Winning Space Race with Data Science

Vivian Lin
2022/03/24

Outline

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

Executive Summary

- The SpaceX launch data were gathered from SpaceX REST API and Web Scrapping technique.
- Then, the launch data were analyzed with SQL and visualized with scatterplots, bar charts, line graphs, interactive Folium maps, and pie charts and scatterplots on Dashboard.
- Finally, different machine learning regression and classification models were used to predict the success rate of first stage rocket landing.
- SpaceX has become increasingly successful in landing their rockets (whether on ground pad or drone ship), but payload mass and orbit types must be taken into consideration.

Introduction

- SpaceX is competitive in space exploration due to its ability to reuse the first stage during rocket launches. The first stage rocket is able to land and be recovered, thus reducing spending of rocket launches.
- The success rate of first stage rocket landing will determine the cost of a launch and help alternate companies determine whether it's beneficial to bid against SpaceX for a rocket launch.
- Question to be answered: How successful is SpaceX at landing their Falcon 9 first stage rocket?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX launch data were collected from SpaceX REST API and web scrapping Wikipedia page.
- Perform data wrangling
  - Rockets with extra boosters and multiple payloads were excluded from analysis.
  - Empty data points for Payload Mass column were replaced by average.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using regression and classification models
  - Logistic Regression, Support Vector Machines (SVM), Decision Tree, and k-Nearest Neighbors models were fitted to the data.
  - GridSearchCV was used to find best parameters and confusion matrix was used to find accuracy.

Methodology

Flowchart

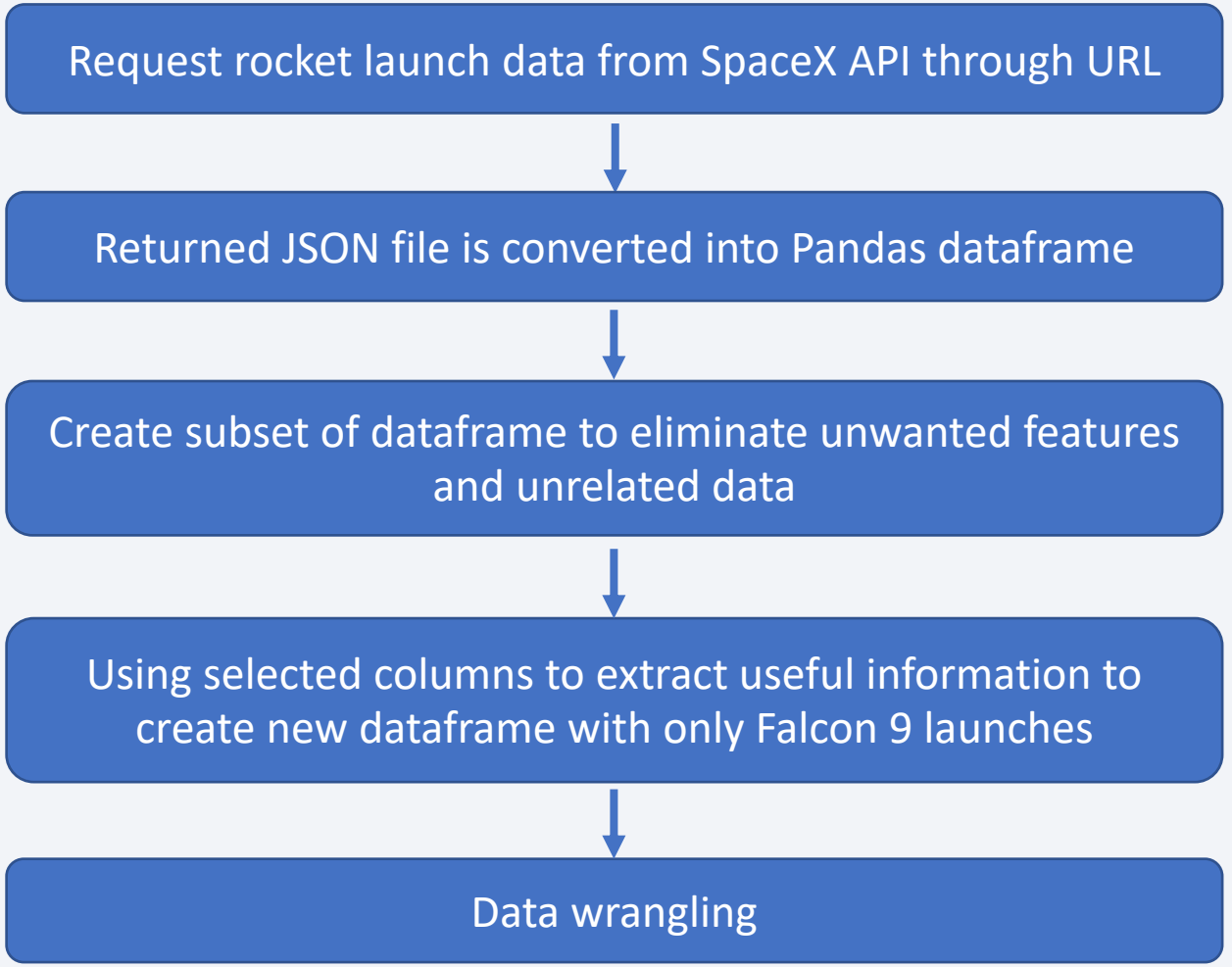
Data Collection

- Two Methods of Data Collection:
  - Using API from SpaceX REST API
  - Using web scrapping from Wikipedia page

Data Collection – SpaceX API

- SpaceX rocket launch data were requested from SpaceX API and converted into dataframe.
- Useful data, such as flight number, date, booster version, payloadmass, orbit, launchsite coordinates, and landing pad name, were extracted.

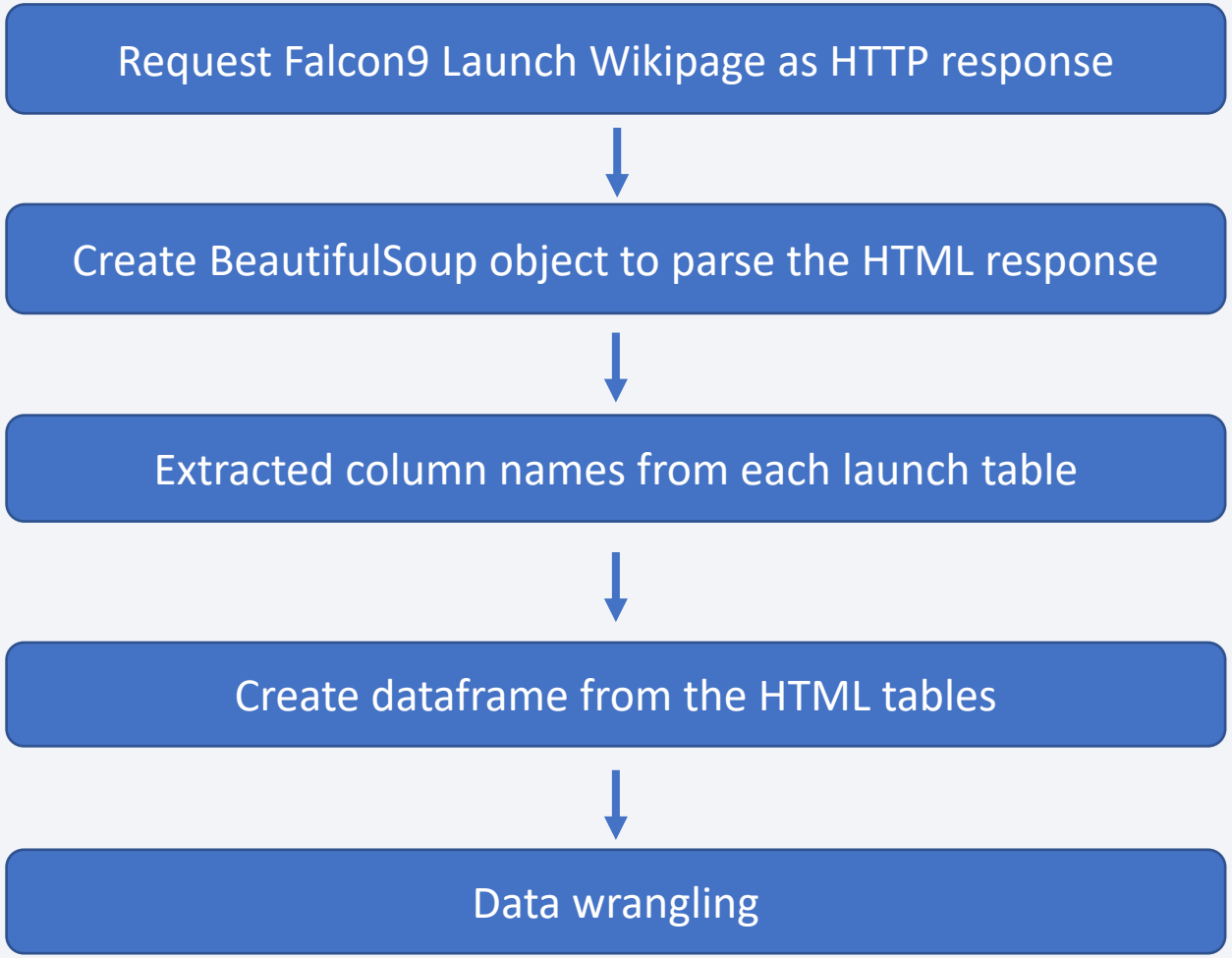
- GitHub URL https://github.com/vivianlin9887/EDX-IBM-Data-Science-Certification/blob/master/CapstoneProject/Data_Collection_API.ipynb



Data Collection - Scraping

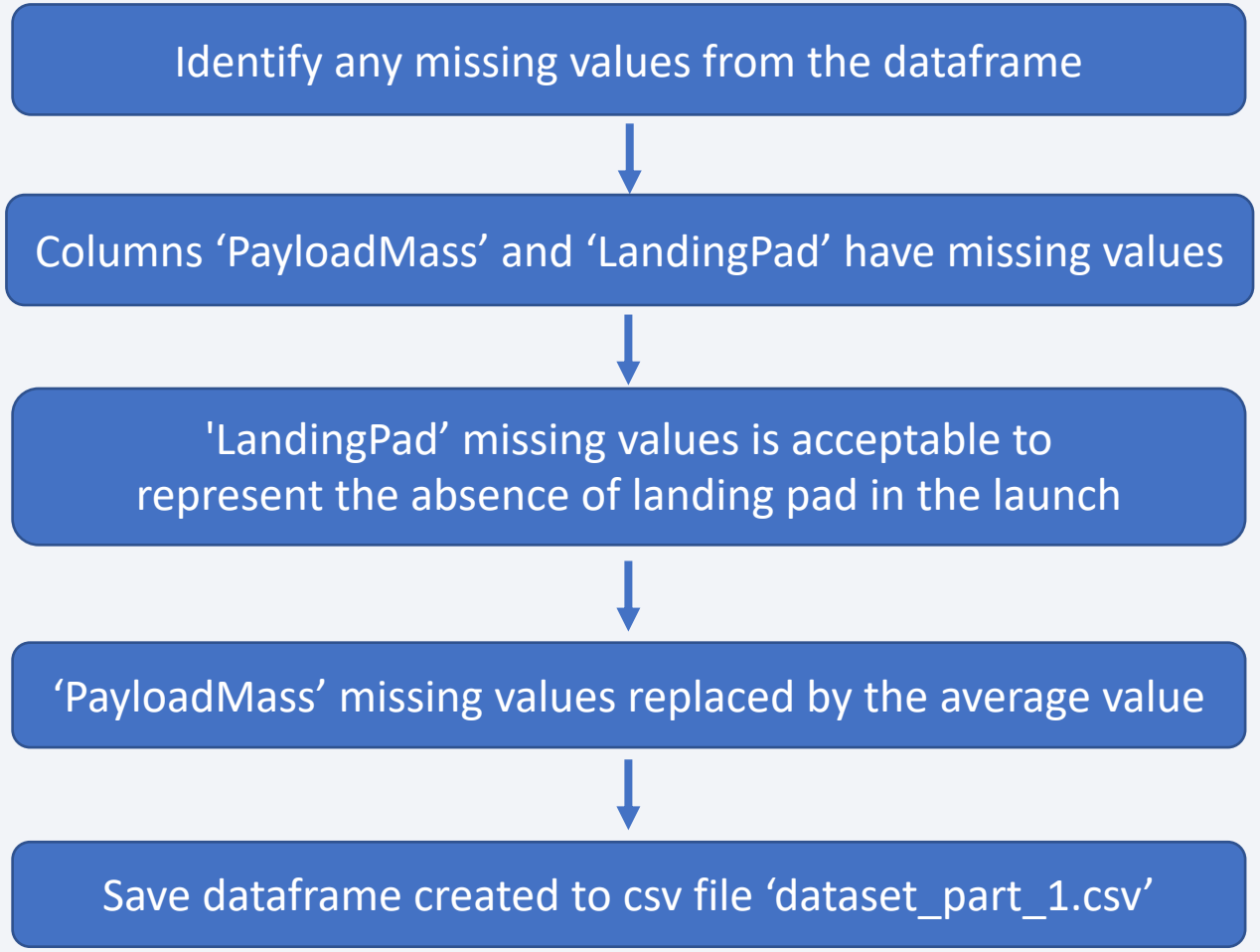
- SpaceX Falcon 9 launches information were extracted from the Wikipedia page and from information extracted, a dataframe was built.

