IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Omkar Suresh Kadam

December 3, 2023

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data collection utilizing REST API and web scraping (beautiful soup).
  - Data wrangling ensuring consistency via cleaning and transformation.
  - Exploratory data analysis leveraging SQL for structured analysis and visualizations to uncover trends.
  - Interactive visualizations aided by folium and plotly.
  - Predictive analysis employing machine learning algorithms for accurate prediction of landing success.

Executive Summary

- **Summary of all results**
  - Outputs Exploratory Data Analysis
  - Dynamic Visualizations (Screenshots)
  - Results of Machine Learning models for predictive analysis

Introduction

- **Project background and context**

  SpaceX promotes Falcon 9 rocket launches on its website at a significantly lower cost of 62 million dollars each, in stark contrast to other providers whose prices soar above 165 million dollars per launch. The substantial cost savings stem from SpaceX's innovative practice of reusing the first stage of the launch. Consequently, the primary objective of this project is to forecast the landing outcome of Falcon 9's first stage in future missions. This prediction not only aids in understanding the success of landings but also provides insights into the potential cost of a launch. Such information becomes invaluable for competing companies interested in bidding against SpaceX for upcoming rocket launches.

Introduction

- **Problems you want to find answers to.**

  Will Falcon 9's first stage be successful in landing?

  Which parameters influence the success of the landing?

  What are most optimal landing conditions?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - With the help of SpaceX REST API
  - Web Scrapping from Wikipedia using Python's BeautifulSoup package.
- Perform data wrangling
  - Removing null/empty values
  - Creating outcome categorical column – "Class"
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Plotting graphs (catplots, scatter plots)
  - One-Hot Encoding categorical features

Methodology

Executive Summary

- Perform interactive visual analytics using Folium and Plotly Dash
  - Displaying launch sites using Folium and their proximity to other landmarks.
  - Interactive Dashboarding with Plotly.
- Perform predictive analysis using classification models
  - Applying Logistic Regression, Support Vector Machine, Decision Tree, KNN
  - Employing GridSearchCV to find best parameters for maximum accuracy

Data Collection

- SpaceX REST API: https://api.spacexdata.com/v4/launchpads/
  - SpaceX REST API provides comprehensive data on launches, including details about the rocket, payload information, specifications for launch and landing, and the outcomes of the landing.
- Web Scrapping: https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922
  - Using Python's package - beautiful soup HTML tables from Wikipedia page were retrieved.

Data Collection – SpaceX API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"

response = requests.get(spacex_url)
```

REST API Response

```
# Use json_normalize meethod to convert the json result into a dataframe
json_data = response.json()
data = pd.json_normalize(json_data)
```

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

Converting Response -> JSON -> Dataframe

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

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

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

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

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

Cleaning the dataframe of Unnecessary Values

Data Collection – SpaceX API

Consolidated Dataframe

```
# Show the head of the dataframe
df.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0

Github URL: https://github.com/OmkarKadam2002/Applied-data-science-capstone-Coursera/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

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

```
response.status_code
```

Response of Wikipedia URL

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

Accessing Table using Beautiful Soup object

```
extracted_row = 0
#Extract each table

launch_data_list = []

for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict = {'Flight No.': flight_number}
            #print(flight_number)
```

Information Extraction

Github URL:
https://github.com/OmkarKadam2002/Applied-data-science-capstone-Coursera/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

- Mapping of outcomes such as True Ocean, False RLTS, etc. was done to 0 and 1 which represents booster not landing successfully, and booster landing successfully respectively.

```
# landing_outcomes = values on Outcome column
landing_outcomes=df['Outcome'].value_counts()
```

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

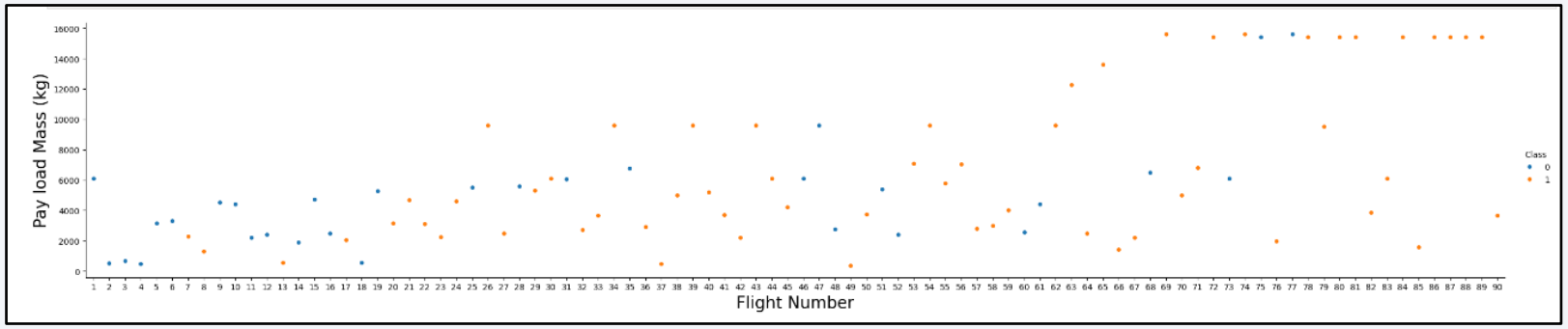
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for key,value in df["Outcome"].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

