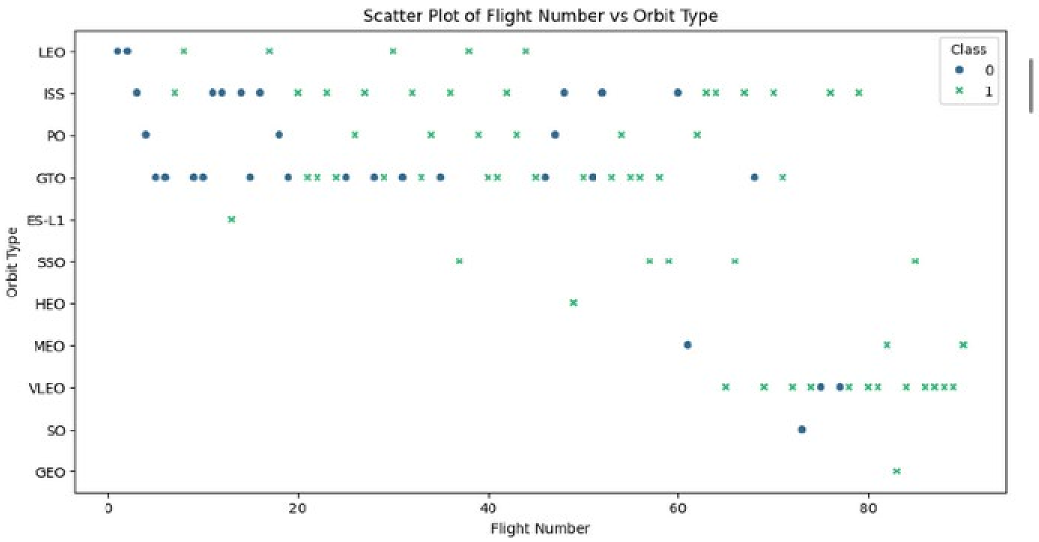
Success Rate vs. Orbit Type

- The most successful orbits by the graph are ES-LI, GEO, HEO, and SSO. These orbits have very little launches compared to the GTO, ISS, LEO, PO and VLEO so further investigation is warranted.

Flight Number vs. Orbit Type

- In more recent launches, SpaceX started to increase use of the Very Low Earth Orbit (VLEO).

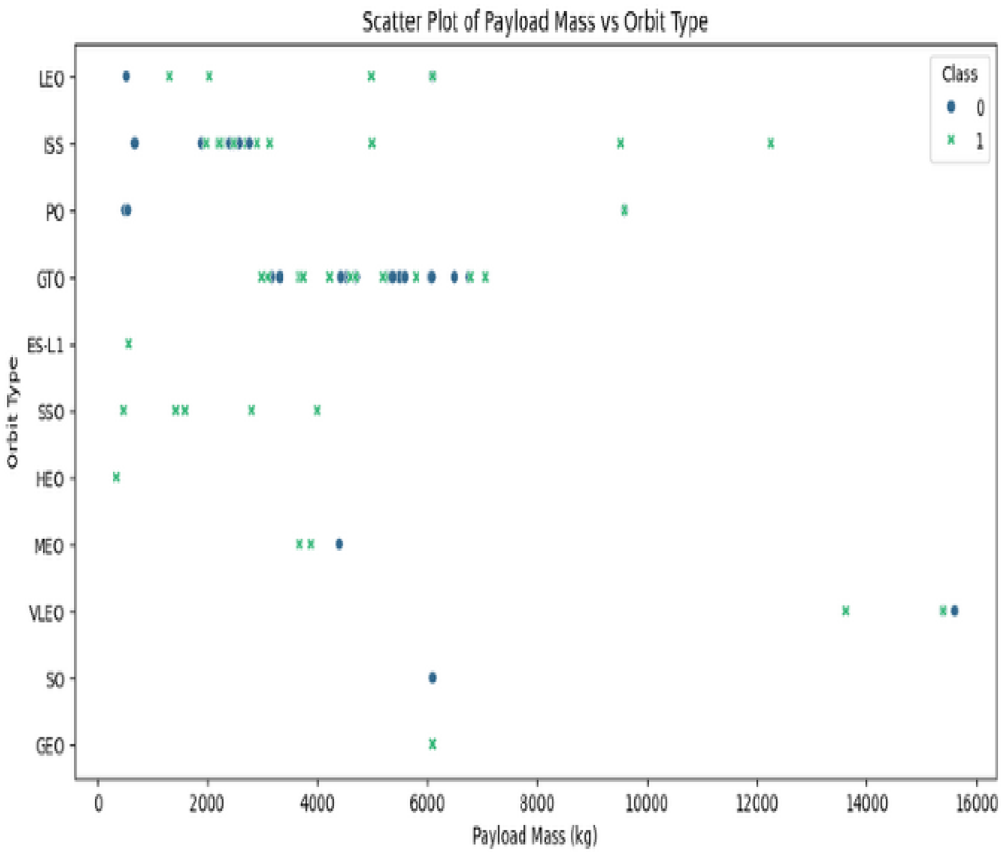
- The more recent launches also increased in having more successful landing outcomes with the first half of launches having 19 failed landing outcomes and the second half having only 11 failed