- GitHub URL https://github.com/vivianlin9887/EDX-IBM-Data-Science-Certification/blob/master/CapstoneProject/Data_Collection_Web_Scraping.ipynb



Data Wrangling

- Identify two columns with missing values from the dataframe and determine missing value from one column must be dealt with.
- Calculated the average payload mass to replace 5 missing values in ‘PayloadMass’ column.
- GitHub URL:
  - https://github.com/vivianlin9887/EDX-IBM-Data-Science-Certification/blob/master/CapstoneProject/Data_Collection_API.ipynb
  - https://github.com/vivianlin9887/EDX-IBM-Data-Science-Certification/blob/master/CapstoneProject/EDA.ipynb



EDA with Data Visualization

- Three types of charts were used:
  - Scatter plots were used to track relationship between flight number, launch site, payload mass, and orbit type.
  - Bar chart was used to track relationship between the successful launches and orbit type.
  - Line chart was used to follow the progression of success rate as time passed.
- GitHub URL: https://github.com/vivianlin9887/EDX-IBM-Data-Science-Certification/blob/master/CapstoneProject/EDA_with_Data_Visualization.ipynb

EDA with SQL (part 1)

- 10 Queries in Total:
  - Queried unique launch sites used.
  - Queried launch sites that began with “KSC”.
  - Queried total payload mass carried by boosters launched by customer ‘NASA (CRS)’.
  - Queried average payload mass carried by booster version ‘F9 v1.1’.
  - Queried date of first successful landing outcome in drone ship.
- GitHub URL: https://github.com/vivianlin9887/EDX-IBM-Data-Science-Certification/blob/master/CapstoneProject/EDA_with_SQL.ipynb

EDA with SQL (part 2)

- 10 Queries in Total:
  - Queried names of boosters that successfully landed on ground pad with payload between 4000-6000 kg.
  - Queried total number of success and failed mission outcomes.
  - Queried names of booster versions that carried the maximum payload mass.
  - Queried months, booster version, and launch site of any successful ground pad landing in the year 2017.
  - Queried the numbers of successful landing outcomes between 2010-06-04 and 2017-03-20 (almost seven years) and rank counts in descending order.
- GitHub URL: https://github.com/vivianlin9887/EDX-IBM-Data-Science-Certification/blob/master/CapstoneProject/EDA_with_SQL.ipynb

Build an Interactive Map with Folium

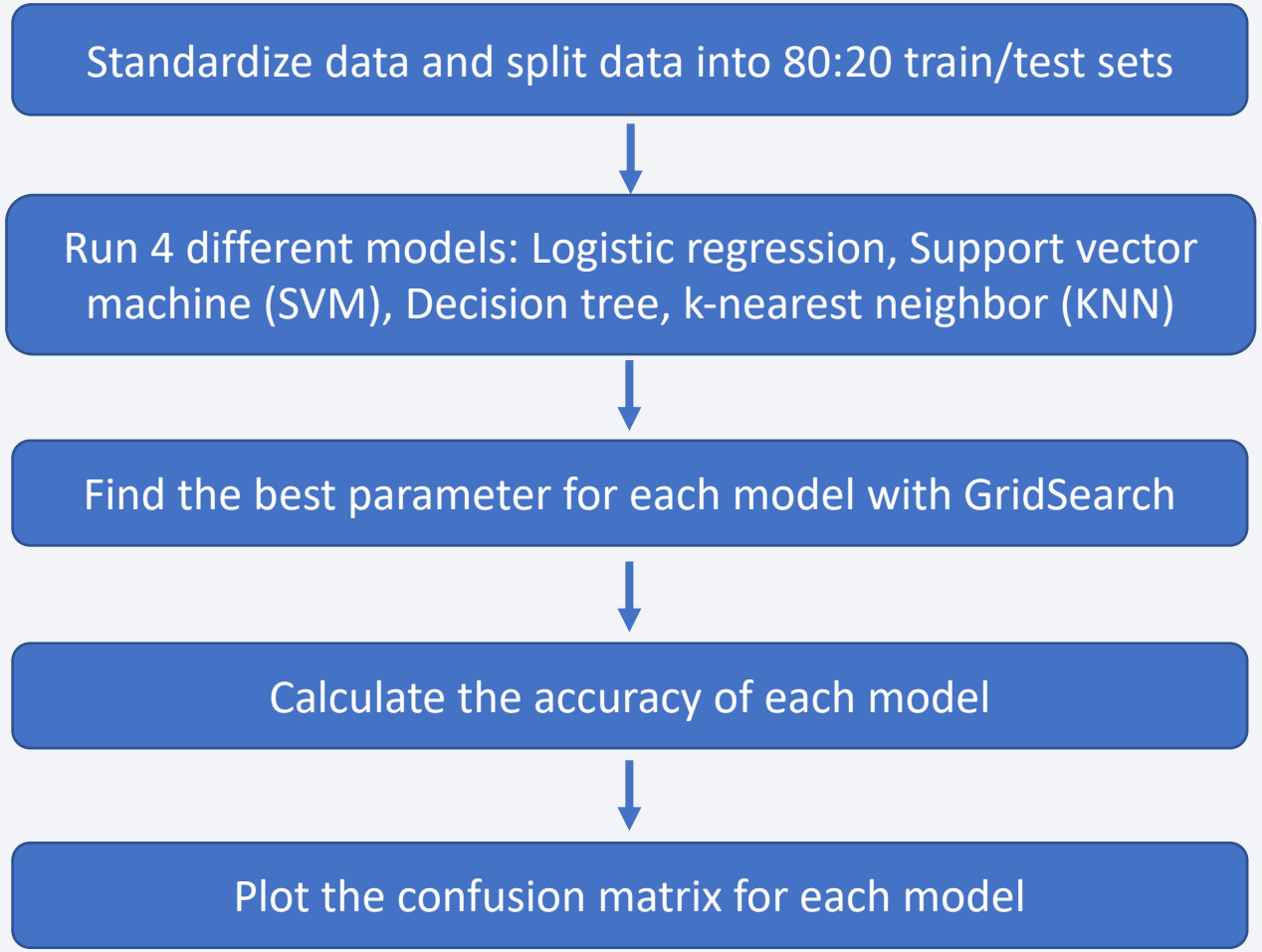
- Three folium maps were created to show:
  - Locations of each launch site,
  - Success rate of launches at each launch site, and
  - Distance to the closest city, railway, highway, and coastline.
- Locations are labeled by red circles and names of launch site are displayed in red.
- Launches at each location are color-labeled by its success/failure status.
- The distance from each launch site to its closest city, railway, highway, and coastline are calculated and the shortest distance of each is displayed on the map.
- GitHub URL: https://github.com/vivianlin9887/EDX-IBM-Data-Science-Certification/blob/master/CapstoneProject/Folium_Interactive_Visuals.ipynb
  - Folium maps cannot be displayed on GitHub, please see screenshots included in Launch Sites Proximities Analysis.

Build a Dashboard with Plotly Dash

- The interactive dashboard is divided into two parts:
  - Pie Chart to show the success rate of all sites and each launch site.
  - Slider bar for payload range that shows the relationship between payload mass and success rate for different booster versions for all sites and each launch site.
- User can use the dashboard to determine the most successful launch site and whether the different booster versions used has relationship between its payload mass and success rate.
- GitHub URL: https://github.com/vivianlin9887/EDX-IBM-Data-Science-Certification/blob/master/CapstoneProject/Interactive_Dashboard.py

Predictive Analysis (Classification)

- After data is standardized, data set is split 80:20 into training and testing sets.
- Four models were evaluated for accuracy and confusion matrices were plotted.
- GitHub URL: https://github.com/vivianlin9887/EDX-IBM-Data-Science-Certification/blob/master/CapstoneProject/ML_Prediction.ipynb
  - Somehow the Jupyter Notebook didn't display my outputs starting from Task 1, please see Appendix B for screenshots

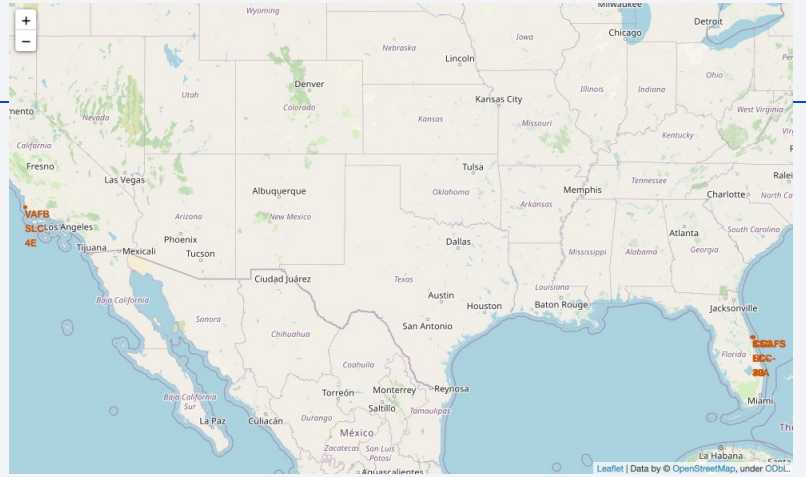


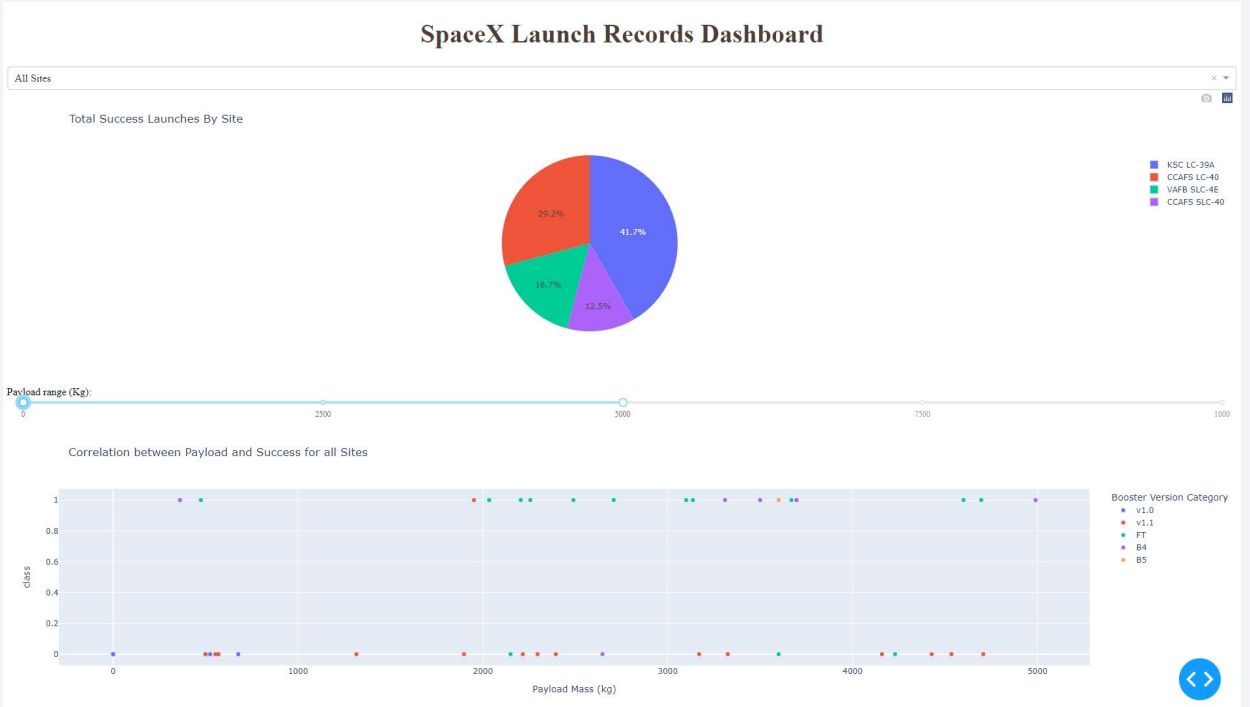
Results: Exploratory data

- There seems to be a relationship between the payload mass and mission outcome for several orbits.
  - LEO, ISS, and PO orbits have higher success rate as payload mass increases.
  - MEO and VLEO orbits have higher success rate as payload mass decreases.
- Overall, there is a general increase in successful missions since 2013.
- Successful landings on ground pad and drone ship had been achieved over the years and with increasingly success.
- Specialized booster versions had been developed to carry heavy payloads.

Results: Interactive Analytics

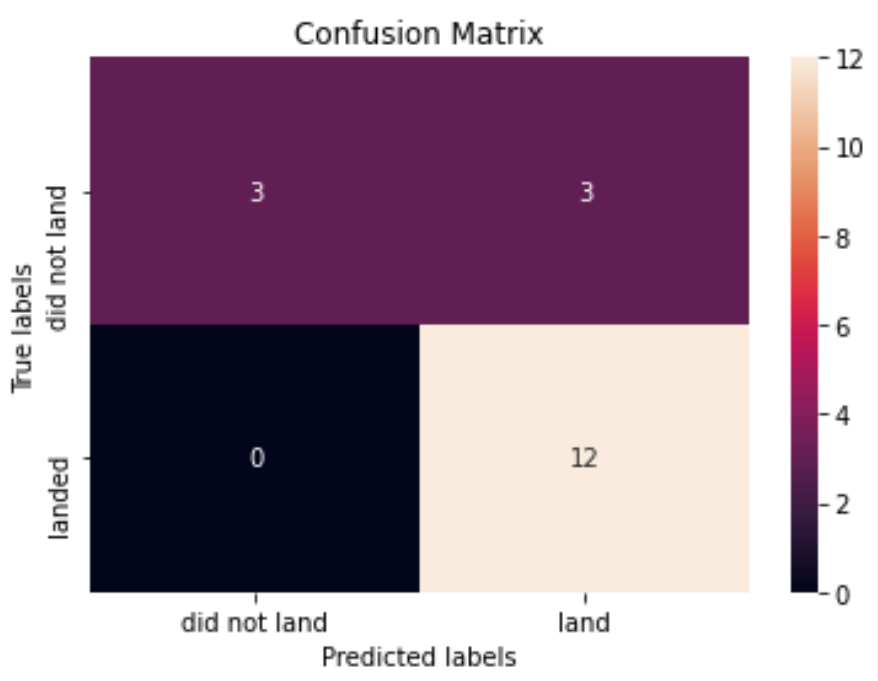
- Proximities Analysis
  - All launch sites are at least 35km away from its closest city and are all very close to the coastlines.
- Dashboard Analysis
  - Launch site KSC LC-39A had the most counts of successful missions.
  - Different booster versions are used to carry different payload mass, but relationship between successful mission outcome and weight of payload remains unclear for most booster versions.