- Github URL: https://github.com/OmkarKadam2002/Applied-data-science-capstone-Coursera/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

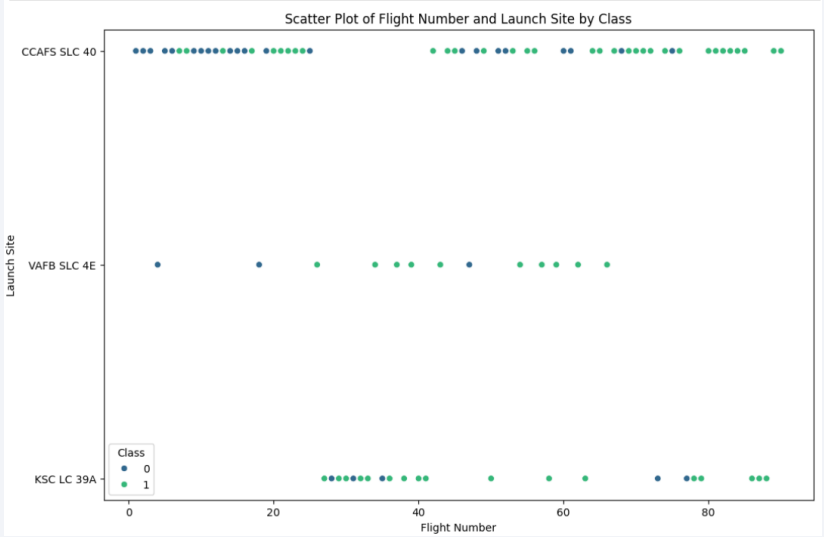
EDA with Data Visualization

- Catplot of Payload Mass against Flight Number:

Rate of Successful Landing increases with increase in Flight Number

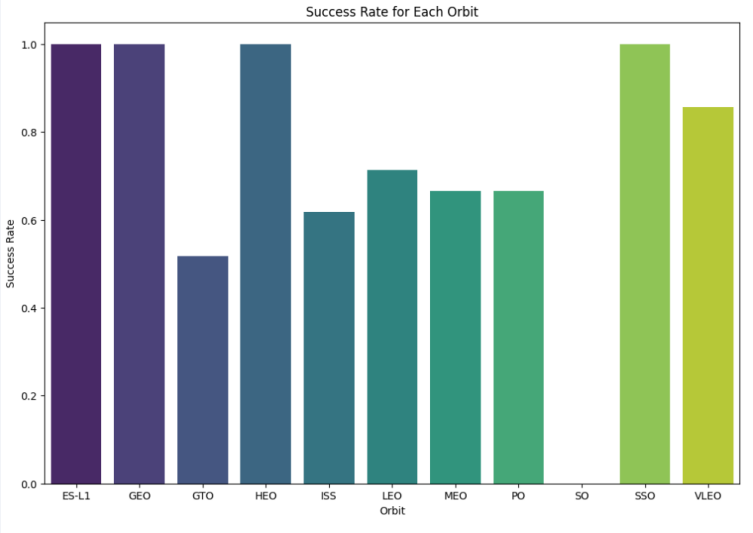


- Multiple scatter plots are able to delineate relations in between variables:
  - Launch Site and Payload Mass
  - Launch Site and Flight Number
  - Flight Number and Orbit Type
  - Payload and Orbit Type

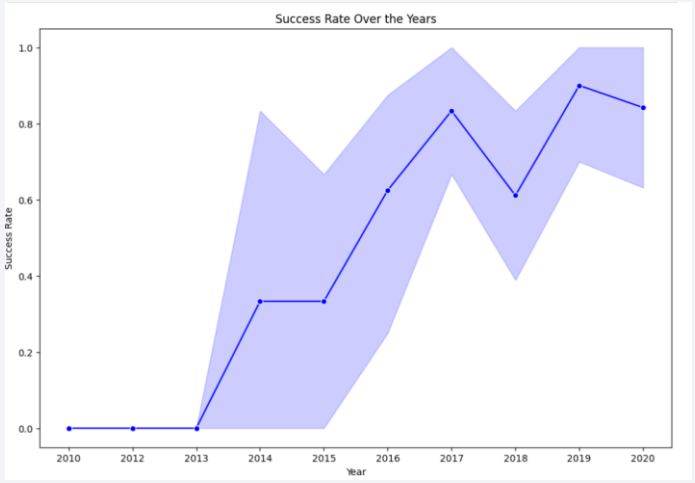


EDA with Data Visualization

- Bar Plot for Success Rate of Each Orbit. Tells which orbits have the highest probability of making the landing.



- Lineplot of success rate over the years – suggesting that success rate has gradually increased.



Github URL: https://github.com/OmkarKadam2002/Applied-data-science-capstone-Coursera/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- The SQL queries were executed on a SpaceX Falcon 9 dataset to gain insights into launch outcomes and booster performance. The initial query filtered records where the date was not null. Subsequent queries revealed unique launch sites, displayed specific records with launch sites starting with 'CCA,' calculated the total payload mass for NASA (CRS) launches, determined the average payload mass for booster version F9 v1.1, identified the date of the first successful ground pad landing, listed boosters with successful drone ship landings and specific payload masses, tallied successful and failed mission outcomes, found booster versions with the maximum payload mass, and ranked landing outcomes based on counts within a specified date range.