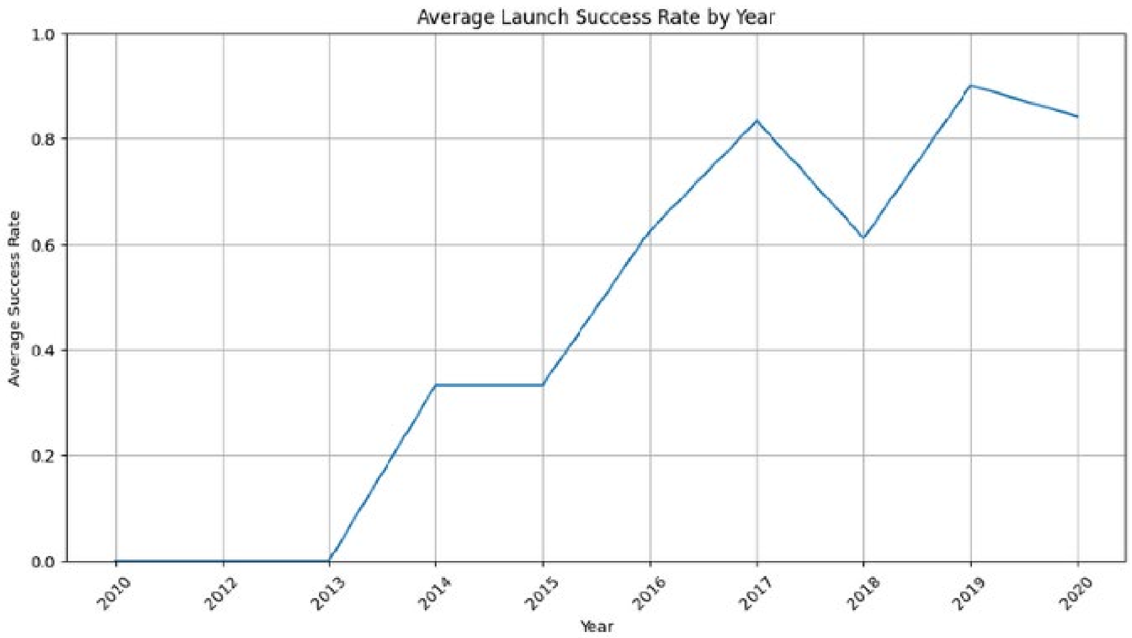
Payload vs. Orbit Type

The only orbit that could support a payload mass of over 13000Kg and has a 66% of having a successful launch outcome.

SSO has a 100% rate of successful launch outcome but cannot support a payload mass of over 4000Kg



Launch Success Yearly Trend



- SpaceX's success rate increases over time. In 2018 and 2020 were the only times that there was a decrease in success rates.

All Launch Site Names

- The unique launch sites are:
  - CCAFS LC-40
  - VAFB SLC-4E
  - KSC LC-39A
  - CCAFS SLC-40

- Using DISTINCT on the column Launch_Site, retrieved the four launch sites

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

```
%%sql
    select distinct "Launch_Site"
    from SPACEXTABLE
```

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

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

Launch Site Names Begin with 'CCA'

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

```
%%sql
    select Launch_Site
    from SPACEXTABLE
    where Launch_Site like 'CCA%'
    limit 5
```

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

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

- To get the result of the 5 records where launch sites begin with the string ‘CCA’ we create the query with WHERE Launch_Site LIKE ‘CCA%’ and add LIMIT 5 to the query

Total Payload Mass

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

- To create the query in the SELECT clause we use SUM(“PAYLOAD_MASS_KG_”) and in the WHERE clause we use “Customer” = ‘NASA (CRS)’

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

```
%%sql
SELECT SUM("PAYLOAD_MASS__KG_")
FROM SPACEXTABLE
WHERE "Customer" = 'NASA (CRS)'
```

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

SUM("PAYLOAD_MASS__KG_")
45596

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 is 2534.6666 Kg

- To retrieve the answer, we use AVG(“PAYLOAD_MASS_KG_”) in the SELECT clause and “Booster_Version” like ‘F9 v1.1%’ in the WHERE clause

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

```
%%sql
SELECT AVG("PAYLOAD_MASS__KG_")
FROM SPACEXTABLE
WHERE "Booster_Version" like 'F9 v1.1%'
```

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

AVG("PAYLOAD_MASS__KG_")
2534.6666666666665

First Successful Ground Landing Date

- The first successful landing outcome on ground pad occurred on 12/22/2015

- The way to create the query is to use MIN(Date) in the SELCET clause and “Landing_Outcome” = ‘Success (ground pad)’ in the WHERE clause

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

Hint:Use min function

```
%%sql
SELECT min(Date)
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (ground pad)'
```

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

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 is: F9 FT B1002, F9 FT B1026, F9 FT B1021.2, F9 FT B1031.2

- To retrieve the result we use DISTINCT in the SELECT clause and in the WHERE clause we use Landing_Outcome = ‘Success (drone ship)’ AND PAYLOADMASSKG > 4000 and PAYLOADMASSKG < 6000

Task 6

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
FROM SPACEXTABLE
WHERE Landing_Outcome = 'Success (drone ship)'
    AND PAYLOADMASSKG > 4000
    AND PAYLOADMASSKG < 6000
```

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

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

Total Number of Successful and Failure Mission Outcomes

- The total number of successful and failure mission outcomes are listed in the table

- To retrieve the total number we write the SELECT Landing_Outcome, COUNT(*) AS Total and add a GROUP BY Landing_Outcome

List the total number of successful and failure mission outcomes