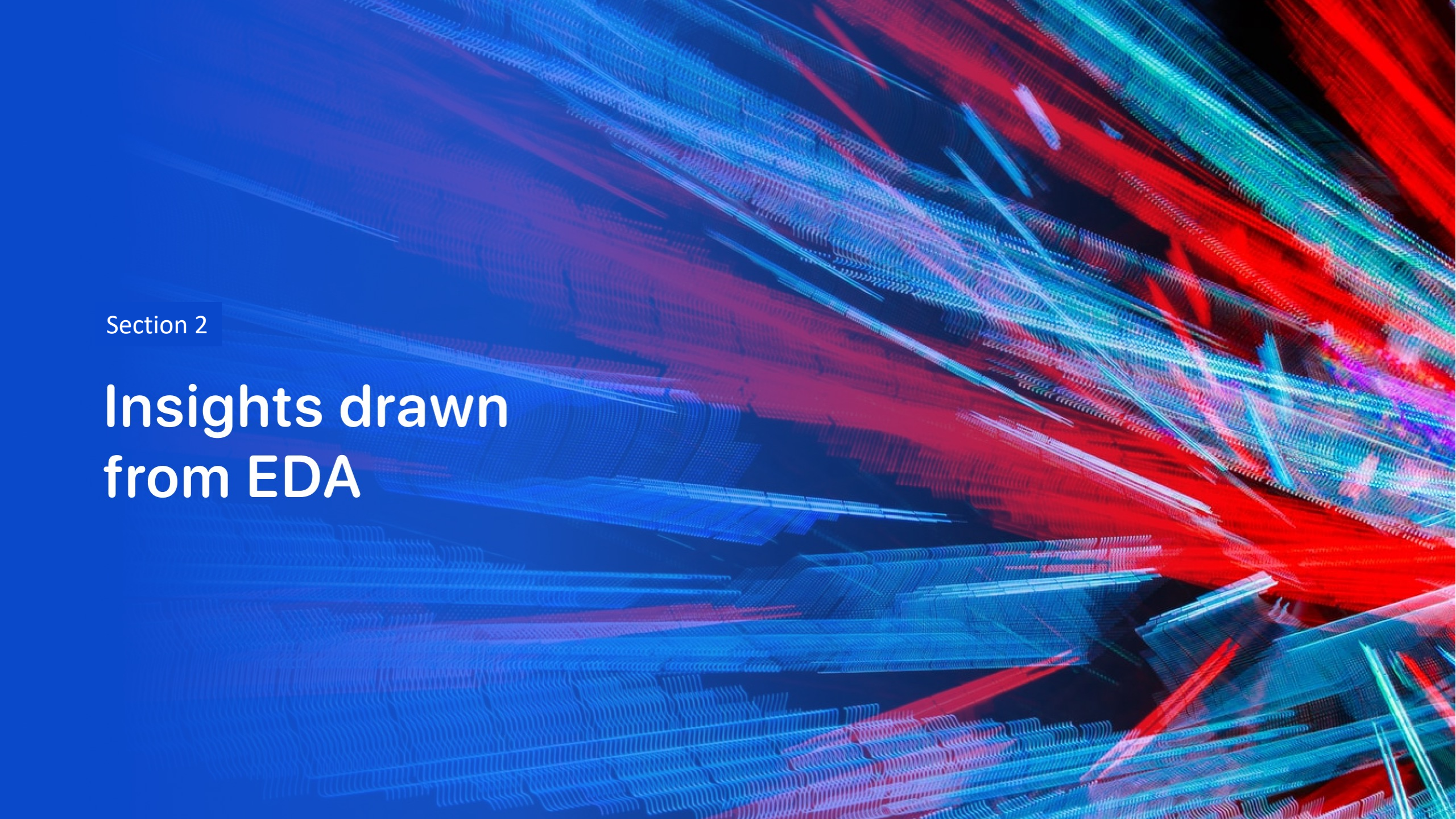




Results: Predictive Analysis

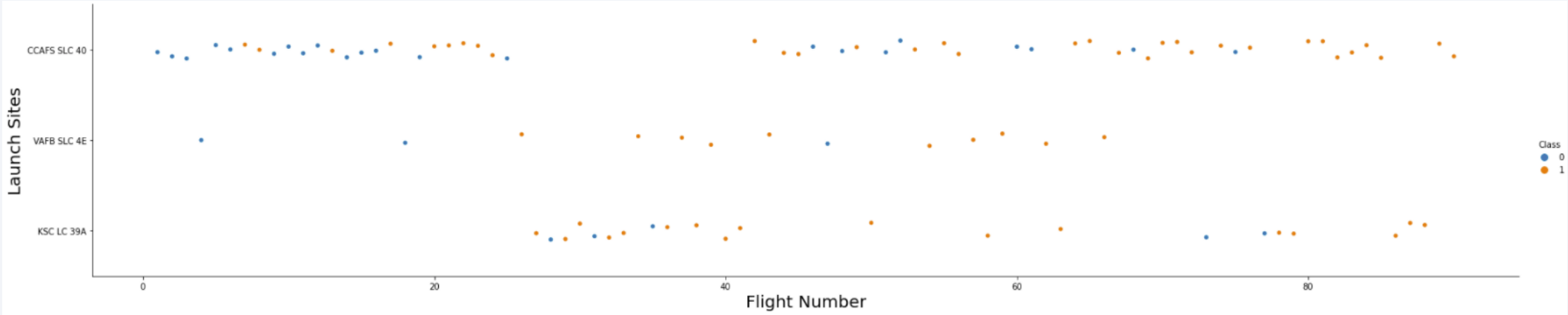
- Predictive analysis yielded that all models predict with equal accuracy (0.83), but from GridSearchCV, decision tree model has the highest accuracy score with the optimized parameters (0.875) and logistic regression model has the lowest accuracy score with optimized parameters (0.846).




Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

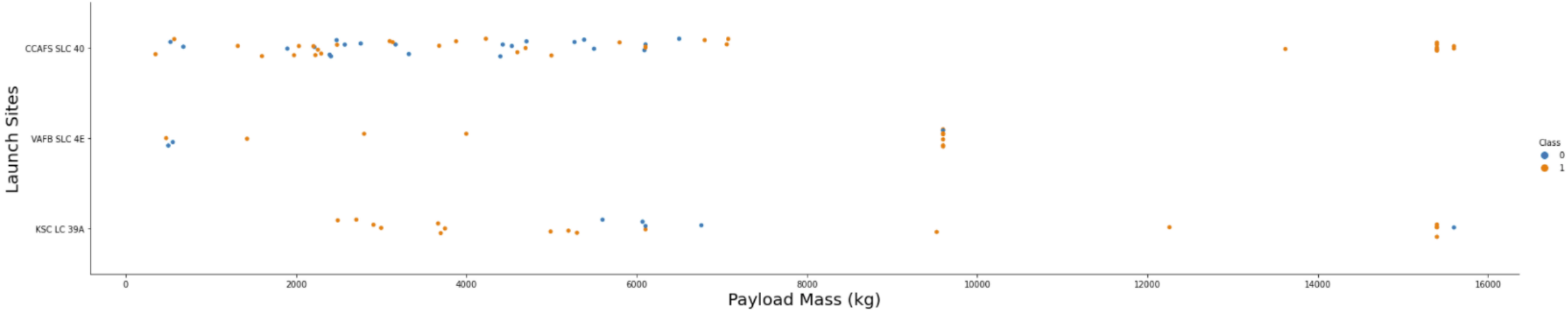
Scatter plot of Flight Number vs. Launch Site



- Scatter plot was used to determine if the launch sites have different success rates.
- The three unique launch sites are: CCAFS SLC 40, VAFB SLC 4E, and KSC LC 39A.
- Successful landing outcomes are denoted by orange dots, whereas failed landing outcomes are denoted by blue dots on the scatter plot.
- The success rate of launch site CCAFS SLC 40 is lower than the other two launch sites.

Payload vs. Launch Site

Scatter plot of Payload Mass vs. Launch Site

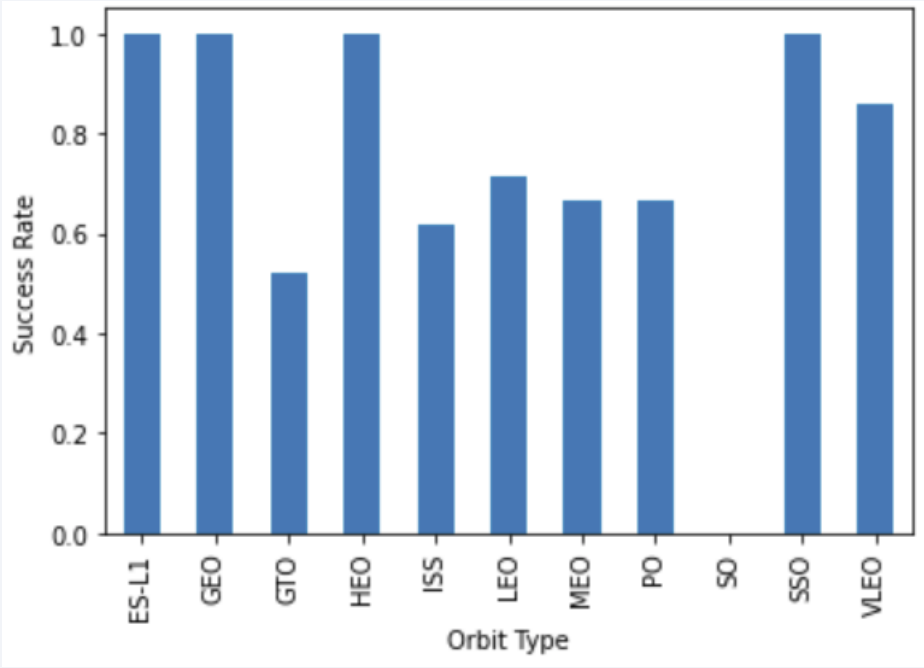


- Scatter plot was used to determine if payload mass affected the success rate at each launch site (orange dots indicate success and blue dots indicate failure).
- Launch site CCAFS SLC 40 launched most of its rockets with payload less than 8000kg and have much success when rockets have heavy payload.
- Launch site VAFB SLC 4E did not launch any rockets with payload more than 10,000kg.
- Launch site KSC LC 39A has most of its failures with rockets with payload around 6000kg.

Success Rate vs. Orbit Type

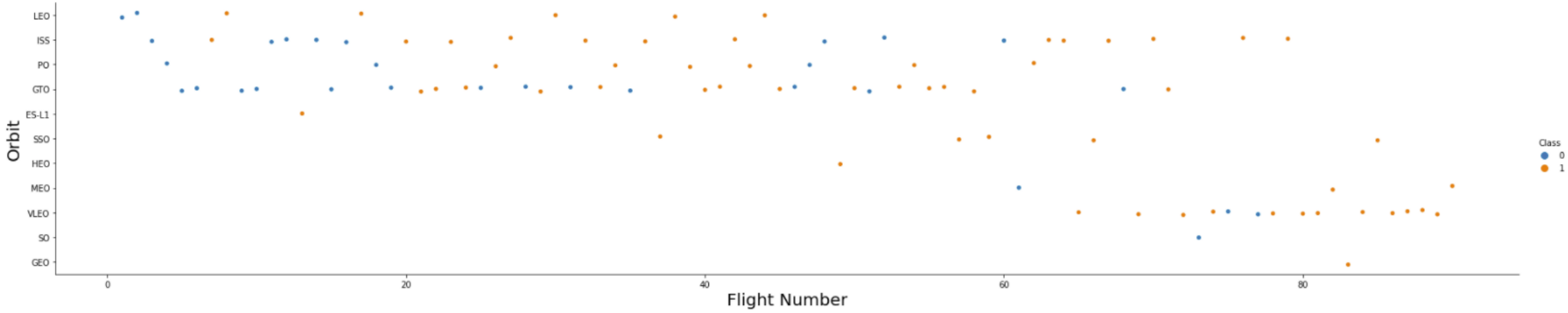
- The success rates were calculated for 11 orbits to determine if relationship exists between them.
- There are 4 orbits with 100% success rate: ES-L1, GEO (geosynchronous orbit), HEO (highly elliptical orbit), and SSO* (Sun-synchronous orbit).
- SO* (also the Sun-synchronous orbit) has no successful missions. This is interesting to note.
- The bar graph needs to be taken with scatter plot since it doesn't show how many samples each orbit has.
- *For future analysis, SO and SSO should be considered in the same category.

Bar chart of the Success Rate of Each Orbit Type



Flight Number vs. Orbit Type

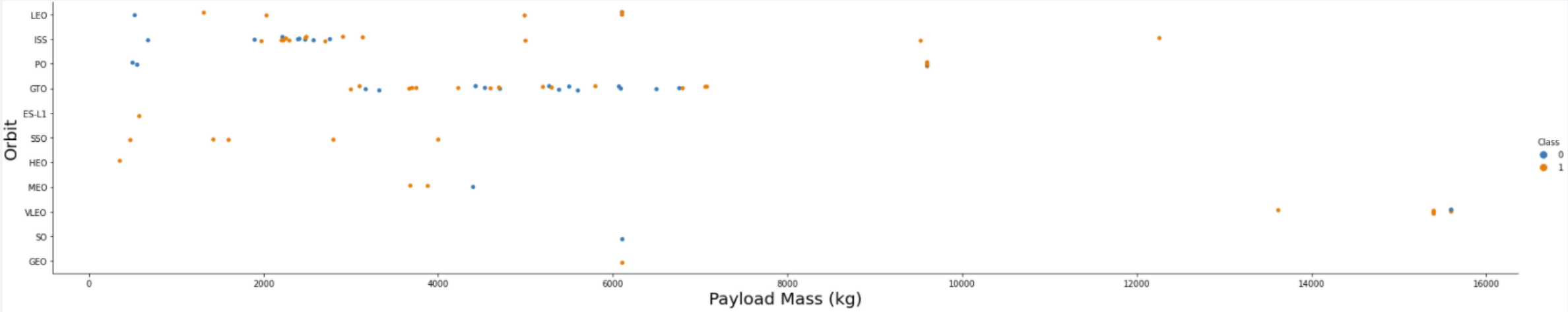
Scatter plot of Flight number vs. Orbit Type



- Scatter plot was used to determine if flight number affected the success rate of each orbit type (orange dots indicate success and blue dots indicate failure).
- There is a relationship between flight number and orbit in terms of success for LEO (low Earth orbit) and MEO (intermediate circular orbit).
- There does not seem to be a relationship for ISS, PO, GTO, and VLEO orbits.
- Multiple orbits only have one data point: HEO, SO, GEO, so relationship cannot be determined and the extreme success rates of these orbits should be taken with caution.