Github URL: https://github.com/OmkarKadam2002/Applied-data-science-capstone-Coursera/blob/main/jupyter-labs-eda-sql-coursera_sqllite_.ipynb

Build an Interactive Map with Folium

- We initialize a Folium map and add Circles around NASA John Space Centre (which acts as the centre to our map). Also, we add red circles around the launch sites which are present in the states of Florida and California. These Circles are named after the location they enclose.
- Then we create a markercluster – to group makers close to each other.
- The come the markers, which represent the launch site coordinates and are represented by colours Red(Unsuccessful Landing) and Green (Successful Landing).
- Another element being used is PolyLine, to represent the distance between a particular launch site and nearby structures (railroad, highway, city)

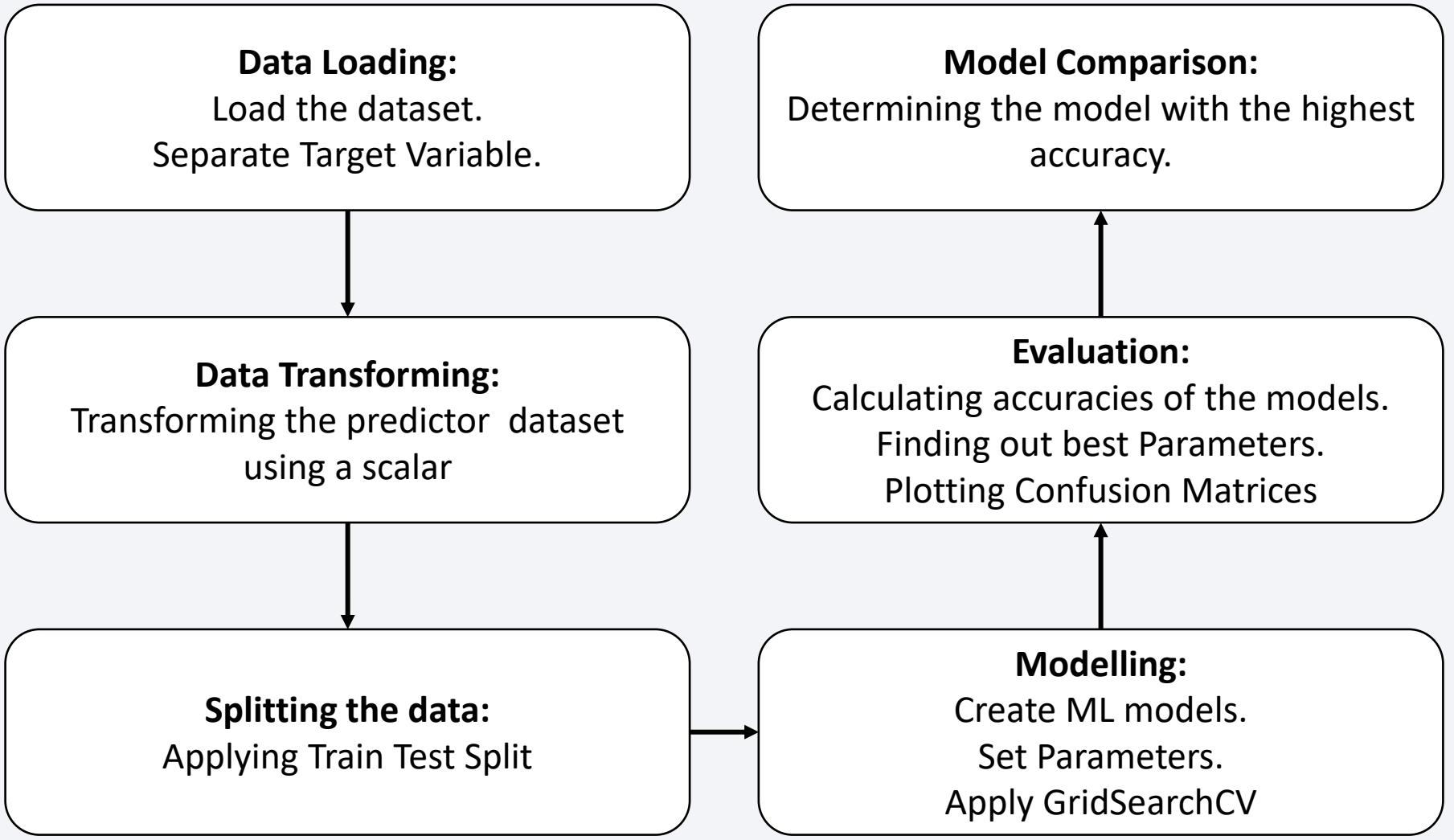
Github URL: https://github.com/OmkarKadam2002/Applied-data-science-capstone-Coursera/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Note: Images of Folium maps are not visible in the notebook. Use this link: https://github.com/OmkarKadam2002/Applied-data-science-capstone-Coursera/tree/main/images_of_folium_maps