```
%%sql
SELECT Landing_Outcome, COUNT(*) AS Total
FROM SPACEXTABLE
GROUP BY Landing_Outcome;
```

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

Landing_Outcome	Total
Controlled (ocean)	5
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	21
No attempt	1
Precluded (drone ship)	1
Success	38
Success (drone ship)	14
Success (ground pad)	9
Uncontrolled (ocean)	2

Boosters Carried Maximum Payload

- The list of the Boosters that carry the maximum payload are recorded in the list from the query

- The query is created by using DISTINCT in the SELECT query and in the WHERE “PAYLOADMASSKG” == (SELECT MAX(“PAYLOADMASSKG”) FROM SPACETABLE)

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

```
%%sql
select DISTINCT"Booster_Version"
from SPACEXTABLE
where "PAYLOADMASSKG" == (select MAX("PAYLOADMASSKG") from SPACEXTABLE)
```

```
 * 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

- The failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015 are:
  - January, F9 V1.1 B1012, CCAFS LC-40
  - April, F9 V1.1 B1015, CCAFS LC-40

- In the query we create a CASE that changes the month number to month name and in the WHERE clause we ensure that Landing_Outcome = ‘Failure (drone ship)’

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.

```
%%sql
    SELECT
    CASE SUBSTR(Date, 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
    END AS Month_Name,
    Landing_Outcome,
    Booster_Version,
    Launch_Site
FROM SPACEXTABLE
WHERE Landing_Outcome = 'Failure (drone ship)'
  AND SUBSTR(Date, 1, 4) = '2015';
```

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

Month_Name	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

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.

```
%%sql
SELECT Landing_Outcome, COUNT(*) AS Outcome_Count,
       RANK() OVER (ORDER BY COUNT(*) DESC) AS Outcome_Rank
FROM SPACEXTABLE
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY Outcome_Count DESC;
```

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

Landing_Outcome	Outcome_Count	Outcome_Rank
No attempt	10	1
Success (drone ship)	5	2
Failure (drone ship)	5	2
Success (ground pad)	3	4
Controlled (ocean)	3	4
Uncontrolled (ocean)	2	6
Failure (parachute)	2	6
Precluded (drone ship)	1	8

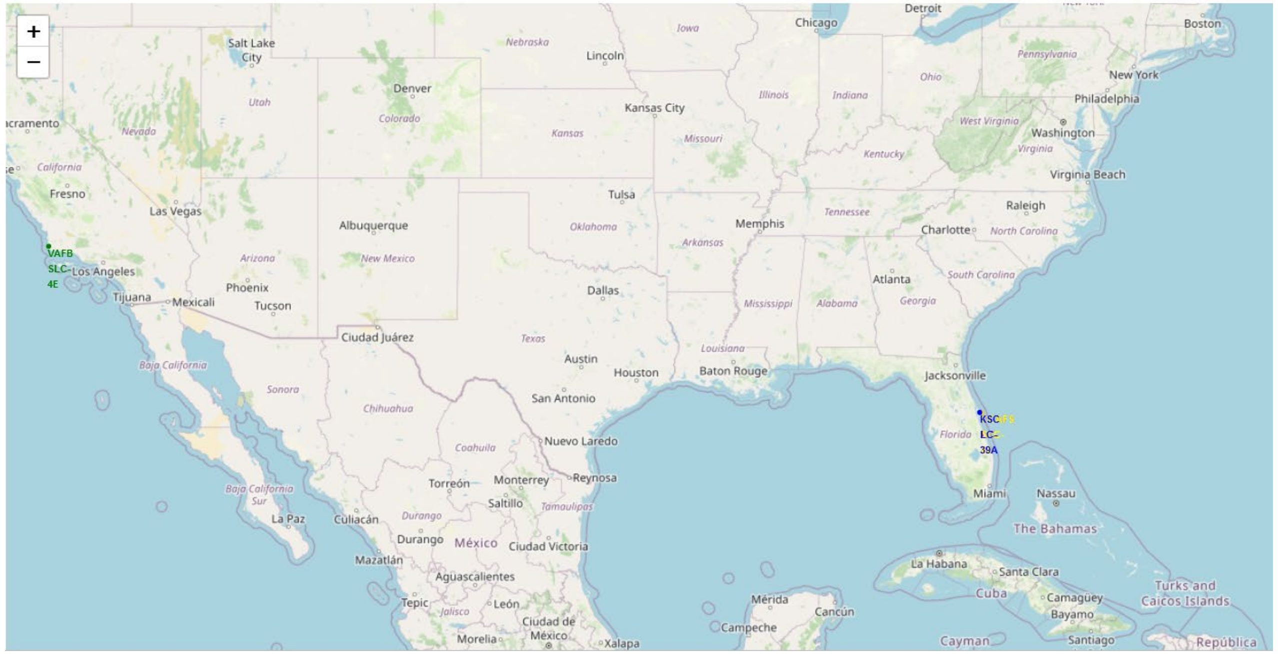
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

- To create the query we use:
  - SELECT Landing_Outcome, COUNT(*) AS Outcome_Count, Rank() OVER (ORDER BY COUNT(*) DESC) AS Outcome_Rank
  - In the WHERE clause we have Date BETWEEN ‘2010-06-04’ AND ‘2017-03-20’
  - Add a GROUP BY Landing_Outcome
  - ORDER BY Outcome_Count DESC;