Payload vs. Orbit Type

Scatter plot of Payload Mass vs. Orbit Type

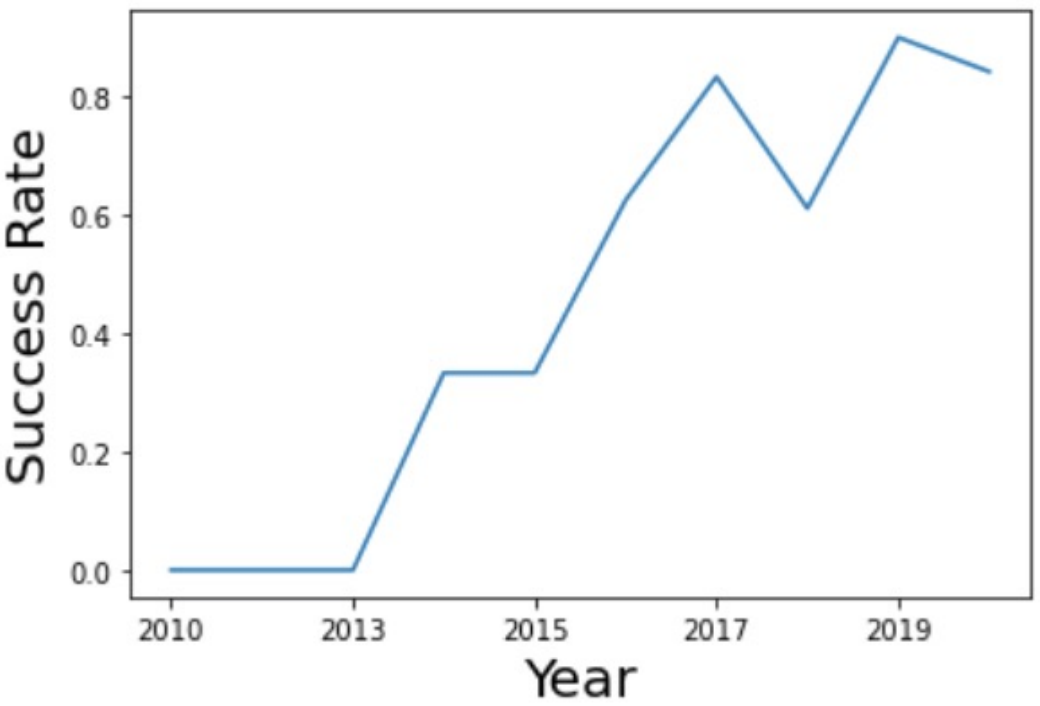


- Scatter plot was used to determine if payload mass affected the success rate of each orbit type (orange dots indicate success and blue dots indicate failure).
- LEO, ISS, and PO orbits have higher success rate as payload mass increases.
- MEO and VLEO orbits have higher success rate as payload mass decreases.
- SSO orbit has 100% success rate with payload below 4000kg.
- There does not seem to be a relationship for GTO orbit.

Launch Success Yearly Trend

- There are no success between 2010 and 2013.
- After 2013, we see a general upward success trend with slight drops in 2018 and 2020.

Line chart of yearly average success rate



All Launch Site Names

- Queried the unique launch sites in the imported CSV file.
- There are four unique launch sites in space mission: CCAFS LC-40, CCAFS SLC-40, KSC LC-39A, and VAFB SLC-4E.

```
%%sql
SELECT DISTINCT LAUNCH_SITE from SPACEXTBL
```

Query result:

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

Launch Site Names Begin with 'KSC'

- Queried 5 records where launch sites' names start with 'KSC'.
- Result is a 5x10 table since query was limited to show only 5 records.

```
%%sql
Select * from SPACEXTBL
    Where LAUNCH_SITE like 'KSC%'
    Limit 5;
```

Query result:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Total Payload Mass

- Queried the total payload mass in kilograms carried by boosters from NASA (CRS).
- Boosters from NASA (CRS) carried a total of 45,596kg of payload.

Query result:

customer	total_payload_mass
NASA (CRS)	45596

```
%%sql
Select CUSTOMER,SUM(PAYLOAD_MASS__KG_) as total_payload_mass
from SPACEXTBL
Where CUSTOMER like 'NASA (CRS)'
Group by CUSTOMER
```

Average Payload Mass by F9 v1.1

- Queried the average payload mass in kilograms carried by booster version F9 v1.1.
- Booster version F9 v1.1 carried an average of 2,928kg payload on its missions.

Query result:

booster_version	average_payload_mass
F9 v1.1	2928

```
%%sql
Select BOOSTER_VERSION,AVG(PAYLOAD_MASS__KG_) as average_payload_mass
from SPACEXTBL
Where BOOSTER_VERSION = 'F9 v1.1'
Group by BOOSTER_VERSION
```

First Successful Drone Ship Date

- Queried the date of the first successful landing outcome on drone ship.
- The first successful landing on a drone ship happened on April 8th, 2016.

Query result:

landing__outcome	first_drone_success
Success (drone ship)	2016-04-08

```
%%sql
Select LANDING__OUTCOME, MIN(DATE) as first_drone_success
from SPACEXTBL
WHERE LANDING__OUTCOME like 'Success (drone%'
group by LANDING__OUTCOME
```

Successful Ground Pad Landing with Payload between 4000 and 6000

- Queried the booster versions that have successfully landed on ground pad and had payload mass greater than 4,000kg but less than 6,000 kg.
- There are three booster versions that satisfied the conditions of this query:
  - F9 FT B1032.1
  - F9 B4 B1040.1
  - F9 B4 B1043.1

Query result:

booster_version	landing__outcome	payload_mass__kg_
F9 FT B1032.1	Success (ground pad)	5300
F9 B4 B1040.1	Success (ground pad)	4990
F9 B4 B1043.1	Success (ground pad)	5000

```
%%sql
Select BOOSTER_VERSION, LANDING__OUTCOME, PAYLOAD_MASS__KG_
from SPACEXTBL
where LANDING__OUTCOME = 'Success (ground pad)'
and PAYLOAD_MASS__KG_ between 4000 and 6000;
```

Total Number of Successful and Failure Mission Outcomes

- Queried the total number of successful and failed mission outcomes.
- There are a total of 101 missions:
  - Failed: 1 mission
  - Success: 99 missions
  - Success (with unknown payload status): 1 mission

Query result:

mission_outcome	total_number
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

```
%%sql
Select MISSION_OUTCOME, COUNT(MISSION_OUTCOME) as total_number
from SPACEXTBL
group by MISSION_OUTCOME
```

Boosters Carried Maximum Payload

- Queried the names of the booster which have carried the maximum payload mass
- The maximum payload is 15,600kg.
- There are 12 boosters that carried this amount of payload, and all are F9 B5 boosters.

```
%%sql
SELECT BOOSTER_VERSION, PAYLOAD_MASS__KG_ from SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

Query result:

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

2017 Launch Records

- Queried the successful landing outcomes in 2017 and displayed only the names of the month, landing outcome, booster version, and launch site.
- Successful landing outcomes happened in six months of 2017: February, May, June, August, September, and December.

Query result:

MONTH	landing__outcome	booster_version	launch_site
2	Success (ground pad)	F9 FT B1031.1	KSC LC-39A
5	Success (ground pad)	F9 FT B1032.1	KSC LC-39A
6	Success (ground pad)	F9 FT B1035.1	KSC LC-39A
8	Success (ground pad)	F9 B4 B1039.1	KSC LC-39A
9	Success (ground pad)	F9 B4 B1040.1	KSC LC-39A
12	Success (ground pad)	F9 FT B1035.2	CCAFS SLC-40

```
%%sql
SELECT EXTRACT(MONTH from DATE) as MONTH, LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
from SPACEXTBL
where EXTRACT(YEAR from DATE) = 2017
and LANDING__OUTCOME = 'Success (ground pad)';
```

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

- Queried the number of successful landing outcomes between the date 2010-06-04 and 2017-03-20 and rank them in descending order.
- There are only 8 successful landings between the specified time period.
  - Drone ship received more success than ground pad landing.

Query result:

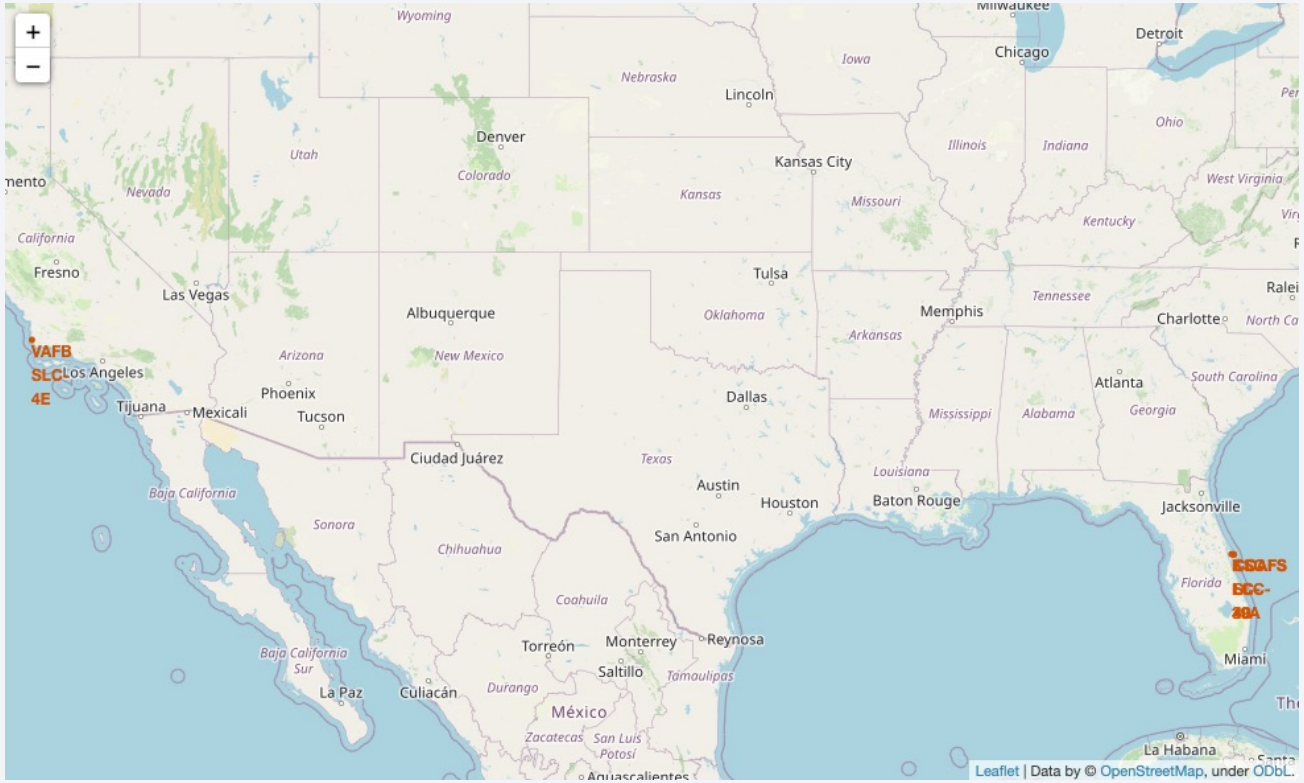
landing__outcome	counts_of_success
Success (drone ship)	5
Success (ground pad)	3