Build a Dashboard with Plotly Dash

- Dashboarding has been done using Plotly. There's a dropdown menu allowing to select launch sites with default value as all sites. This dropdown interacts with the pie chart placed just below it. Which renders a pie chart containing pieces with values equal to individual number of launch sites records (when drop is at default value or All sites).
- In the case of the particular launch site being selected, the pie chart displayes outcomes of landing.
- A Rangeslider at the bottom allows to control the payload mass which is plotted underneath it with the help of a scatter plot.
- The launch selected in the dropdown is also communicated to the bottommost scatter plot.
- These visualizations have been created mainly to make them dynamic and help gather more in-depth insights.
- Github URL: https://github.com/OmkarKadam2002/Applied-data-science-capstone-Coursera/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)



Github URL:
https://github.com/OmkarKadam2002/Applied-data-science-capstone-Coursera/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

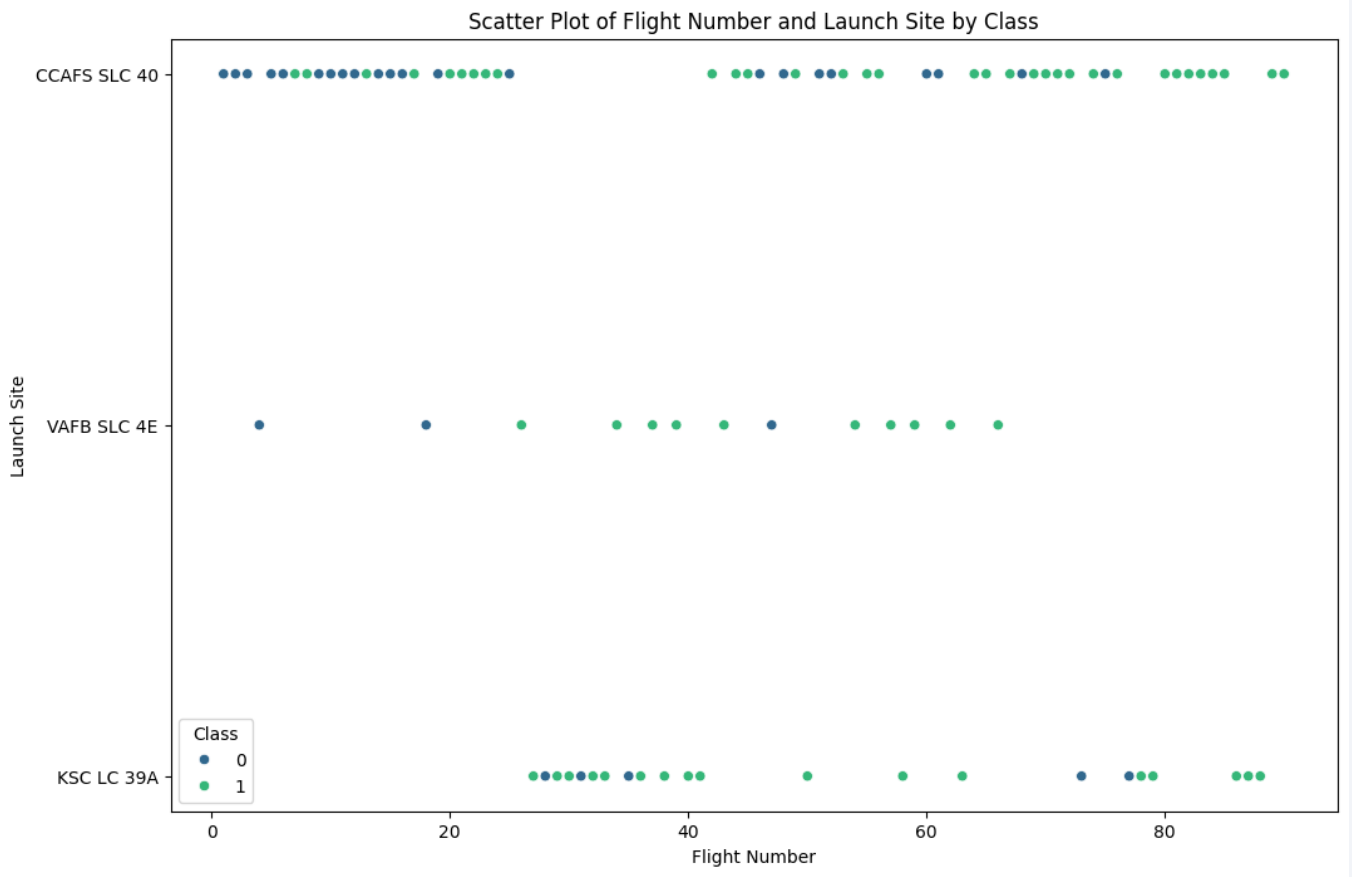
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