Section 3

Launch Sites Proximities Analysis

SpaceX Launch Sites

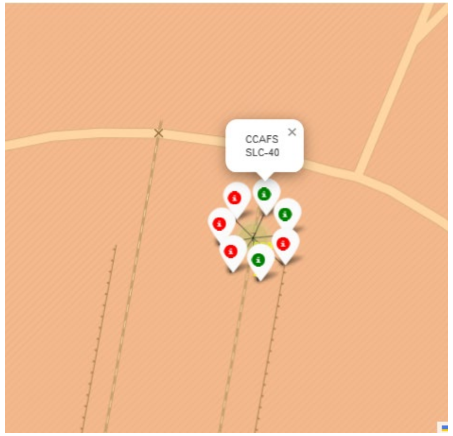


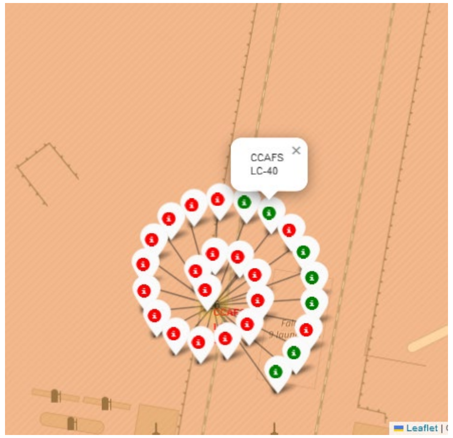
- SpaceX launch sites are located on the coast of the United States and in the Southern half of the country.

SpaceX Launch Sites

Success vs Failed Launches at Sites

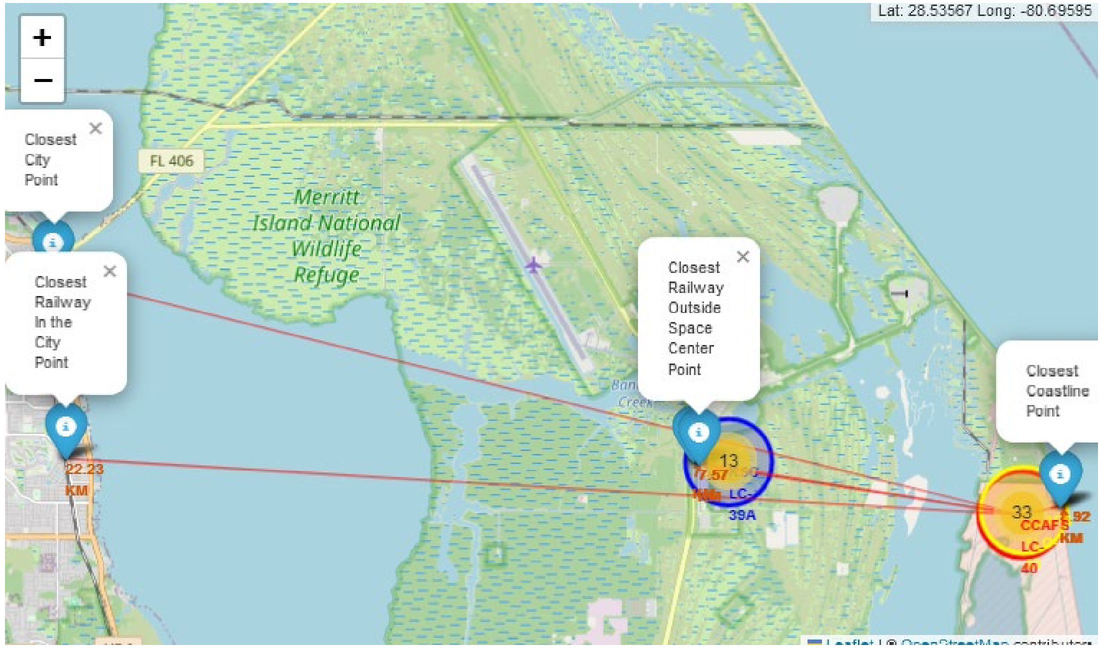
- The maps show the success and failure of landing outcomes using marker clusters. It is easy to see that KSC LC-39A has the highest success rate of launch sites.