```
%%sql
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) as counts_of_success from SPACEXTBL
where DATE between '2010-06-04' and '2017-03-20'
and LANDING__OUTCOME like 'Success%'
Group by LANDING__OUTCOME
ORDER BY COUNT(LANDING__OUTCOME) DESC;
```


Section 3
Launch Sites
Proximities Analysis

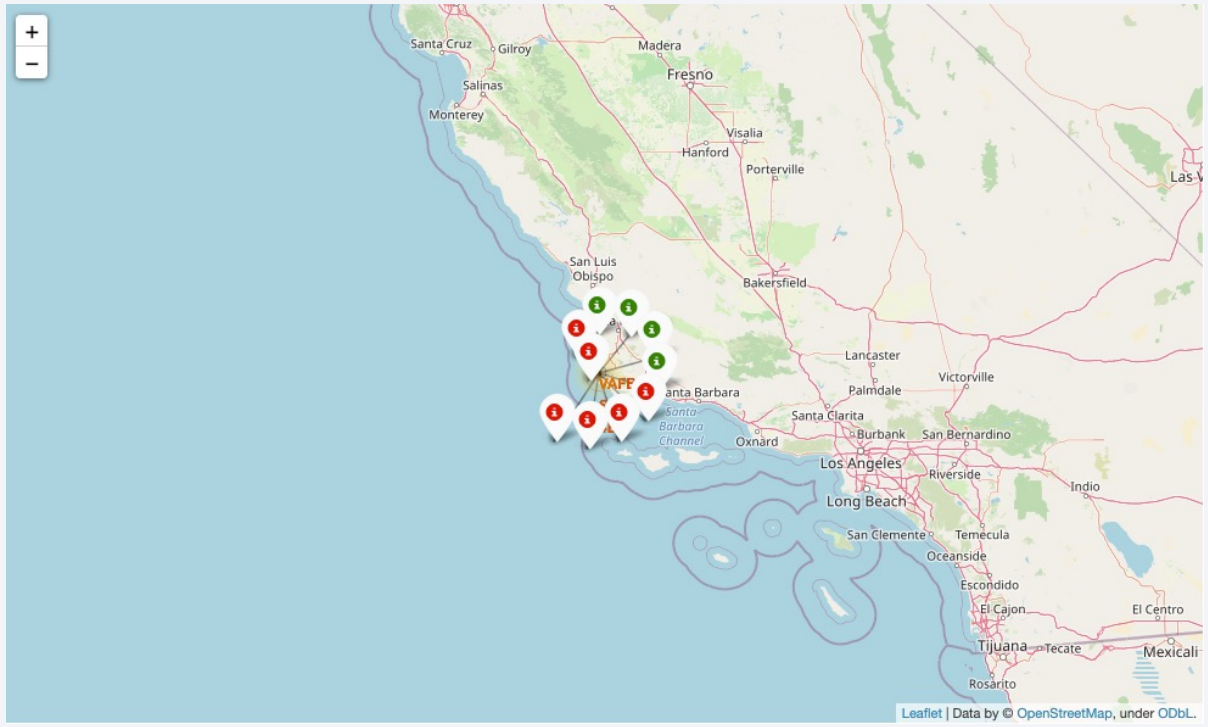
Folium Map: Launch Sites Locations

- This folium map shows the general distribution of the launch sites used in Falcon9 launches.
- There are total of four launch sites.
  - One is on the west coast.
  - Three are on the east coast.



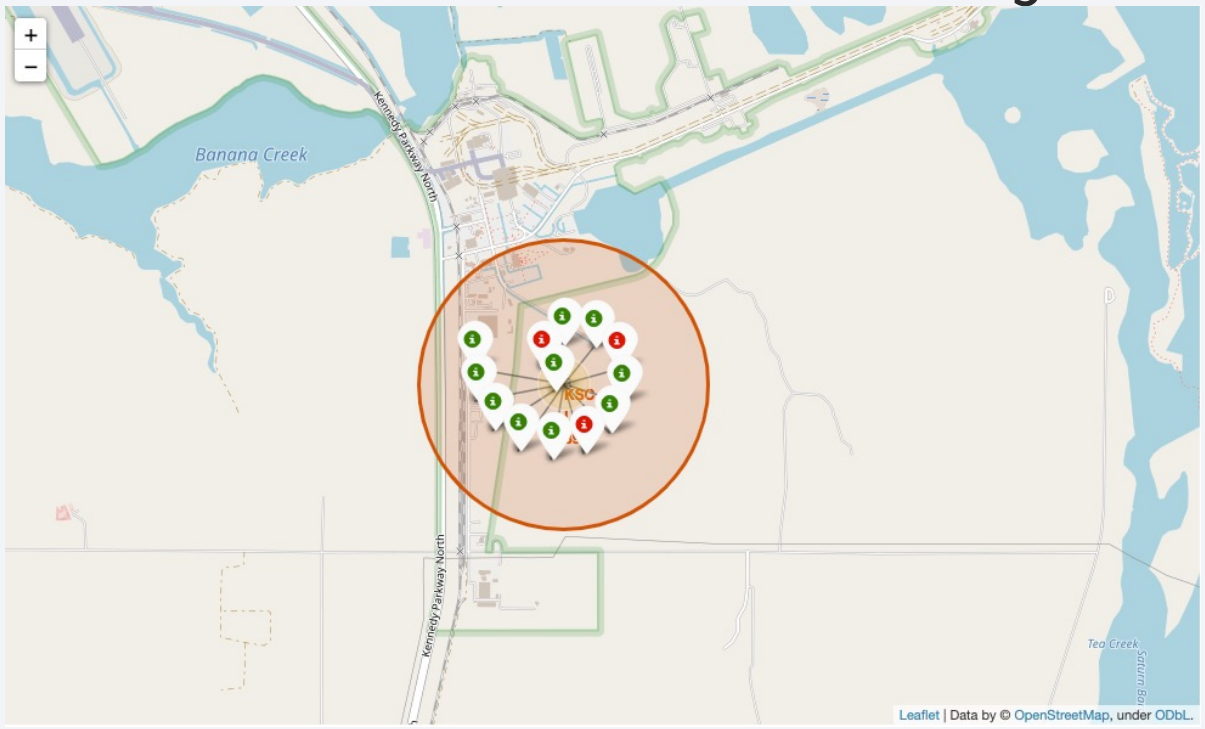
Folium Map: Success Rate of Launch Sites

- Success launches are labeled green and failed launched are labeled red on the folium map.
- Of the 10 launch attempts at VAFB SLC-4E launch site, only 4 launches were successful.



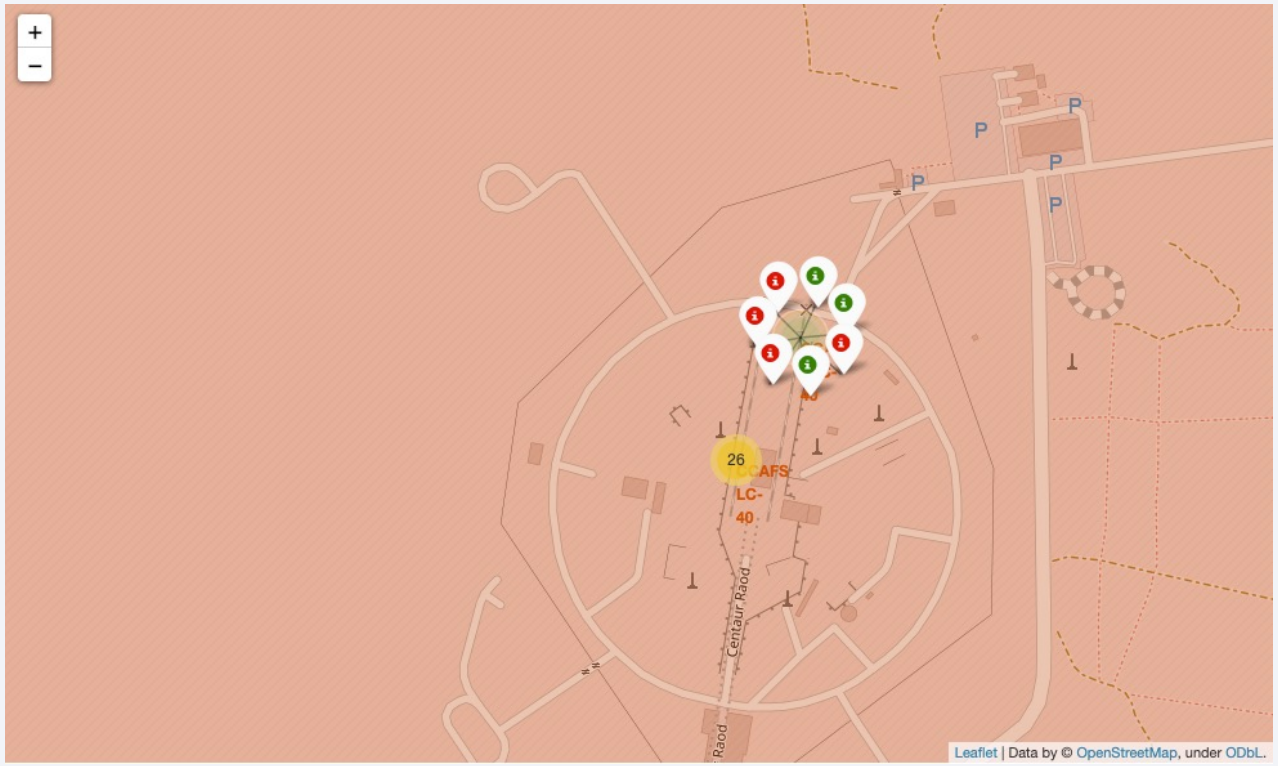
Folium Map: Success Rate of Launch Sites

- Success launches are labeled green and failed launched are labeled red on the folium map.
- Of the 13 launch attempts at KSC LC-39A launch site, only 3 launches failed and all other missions succeed. This launch site has the highest success rate.



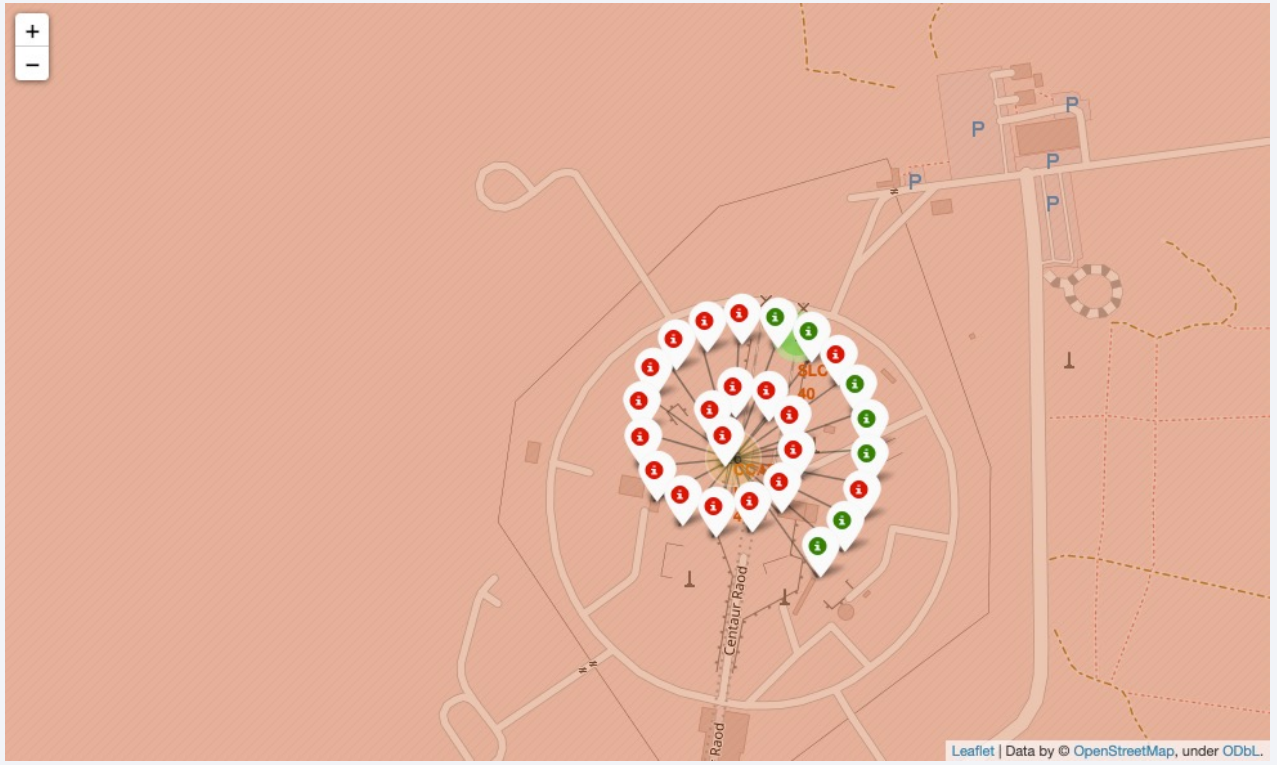
Folium Map: Success Rate of Launch Sites

- Success launches are labeled green and failed launched are labeled red on the folium map.
- Of the 7 launch attempts at CCAFS SLC-40 launch site, only 3 launches were successful.



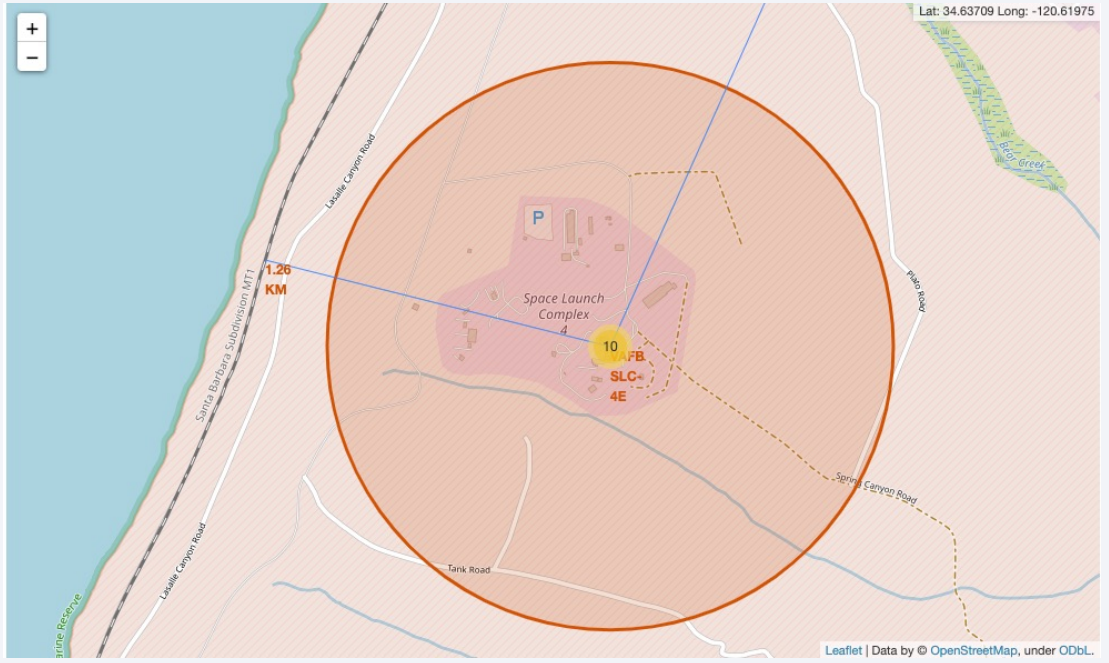
Folium Map: Success Rate of Launch Sites

- Success launches are labeled green and failed launched are labeled red on the folium map.
- Of the 26 launch attempts at CCAFS LC-40 launch site, only 7 launches were successful.



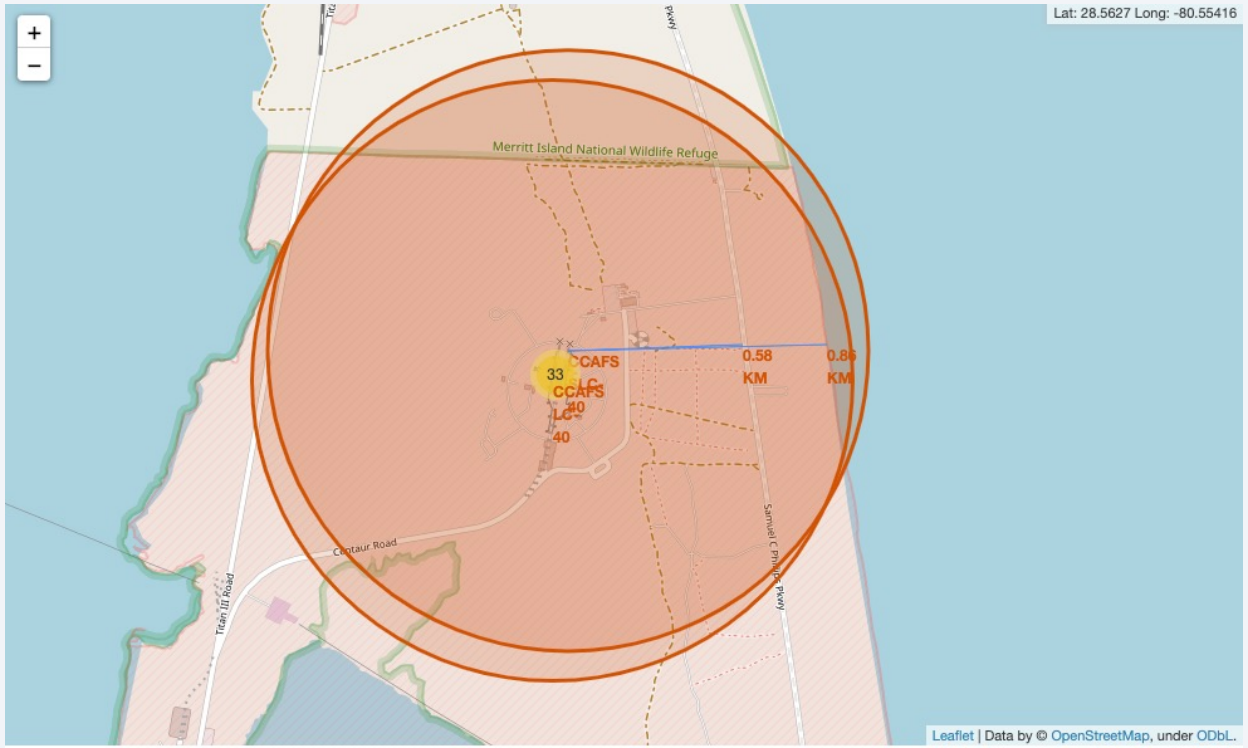
Folium Map: Closest Railway is on West Coast

- Distances between launch sites and closest railway, highway, and coastline are calculated and plotted on the Folium map.
- The shortest distance between a launch site and a railway is between VAFB SLC-4E and the Santa Barbara Subdivision MT1 (1.26km) on the west coast.



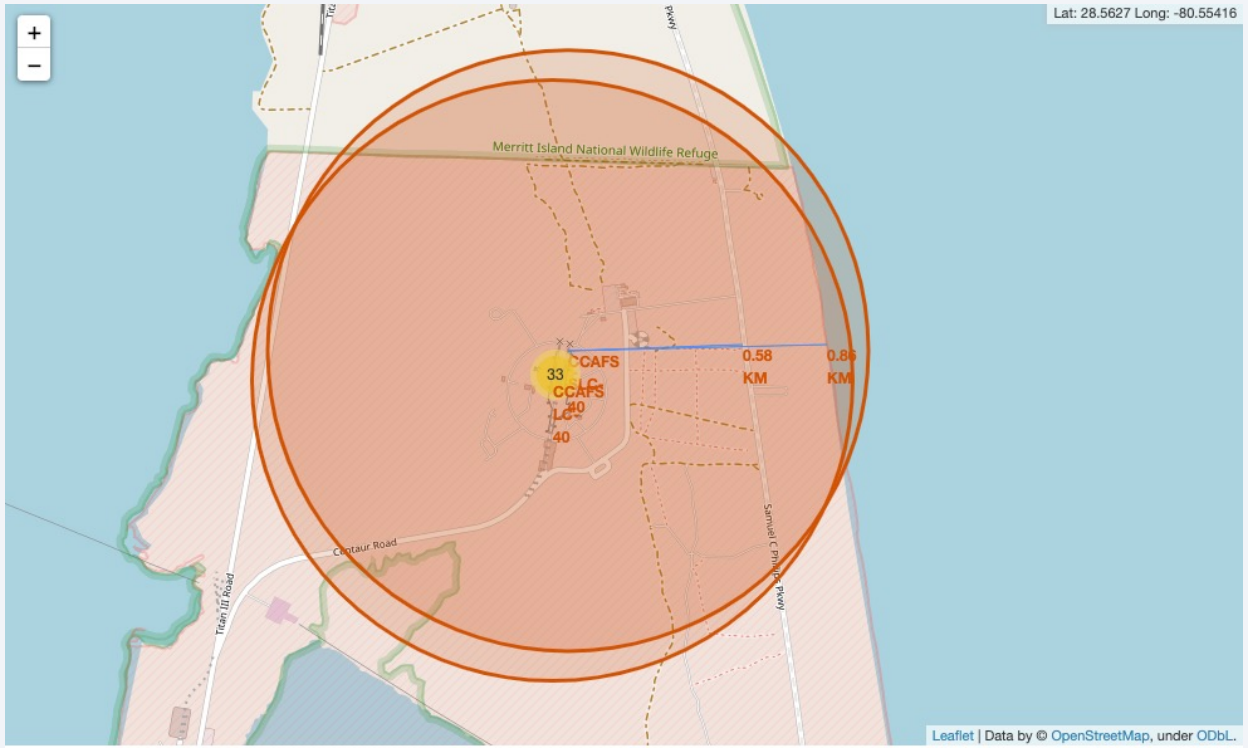
Folium Map: Closest Highway is on East Coast

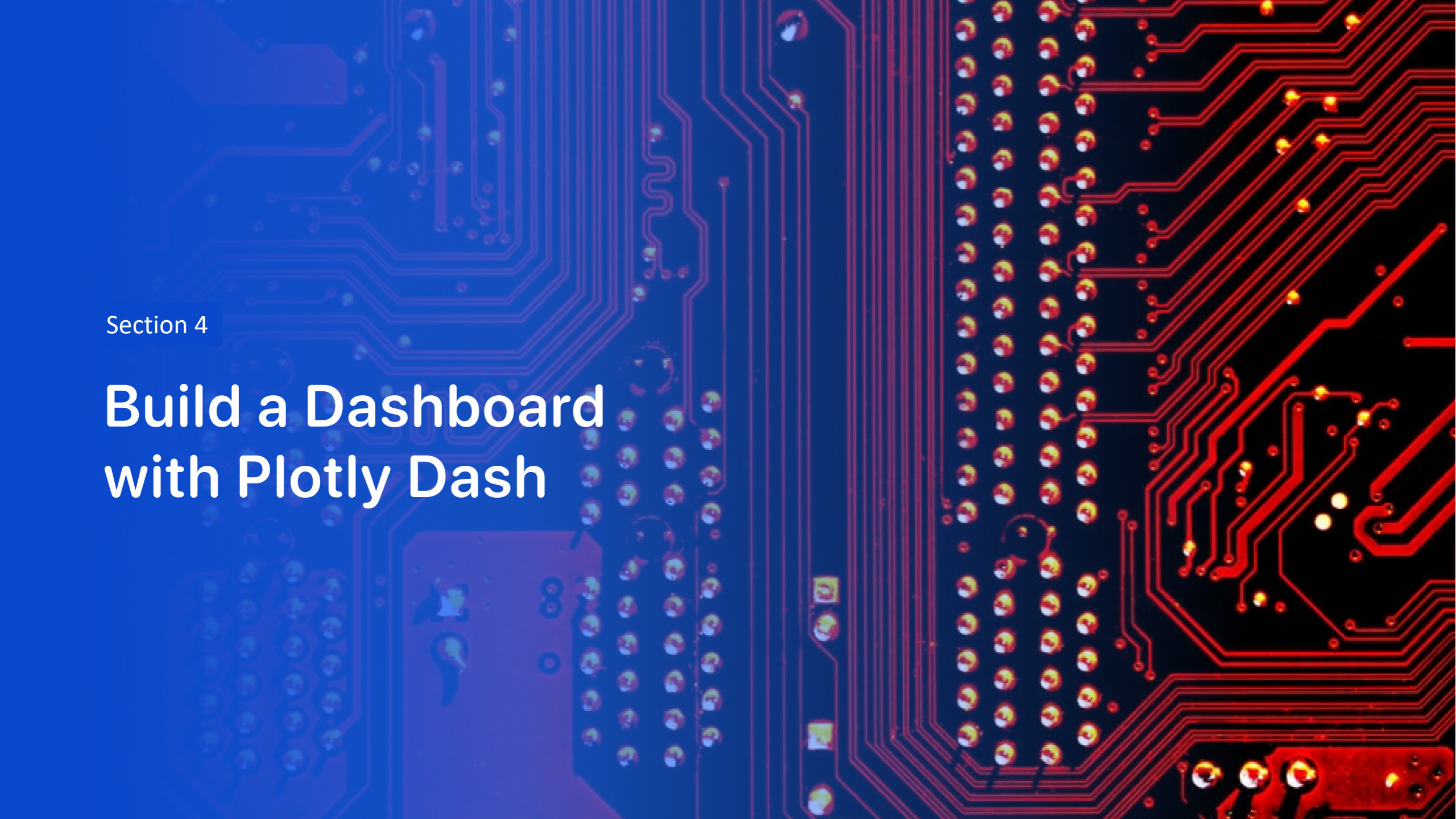
- Distances between launch sites and closest railway, highway, and coastline are calculated and plotted on the Folium map.
- The shortest distance between a launch site and a highway is between CCAFS SLC-40 and the Samuel C Philips Parkway (0.58km) on the east coast.



Folium Map: Closest Coastline is on East Coast

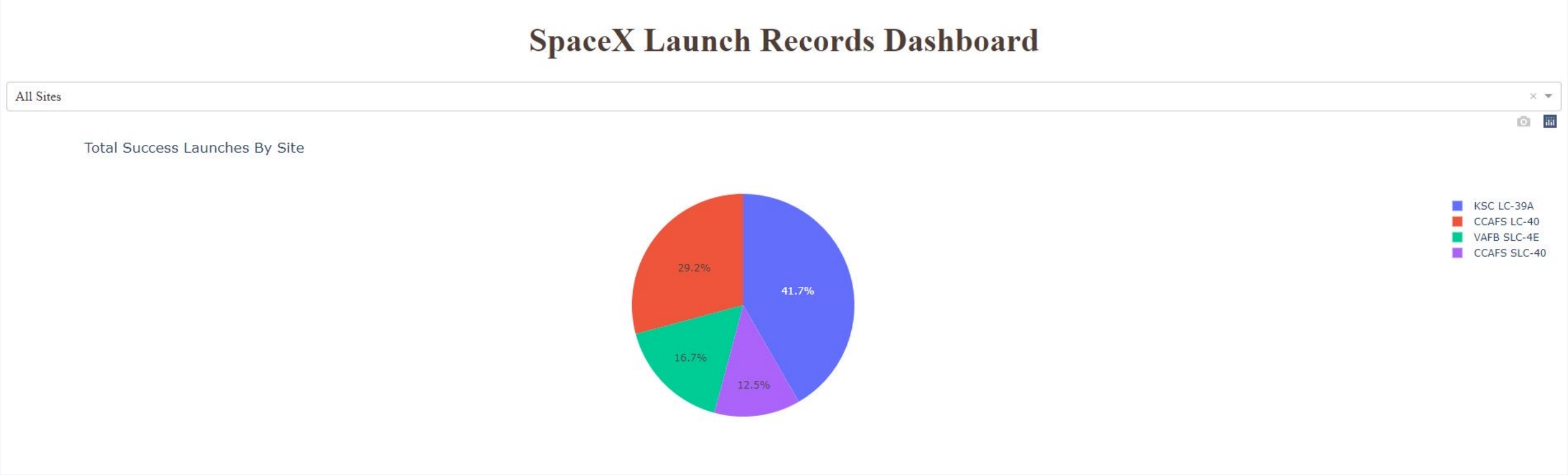
- Distances between launch sites and closest railway, highway, and coastline are calculated and plotted on the Folium map.
- The shortest distance between a launch site and the coastline is between CCAFS SLC-40 and the eastern coastline (0.86km).




Section 4
Build a Dashboard with Plotly Dash

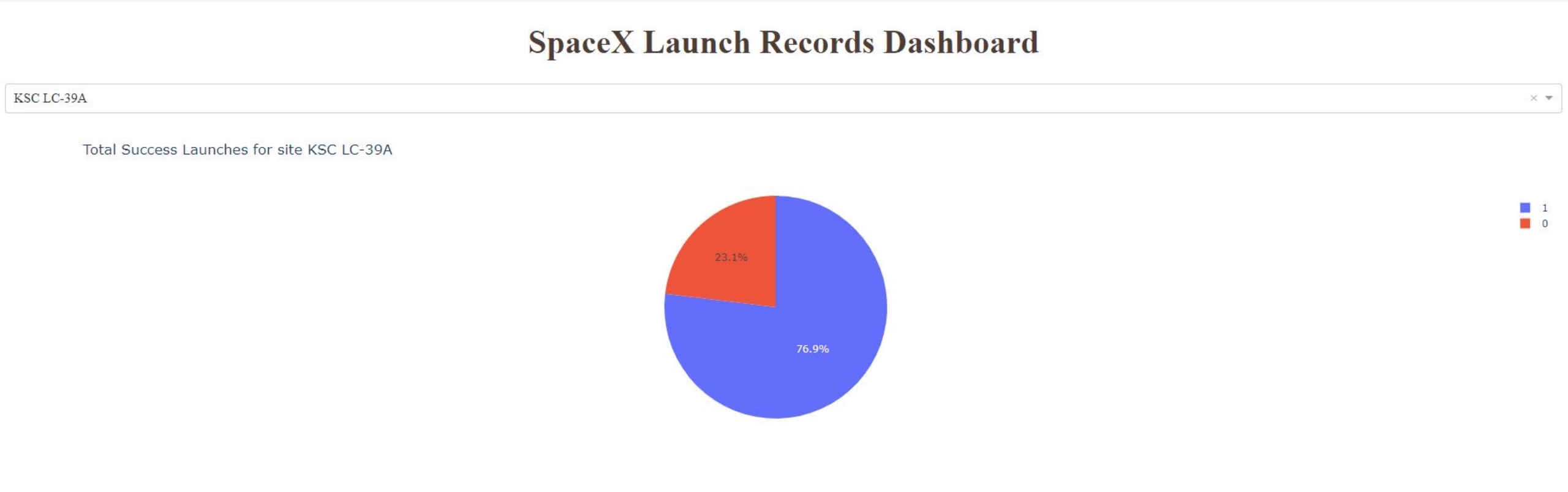
Dashboard: Pie Chart of Success Rate of All Sites

- Dashboard shows how much each launch site contributed to the total success rate.
- KSC LC-39A has the highest contribution (41.7%).
- CCAFS SLC-40 has the lowest contribution (12.5%).



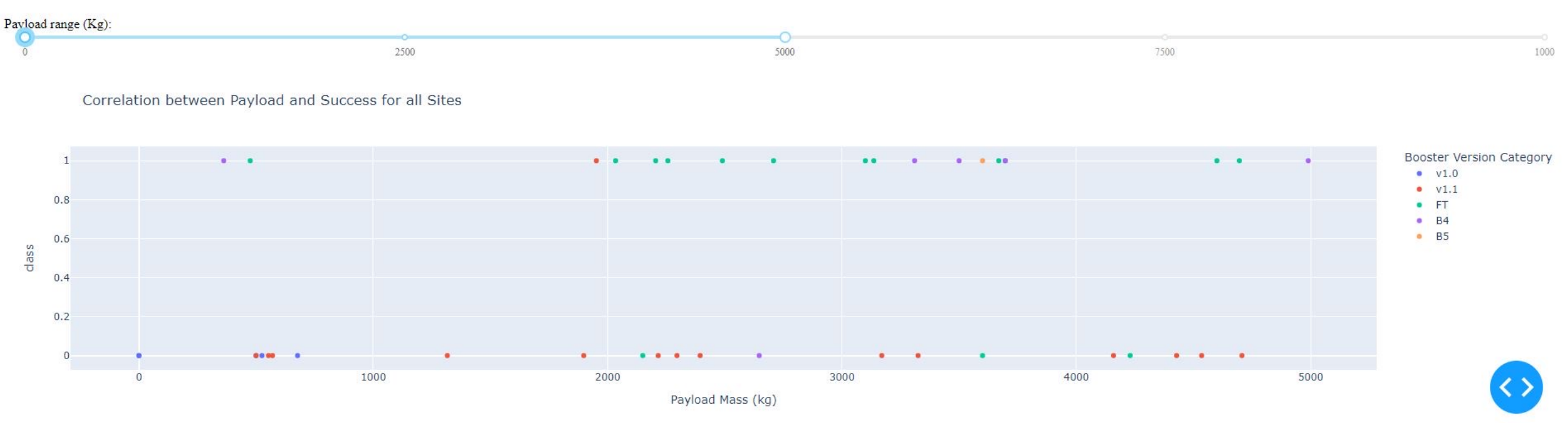
Dashboard: Success rate of KSC LC-39A

- KSC LC-39A has the highest success rate (76.9%).
- KSC LC-39A only has FT, B4, and B5 booster versions and light payload (not shown on screenshot), which may contribute to its high success rate.



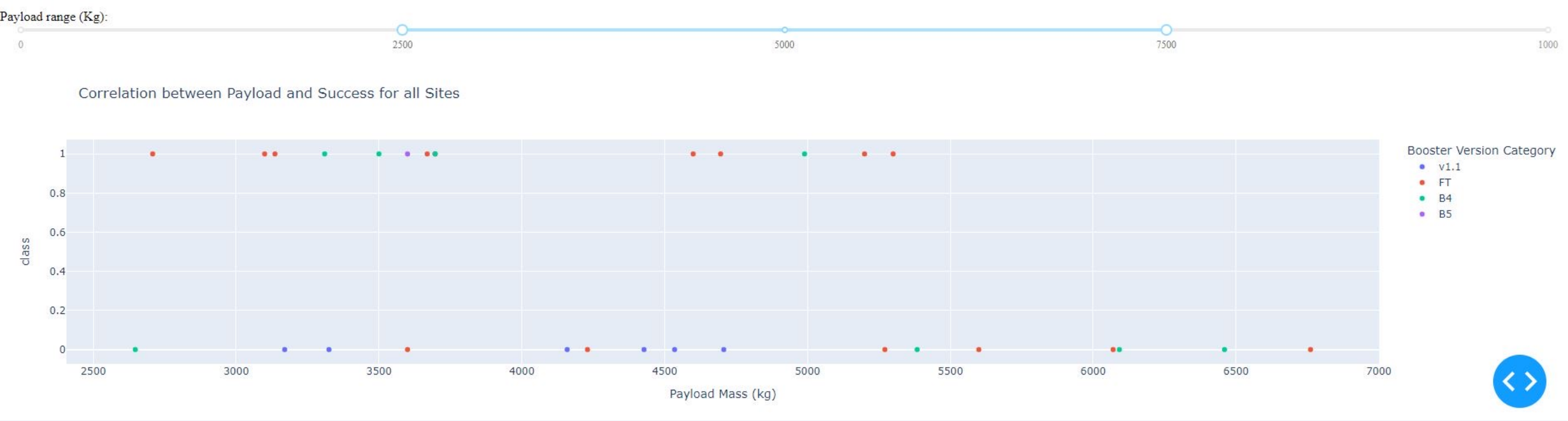
Dashboard: Payload vs Launch Outcome for All Sites

- Evaluating payload range between 0 - 5,000 kg.
- Booster version FT has the highest success rate, whereas booster version v1.1 has the lowest success rate.