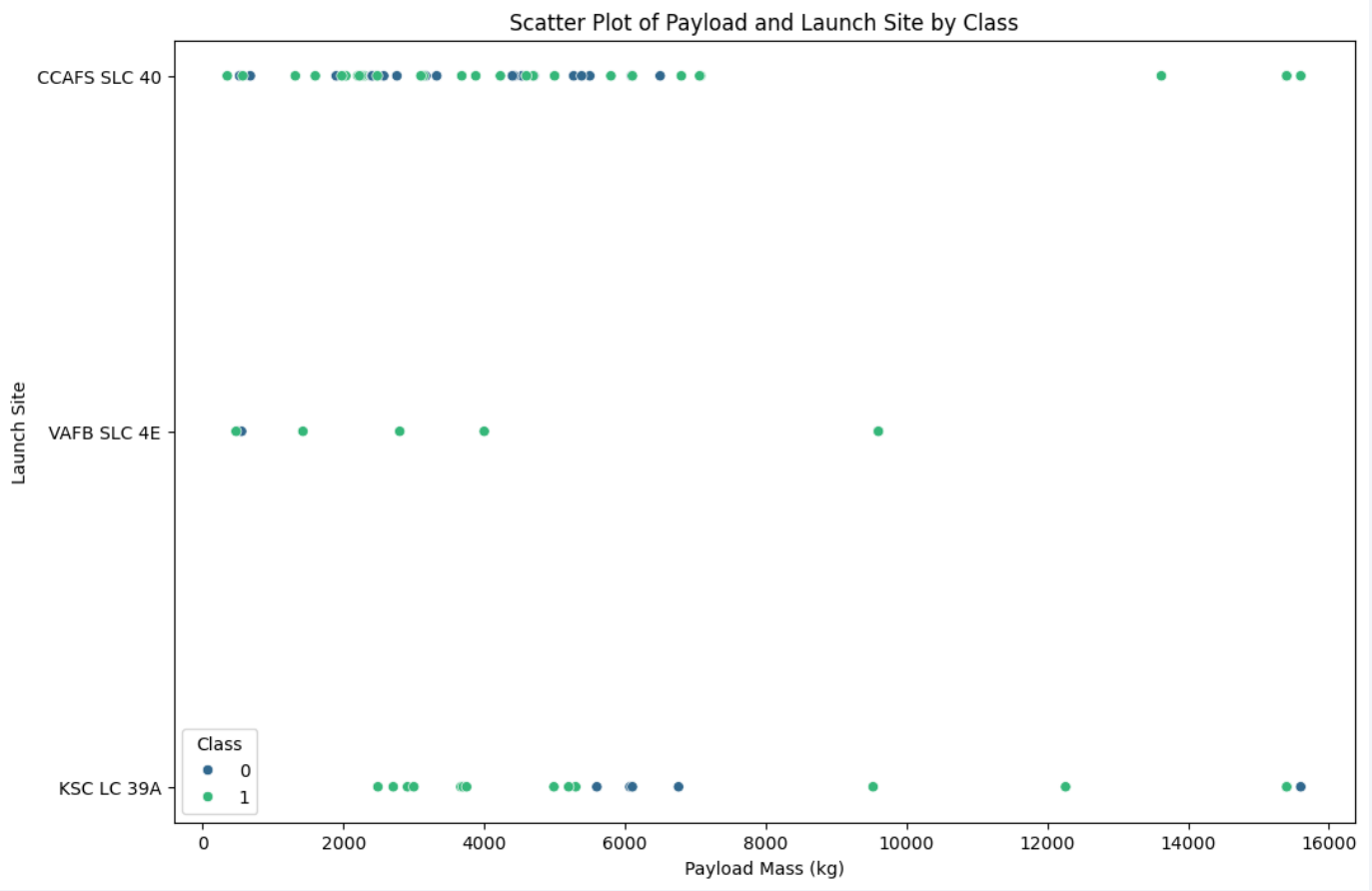
Flight Number vs. Launch Site

- In general, for every launch site, as the flight number increased, so did the number of successful outcomes.
- From the scatter plot, we can also say that the launch site KSC LC 39A has the highest success rate.