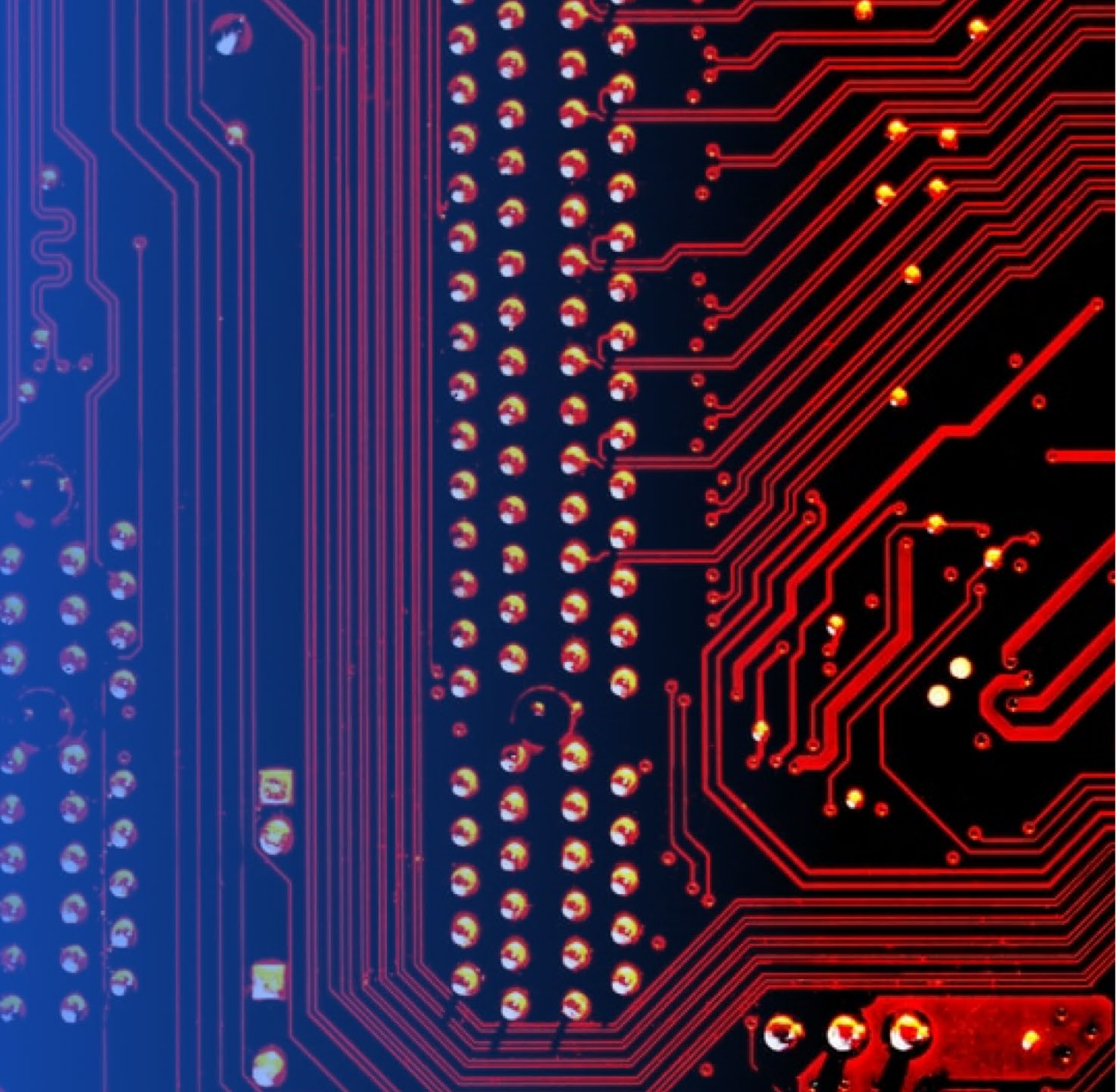
Distance from Areas of Interest to Launch Sites

- The map shows that launch sites are not near populated areas or highways. There are railways near the sites, indicating they could be used in case of evacuation and movement of goods.