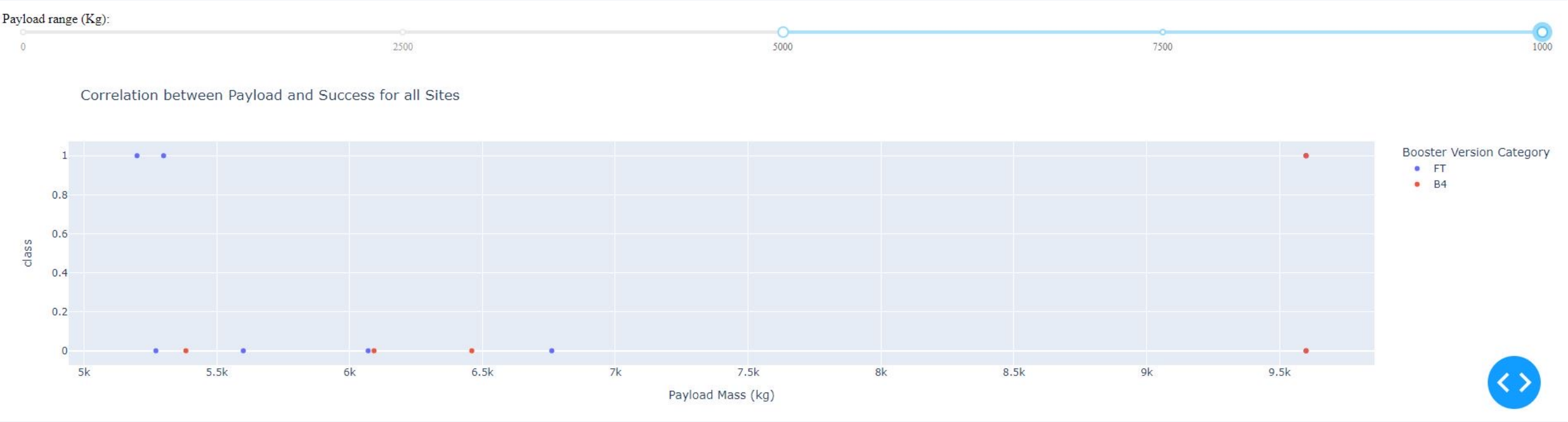
Dashboard: Payload vs Launch Outcome for All Sites

- Evaluating payload range between 2,500 – 7,500 kg.
- The success rate of booster B4 is highly dependent on payload weight.
- The relationship between success rate of booster v1.1 and its payload weight cannot be determined.



Dashboard: Payload vs Launch Outcome for All Sites

- Evaluating payload range between 5,000 - 10,000 kg.
- Only booster version FT and B4 were used with heavy payloads.
- Both booster versions do not have high success rate and no clear relationship between success rate and payload is shown.

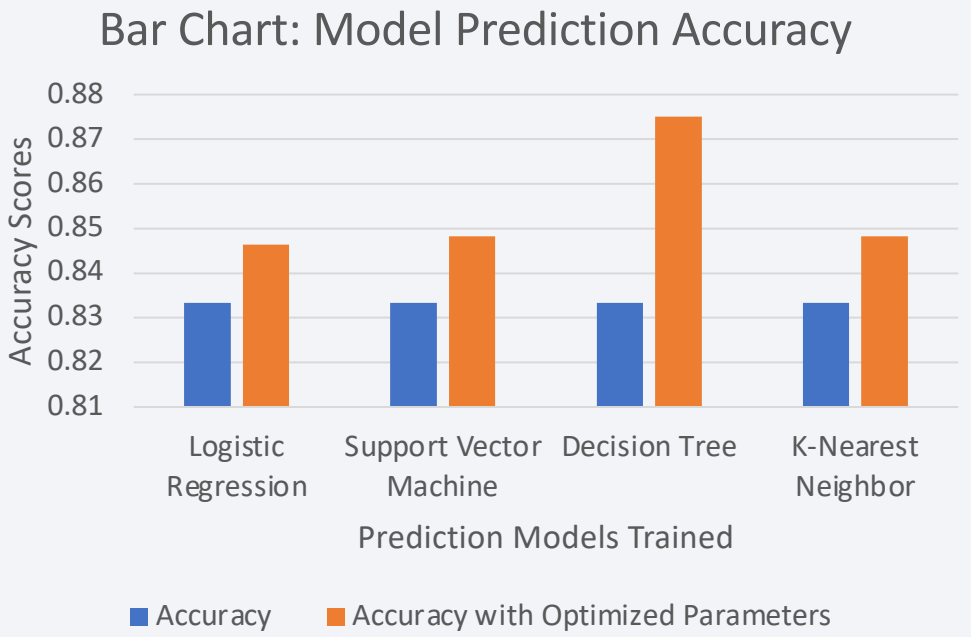


Section 5

Predictive Analysis (Classification)

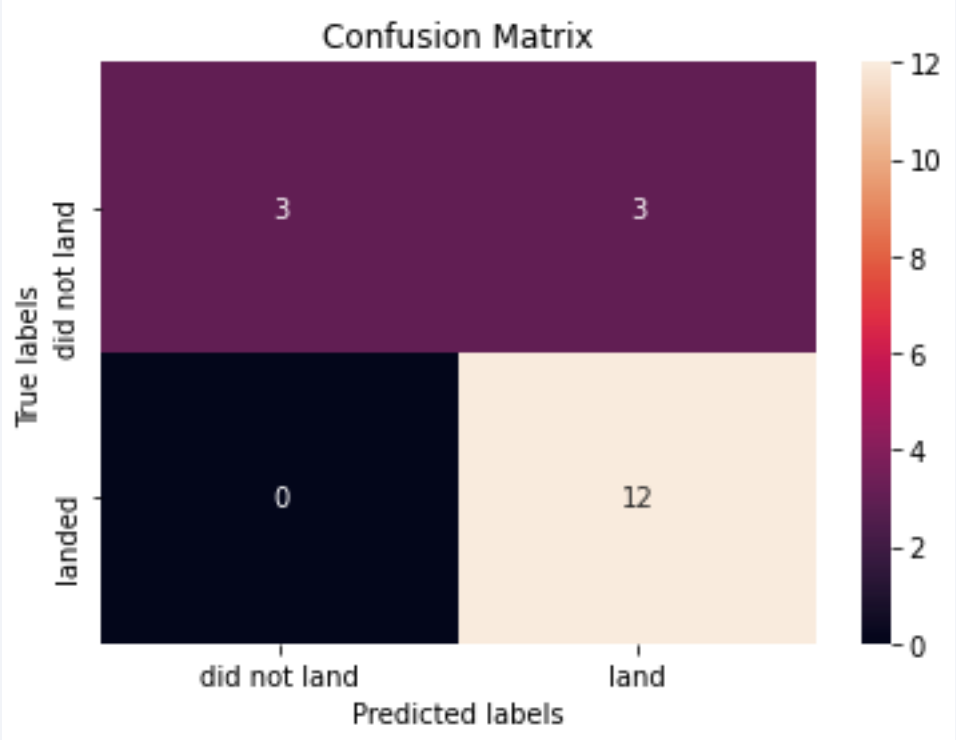
Classification Accuracy

- The accuracy score for all four models are the same: 0.833.
- After optimizing the models with the best parameters, the best accuracy score was calculated.
- Decision tree model has the highest accuracy 0.875 while logistic regression model has lowest accuracy 0.846.



Confusion Matrix

- Confusion matrix shows that there are no false negative predictions while there are 3 false positive predictions.
  - False negative: Predicting rocket will not land while it landed.
  - False positive: Predicting rocket will land while it did not land.
- Shows that the model is good at predicting successful outcomes, but not very good at predicting failed outcomes.



Conclusions

- Analysis has shown that SpaceX Falcon 9 rockets have become increasingly successful in landing after delivering their payloads.
- However, multiple factors, including orbit type, payload mass, and launch site, needs to be considered when making decision.
  - KSC LC 39A launch site has the most success.
- Decision tree model is the best prediction model to use.
  - Model is generally good at predicting successful mission outcomes.
  - Parameters to be used are optimized and included in Appendix B.

Appendix A1 Folium map distance calculations

- Folium Map distance calculations: To closest coastline
  - Calculated the distance of each launch site to the closest coastline (East coast)
  - Determined that CCAFS SLC-40 is the closest to the eastern coastline

In [16]:
```
# find coordinate of the closet coastline
# e.g.,: Lat: 28.56367  Lon: -80.57163
launch_site_lat = list(launch_sites_df['Lat'])
launch_site_long = list(launch_sites_df['Long'])
launch_site_label = list(launch_sites_df['Launch Site'])
coast_lat = 28.56338
coast_long = -80.56799
distance_coastline = {}
for lat, long, label in zip(launch_site_lat, launch_site_long, launch_site_label):
    distance_coastline[label] = calculate_distance(lat, long, coast_lat, coast_long)
distance_coastline
```

Out[16]:
```
{'CCAFS LC-40': 0.9228567649536887,
 'CCAFS SLC-40': 0.8628630313083624,
 'KSC LC-39A': 7.785907141082712,
 'VAFB SLC-4E': 3827.85635744151}
```

Appendix A2 Folium map distance calculations

- Folium Map distance calculations: To closest railroad
  - Calculated the distance of each launch site to the closest railroad
  - Determined the shortest distance to be between the Santa Barbara Subdivision MT1 Railroad and the launch site 'VAFB SLC-4E'

In [21]:
```
# To the closest railway on east coast- NASA Railroad
NASA_rail_lat = 28.57205
NASA_rail_long = -80.58526
distance_railway = {}
for lat, long, label in zip(launch_site_lat, launch_site_long, launch_site_label):
    distance_railway[label] = calculate_distance(lat, long, NASA_rail_lat, NASA_rail_long)
print("Distance from launch sites to NASA Railroad: ", distance_railway)