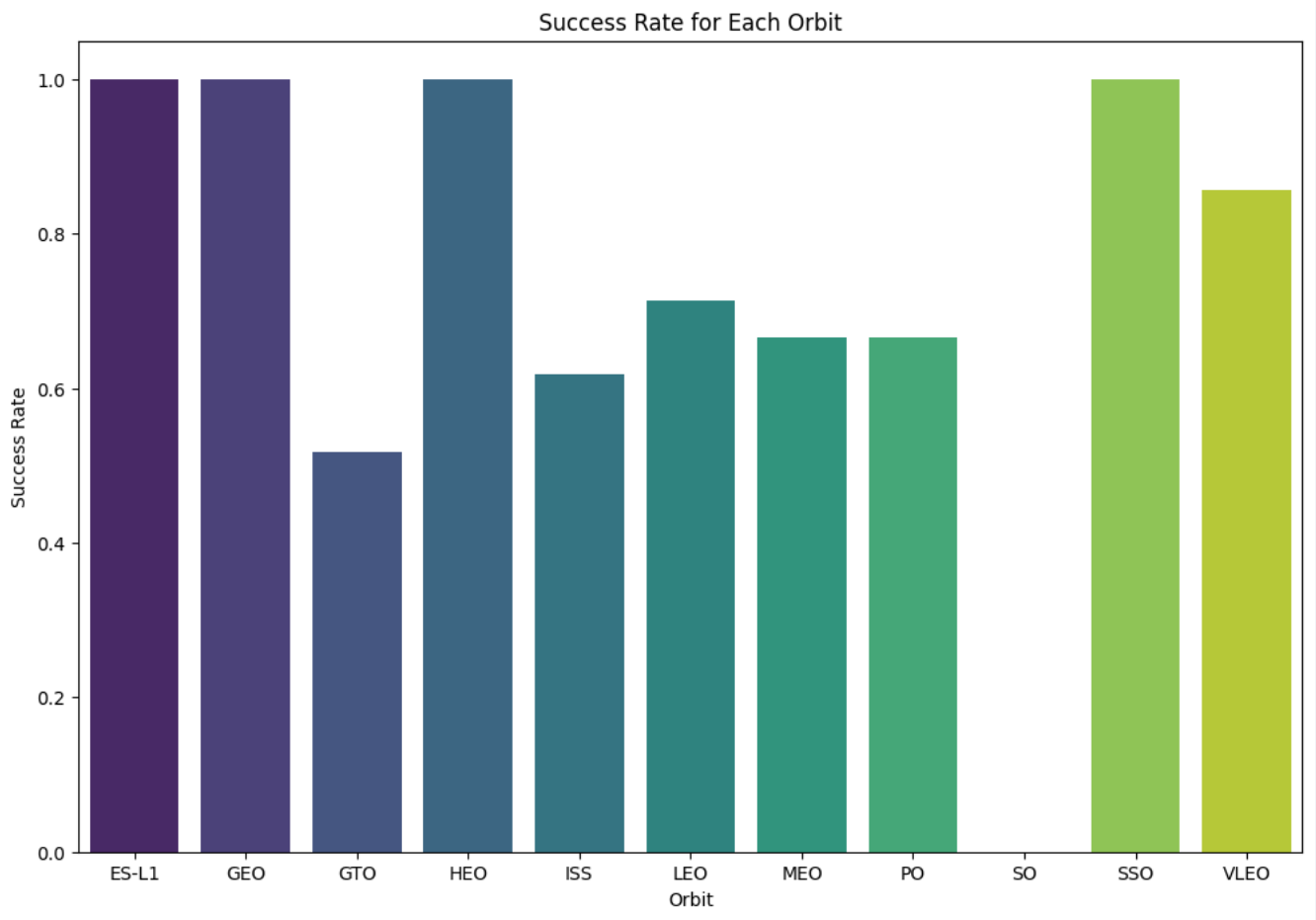
Payload vs. Launch Site

- Payloads above 8000 are successful (with the exception of payload in near to 16000 at launch site KSC LC 39A)
- Consequently, lower payloads have more chances of being a failure
- The larger payloads are generally deployed at sites CCAFS SLC 40 and KSC LC 39A