Section 4

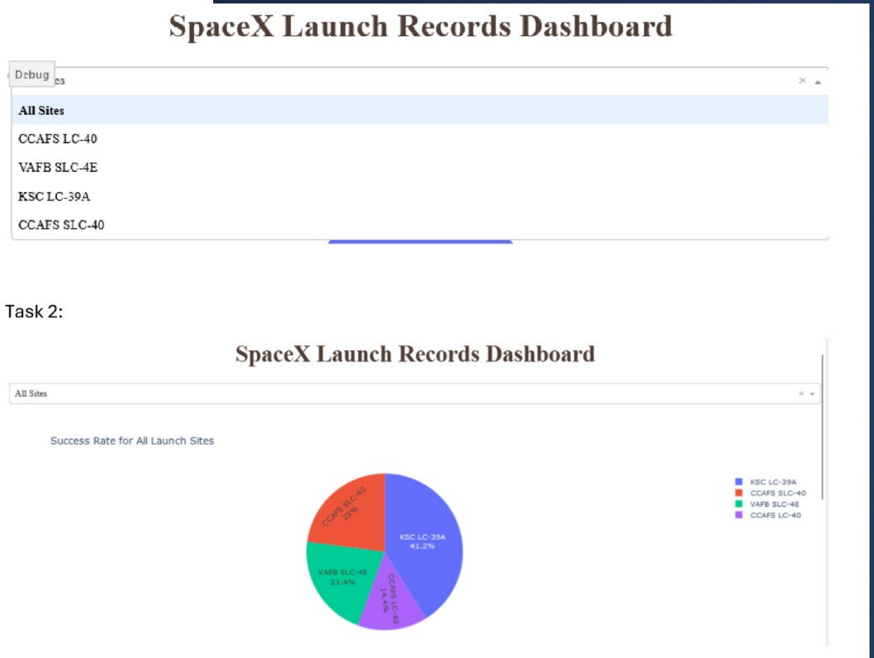
Build a Dashboard with Plotly Dash

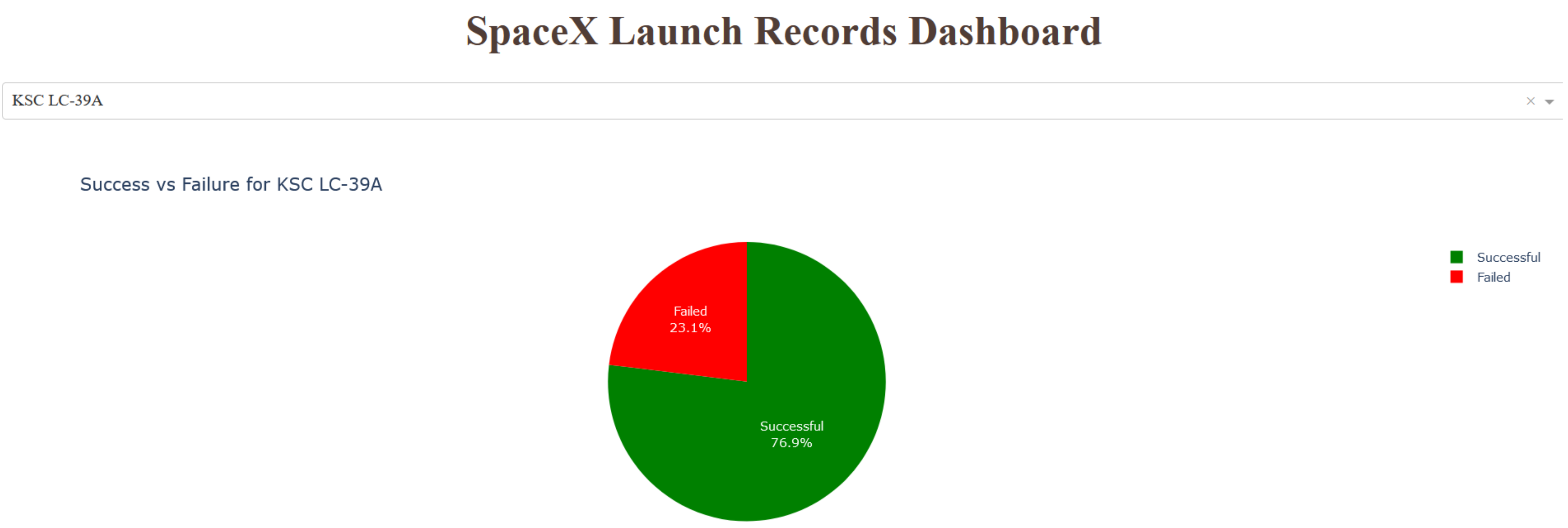


Plotly Dashboard - All Sites

- The Dashboard allows user to choose from either all sites or a specific site

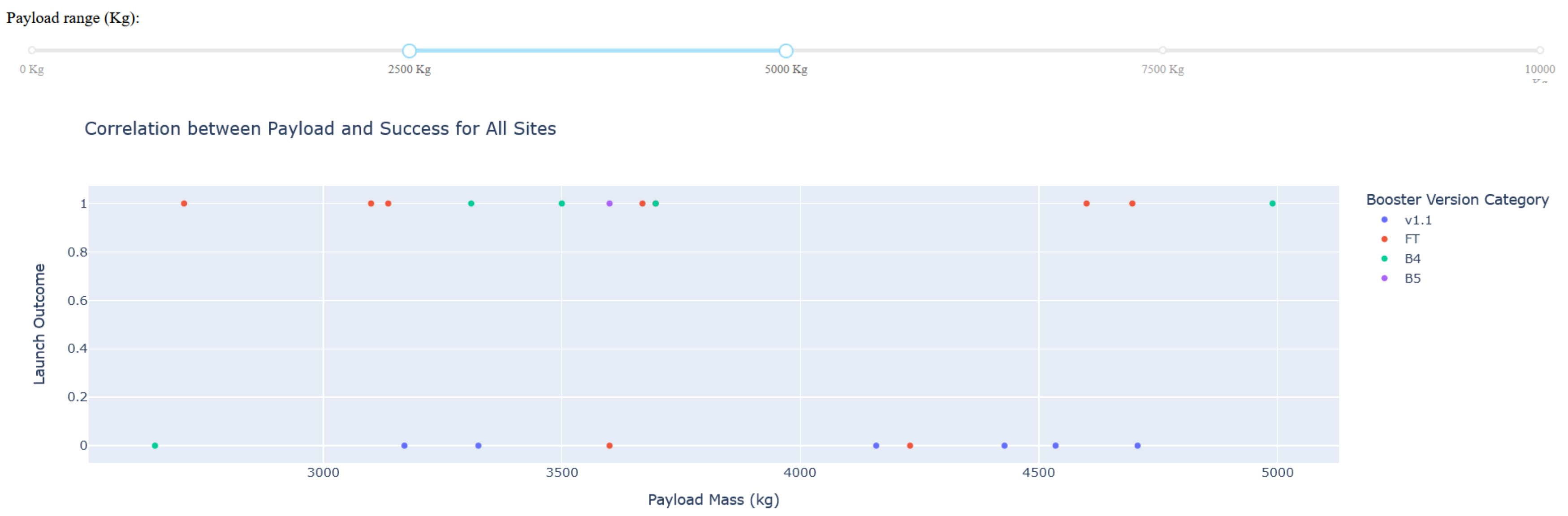
- The percent of total successful launches out of all site locations are as follows:
    - CCAFS SLC-40 with 23%
    - VAFB SLC-4E with 21.4%
    - CCAFS LC-40 with 14.4%
    - KSC LC-39A with 41.2%





Most Successful Launch Site

- The KSC LC-39A Launch Site had the highest success rate of launches compared to all other sites. Their success rate was 76.9%.