# To the closest railway on west coast- Santa Barbara Subdivision MT1
SB_rail_lat = 34.63558
SB_rail_long = -120.62407
distance_railway = {}
for lat, long, label in zip(launch_site_lat, launch_site_long, launch_site_label):
    distance_railway[label] = calculate_distance(lat, long, SB_rail_lat, SB_rail_long)
print("Distance from launch sites to Santa Barbara Subdivision MT1 Railroad: ", distance_railway)
```

```
Distance from launch sites to NASA Railroad:  {'CCAFS LC-40': 1.3310737253149305, 'CCAFS SLC-40': 1.284290868365806,
'KSC LC-39A': 6.02222924348479, 'VAFB SLC-4E': 3825.9384107022897}
Distance from launch sites to Santa Barbara Subdivision MT1 Railroad:  {'CCAFS LC-40': 3828.2544390612934, 'CCAFS SLC
-40': 3828.268635959349, 'KSC LC-39A': 3821.4649364827824, 'VAFB SLC-4E': 1.257115450405437}
```

Appendix A3 Folium map distance calculations

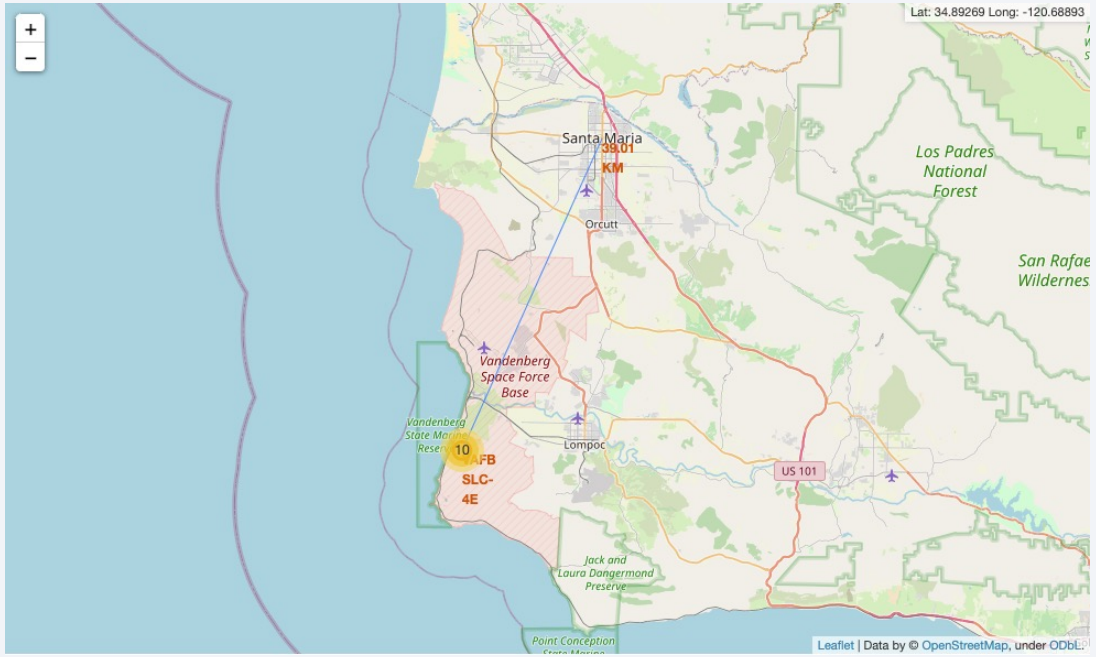
- Folium Map distance calculations: To closest highway
  - Calculated the distance of each launch site to the closest highway
  - Determined the shortest distance is between the Samuel C Phillips Pkwy and launch site 'CCAFS SLC-40'

In [23]:
```
# There are no close highway on the west coast, but two potential highways for east coast