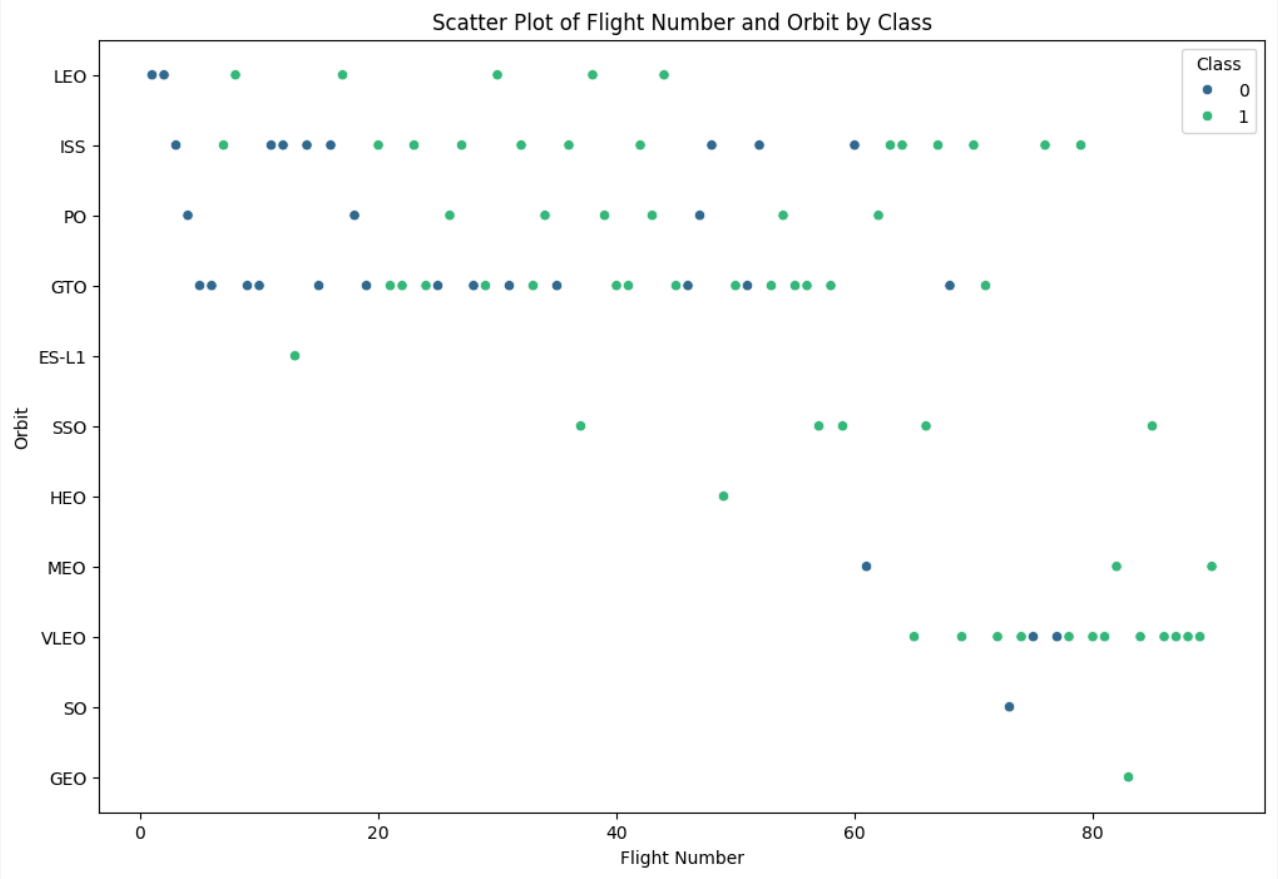
Success Rate vs. Orbit Type

- Orbits ES-L1, GEO, HEO, SSO have the highest success rates.
- Orbit SO has not had a single successful outcome



Flight Number vs. Orbit Type

- Out of the four orbits with highest success rates identified in the previous slide, only SSO has had more than one launch, and is seemingly the best orbit for a successful outcome.

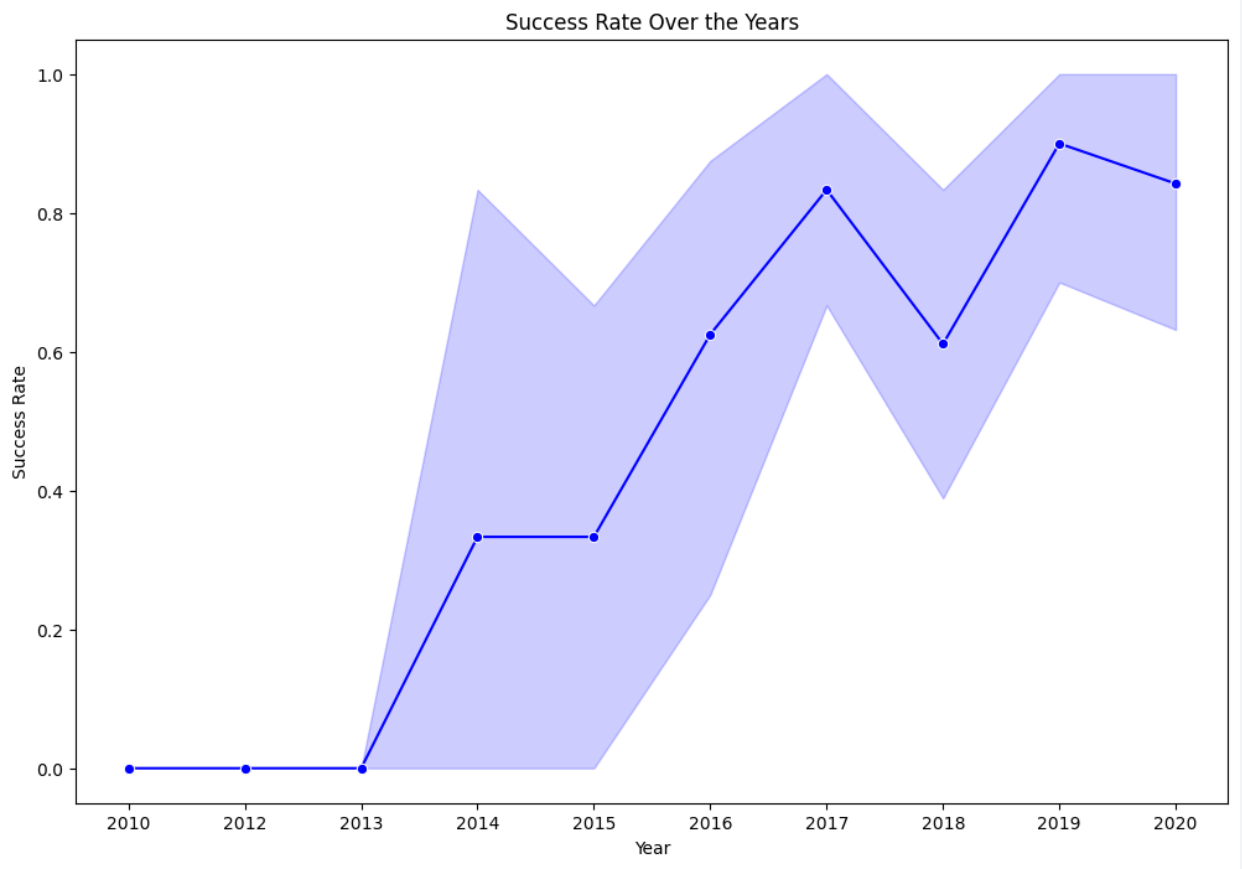


Payload vs. Orbit Type

- ISS, PO and VLEO are the only orbits which have had significantly higher payloads. Out of which ISS and LEO have the highest success rates.
- GTO orbit is tough to analyze at it has positive and negative outcomes.