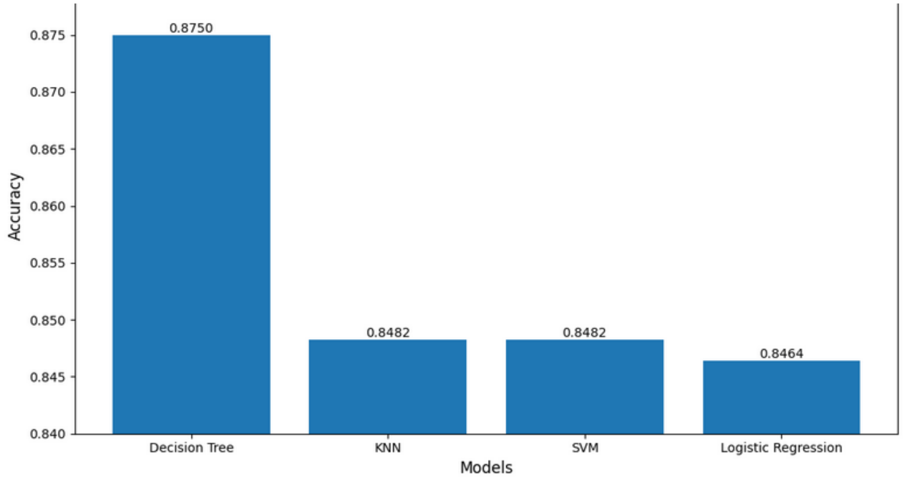
- The B5 Booster Category is the most successful when the payload mass is between 2500Kg and 5000Kg with a 100% success rate where the v1.1 made no successful launch outcomes for the same payload mass.

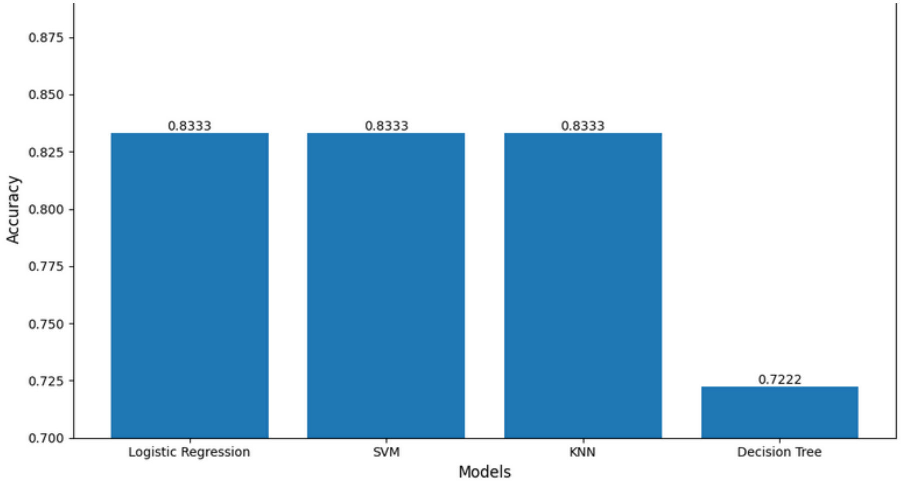
Launch Outcome vs Payload Mass (2500Kg vs 5000Kg)

Section 5

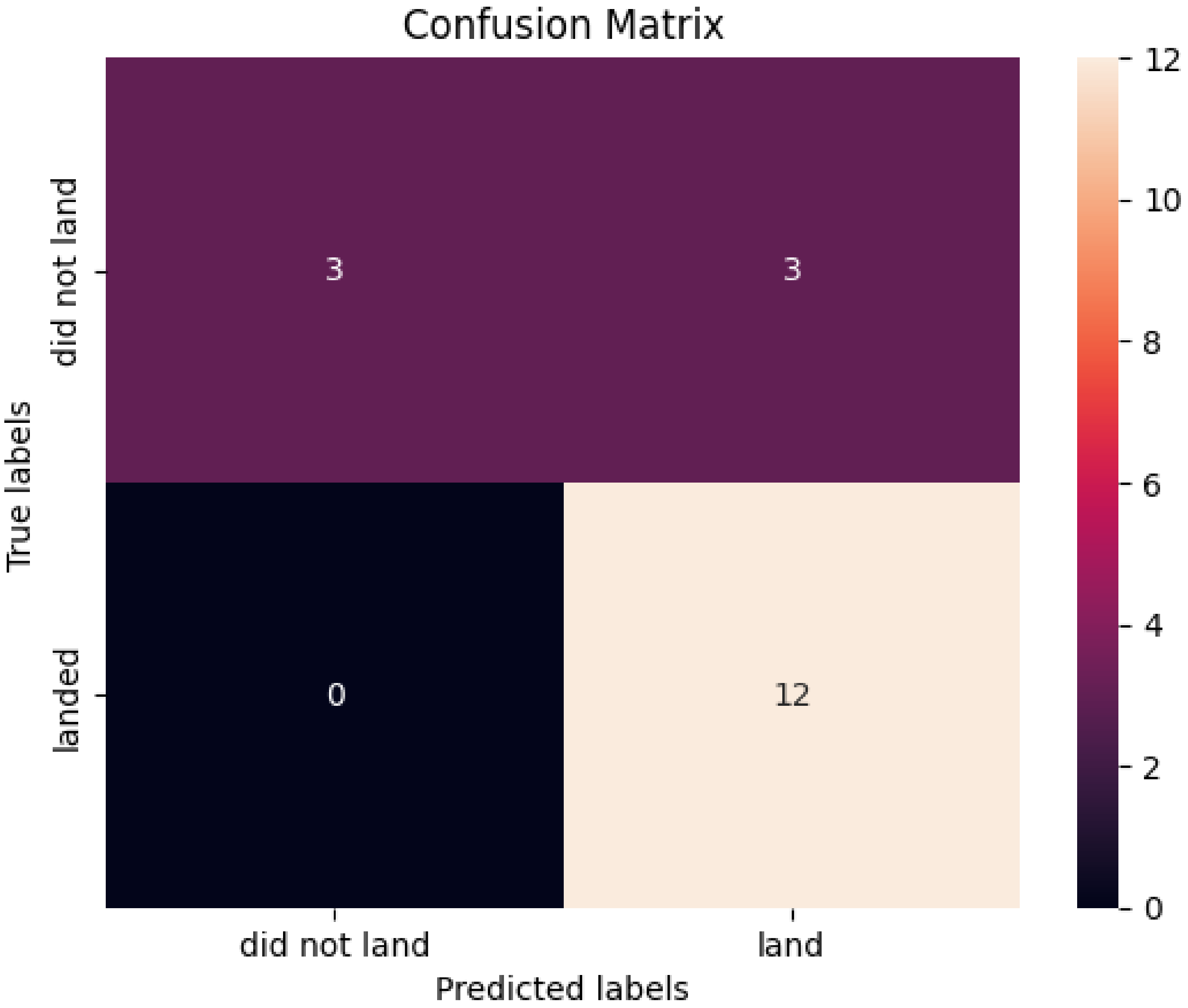
Predictive Analysis (Classification)