# To the Kennedy Parkway North highway
KPN_rail_lat = 28.57331
KPN_rail_long = -80.65551
distance_railway = {}
for lat, long, label in zip(launch_site_lat, launch_site_long, launch_site_label):
    distance_railway[label] = calculate_distance(lat, long, KPN_rail_lat, KPN_rail_long)
print("Distance from launch sites to Kennedy Parkway North highway: ", distance_railway)

# To the Samuel C Phillips Pkwy
SCPP_rail_lat = 28.56336
SCPP_rail_long = -80.57086
distance_railway = {}
for lat, long, label in zip(launch_site_lat, launch_site_long, launch_site_label):
    distance_railway[label] = calculate_distance(lat, long, SCPP_rail_lat, SCPP_rail_long)
print("Distance from launch sites to Samuel C Phillips Pkwy: ", distance_railway)
```

```
Distance from launch sites to Kennedy Parkway North highway:  {'CCAFS LC-40': 7.732207109796197, 'CCAFS SLC-40': 7.76
8881282735071, 'KSC LC-39A': 0.8415309118826659, 'VAFB SLC-4E': 3819.4612469396093}
Distance from launch sites to Samuel C Phillips Pkwy:  {'CCAFS LC-40': 0.6454753321668273, 'CCAFS SLC-40': 0.58252899
95976455, 'KSC LC-39A': 7.5087788037592444, 'VAFB SLC-4E': 3827.594523350465}
```

Appendix A4 Folium map distance calculations

- Folium Map distance calculations: To closest cities
  - Calculated the distance of each launch site to the closest cities on both West and East coast
  - Determined the shortest distance is between Santa Maria and launch site ‘VAFB SLC-4E’



Appendix A4 Folium map distance calculations

- Folium Map distance calculations: To closest cities
  - Launch sites on east coast are all at least 50km away from cities.
  - Launch site on west coast is the closest to the city (39km).

In [19]:
```
# Create a marker with distance to a closest city, railway, highway, etc.
# Draw a line between the marker to the launch site
# To the closest city on east coast- Melbourne
Mel_lat = 28.10651
Mel_long = -80.6369
distance_city = {}
for lat, long, label in zip(launch_site_lat, launch_site_long, launch_site_label):
    distance_city[label] = calculate_distance(lat, long, Mel_lat, Mel_long)
print("Distance from launch sites to Melbourne, FL: ", distance_city)

# To the closest city on west coast- Santa Maria
SM_lat = 34.95279
SM_long = -120.43582
distance_city = {}
for lat, long, label in zip(launch_site_lat, launch_site_long, launch_site_label):
    distance_city[label] = calculate_distance(lat, long, SM_lat, SM_long)
print("Distance from launch sites to Santa Maria, CA: ", distance_city)
```

```
Distance from launch sites to Melbourne, FL:  {'CCAFS LC-40': 51.031718282618634, 'CCAFS SLC-40': 51.13665333957838,
'KSC LC-39A': 51.92514155956699, 'VAFB SLC-4E': 3839.6080684727576}
Distance from launch sites to Santa Maria, CA:  {'CCAFS LC-40': 3810.5336778623614, 'CCAFS SLC-40': 3810.546786626534
4, 'KSC LC-39A': 3803.756570697757, 'VAFB SLC-4E': 39.01102181521817}
```

Appendix B1 Prediction Analysis: Task1 & Task2

- Screenshot of input code and output for TASK 1 and TASK2.

TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` then assign it to the variable `Y`,make sure the output is a Pandas series (only one bracket df['name of column']).

```
In [5]: Y = data['Class'].to_numpy()
        Y
```

```
Out[5]: array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
               1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
               1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1])
```

TASK 2

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

```
In [6]: transform = preprocessing.StandardScaler()
```

```
In [7]: X = transform.fit(X).transform(X)
        X
```

```
Out[7]: array([[-1.71291154e+00, -1.94814463e-16, -6.53912840e-01, ...,
                -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
               [-1.67441914e+00, -1.19523159e+00, -6.53912840e-01, ...,
                -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
               [-1.63592675e+00, -1.16267307e+00, -6.53912840e-01, ...,
                -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
               ...,
               [ 1.63592675e+00,  1.99100483e+00,  3.49060516e+00, ...,
                 1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
               [ 1.67441914e+00,  1.99100483e+00,  1.00389436e+00, ...,
                 1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
               [ 1.71291154e+00, -5.19213966e-01, -6.53912840e-01, ...,
                -8.35531692e-01, -5.17306132e-01,  5.17306132e-01]])
```

Appendix B2 Prediction Analysis: TASK3

- Screenshot of input code and output for TASK 3.

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

```
In [8]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

```
In [9]: Y_test.shape
Out[9]: (18,)
```

Appendix B3 Prediction Analysis: TASK4 & Task5

- Screenshot of input code and output for TASK 4 and TASK 5.

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

In [10]:
```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']} # l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, param_grid = parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```

Out[10]:
```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

In [11]:
```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

TASK 5

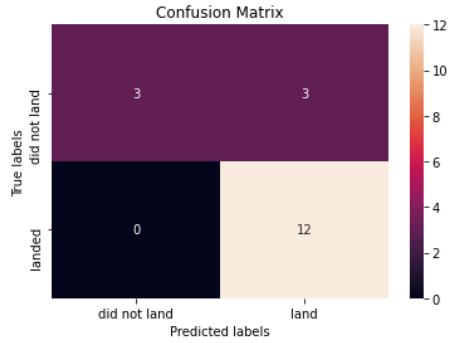
Calculate the accuracy on the test data using the method `score`:

In [12]:
```
print(logreg_cv.score(X_train, Y_train))
print(logreg_cv.score(X_test, Y_test))
```

```
0.875
0.8333333333333334
```

Lets look at the confusion matrix:

In [13]:
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Appendix B4 Prediction Analysis: TASK6 & Task7

- Screenshot of input code and output for TASK 6 and TASK 7.

TASK 6

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with cv - 10. Fit the object to find the best parameters from the dictionary `parameters`.

In [14]:
```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

In [15]:
```
svm_cv = GridSearchCV(svm, param_grid=parameters, cv=10)
svm_cv.fit(X_train, Y_train)
```

Out[15]:
```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
```

In [16]:
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

TASK 7

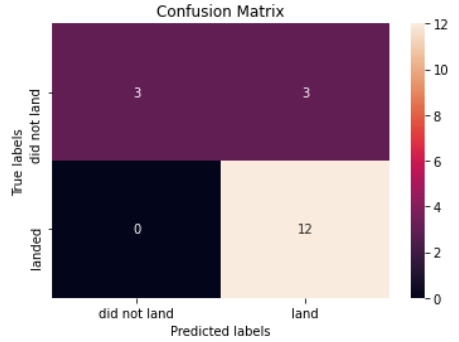
Calculate the accuracy on the test data using the method `score`:

In [17]:
```
print(svm_cv.score(X_train, Y_train))
print(svm_cv.score(X_test, Y_test))
```

```
0.8888888888888888
0.8333333333333334
```

We can plot the confusion matrix

In [18]:
```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Appendix B5 Prediction Analysis: TASK8 & Task9

- Screenshot of input code and output for TASK 8 and TASK 9.

TASK 8

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

In [19]:
```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

In [20]:
```
tree_cv = GridSearchCV(tree, param_grid=parameters, cv=10)
tree_cv.fit(X_train, Y_train)
```

Out[20]:
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

In [21]:
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'best'}
accuracy : 0.875
```

TASK 9

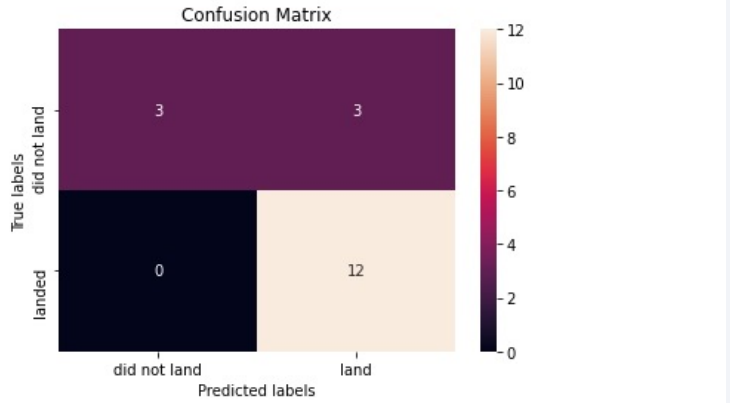
Calculate the accuracy of tree_cv on the test data using the method `score`:

In [22]:
```
tree_cv.score(X_test, Y_test)
```

Out[22]: 0.8333333333333334

We can plot the confusion matrix

In [23]:
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Appendix B6 Prediction Analysis: TASK10 & Task11

- Screenshot of input code and output for TASK 10 and TASK 11.

TASK 10

Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

In [24]:
```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

In [25]:
```
knn_cv = GridSearchCV(KNN, param_grid=parameters, cv=10)
knn_cv.fit(X_train, Y_train)
```

Out[25]:
```
GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'p': [1, 2]})
```

In [26]:
```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

TASK 11

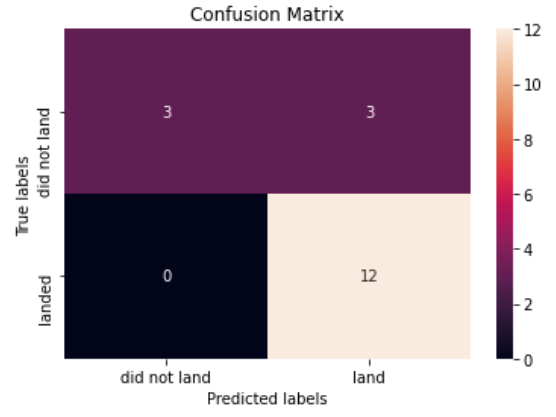
Calculate the accuracy of tree_cv on the test data using the method `score`:

In [27]:
```
knn_cv.score(X_test, Y_test)
```

Out[27]:
```
0.8333333333333334
```

We can plot the confusion matrix

In [28]:
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Thank you!