Classification Accuracy





- Once the test data was used on the models we can see that Logistic Regression, Support Vector Machine (SMV), K to the Nearest Neighbor (KNN) are all tied for the highest accuracy at .8333 where decision tree is at .7222



SMV Confusion Matrix

- This confusion matrix shows that the SVM model performs well in predicting when the first stage does not land (13 out of 15 correct), but it struggles more with predicting successful landings (2 out of 3 correct).

Conclusions

Model Performance: Logistic Regression, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) models all achieved the same test set accuracy of 0.8333 (83.33%).

The Decision Tree Classifier performed worse with a test set accuracy of 0.7222 (72.22%).

Generalization: The models perform well on the test set, indicating good generalization from the training data. However, the small test set size (18 samples) limits the reliability of these results.

False Positives: The logistic regression model showed a tendency for false positives, incorrectly predicting landings for some cases where landings did not occur.

Hyperparameter Tuning: GridSearchCV was effective in finding optimal hyperparameters for each model, improving their performance.

Data Preprocessing: Standardizing the data using StandardScaler was beneficial for model performance.

Class Imbalance: The confusion matrices suggest a possible class imbalance in the dataset, with more "did not land" cases than "landed" cases.

Further Investigation: Given the identical performance of multiple models, it would be worth exploring ensemble methods or more advanced techniques to potentially improve accuracy. Collecting more data could provide more reliable insights and potentially improve model performance.

Appendix

Code Snippets from Plotly Dash

```python
# TASK 1: Add a dropdown list to enable Launch Site selection
dcc.Dropdown(
    id='site-dropdown',
    options=[{'label': 'All Sites', 'value': 'ALL'}] +
            [{'label': site, 'value': site} for site in spacex_df['Launch Site'].unique()],
    value='ALL',
    placeholder='Select a Launch Site here',
    searchable=True
),
html.Br(),
```

```python
# TASK 2:
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
@app.callback(
    Output(component_id='success-pie-chart', component_property='figure'),
    Input(component_id='site-dropdown', component_property='value')
)
def update_pie_chart(selected_site):
    filtered_df=spacex_df
    if selected_site == 'ALL':
        fig = px.pie(spacex_df, values='class',
        names='Launch Site',
        title='Total Success Launches by Site')
    else:
        filtered_df = spacex_df[spacex_df['Launch Site'] == selected_site].copy()
        filtered_df['class']=filtered_df['class'].map({0:'Failed', 1:'Successful'})
        fig = px.pie(filtered_df, names='class', title=f'Success vs Failure for {selected_site}',color='class',
        color_discrete_map={'Failed': 'red','Successful':'green'})
        fig.update_traces(textinfo='percent+label')
    return fig
```

```python
# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as output
@app.callback(
    Output(component_id='success-payload-scatter-chart', component_property='figure'),
    [Input(component_id='site-dropdown', component_property='value'),
     Input(component_id='payload-slider', component_property='value')]
)
def update_scatter_plot(selected_site, payload_range):
    low, high = payload_range
    filtered_df = spacex_df[(spacex_df['Payload Mass (kg)'] >= low) & (spacex_df['Payload Mass (kg)'] <= high)]

    if selected_site == 'ALL':
        fig = px.scatter(
            filtered_df,
            x='Payload Mass (kg)',
            y='class',
            color='Booster Version Category',
            title='Correlation between Payload and Success for All Sites',
            labels={'class': 'Launch Outcome'}
        )
    else:
        site_filtered_df = filtered_df[filtered_df['Launch Site'] == selected_site]
        fig = px.scatter(
            site_filtered_df,
            x='Payload Mass (kg)',
            y='class',
            color='Booster Version Category',
            title=f'Correlation between Payload and Success for {selected_site}',
            labels={'class': 'Launch Outcome'}
        )
    return fig
```

```python
# TASK 3: Add a slider to select payload range
dcc.RangeSlider(
    id='payload-slider',
    min=0,
    max=10000,
    step=1000,
    value=[min_payload, max_payload],
    marks={0: '0 Kg', 2500: '2500 Kg', 5000: '5000 Kg', 7500: '7500 Kg', 10000: '10000 Kg'}
),
```

Thank you!