Launch Success Yearly Trend

- The success rate was practically zero for almost 4 years.
- After that it has been steadily increasing with the peek observed in the year 2019.



All Launch Site Names

%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE

Usage of the word distinct is what removes identical values.

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

Launch Site Names Begin with 'CCA'

%sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5

The % is what allows for matching for all strings containing CCA, regardless of whatever it follows.

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

Total Payload Mass

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

SUM() provides the cumulative addition of Payload_Mass_Kg_ for NASA (CRS) customers and displays it has Total_Payload_Mass

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

- %sql SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Payload_Mass FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1';
- This SQL query calculates the average payload mass for the booster version 'F9 v1.1' in the SPACEXTABLE dataset.

Average_Payload_Mass
2928.4

First Successful Ground Landing Date

%sql SELECT MIN(Date) AS First_Successful_Landing_Date FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)';

Selects the oldest year from the table when a landing was successful on the ground pad.

First_Successful_Landing_Date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

%sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS_KG < 6000

Selects the booster version from the table for which landing has been successfully done on drone ship and the payload mass was been 4000 and 600

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

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Success%') AS 'SUCCESS', (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS 'FAILURE'
```

Query returns counts of mission outcomes in for the launches grouped by success and failure.

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

SUCCESS	FAILURE
100	1

Boosters Carried Maximum Payload

%sql SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABL)

A subquery is used to select boosters with max payloads.

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

%sql SELECT substr(Date, 6, 2) AS Month, Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXTABLE WHERE substr(Date, 0, 5) = '2015' AND Landing_Outcome LIKE 'Failure (drone ship)

This query returned month, booster version, launch site, and additionally, Landing outcome

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

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

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

Selects Landing outcomes as distinct values and ranks them according to their magnitude.

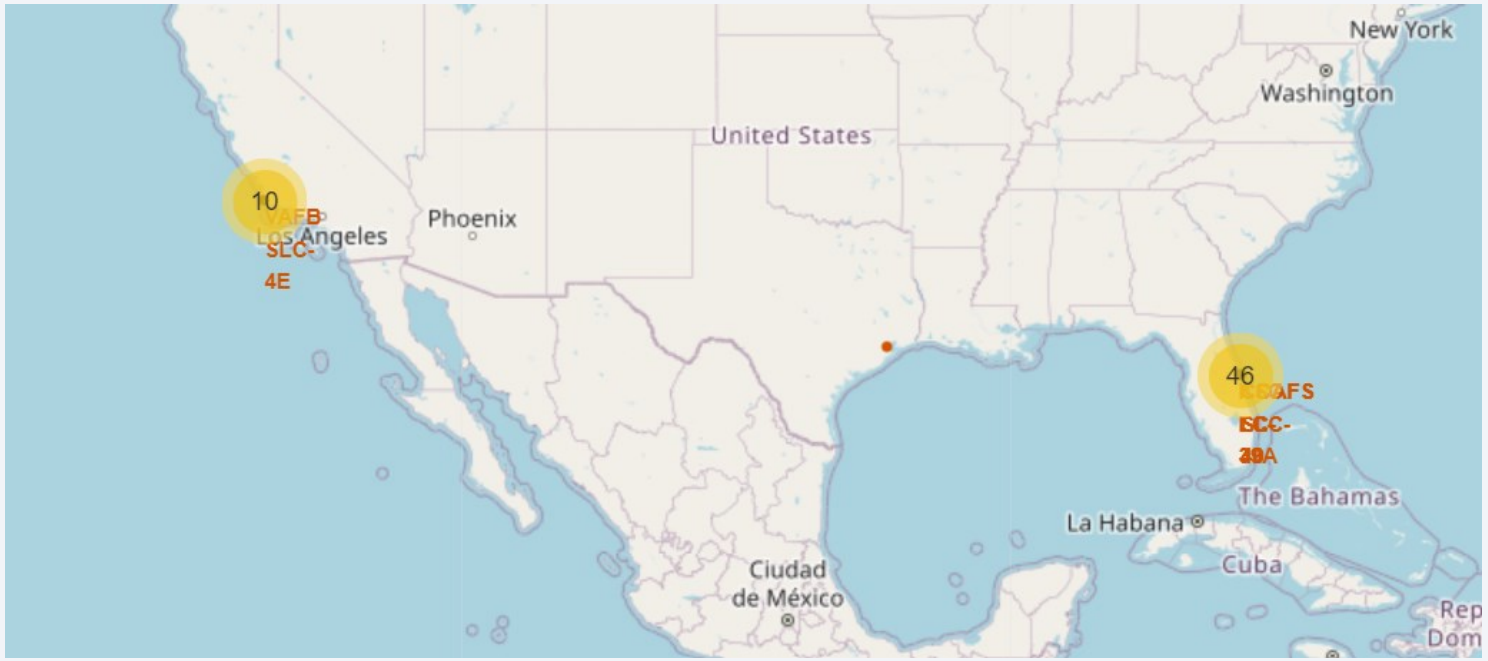
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

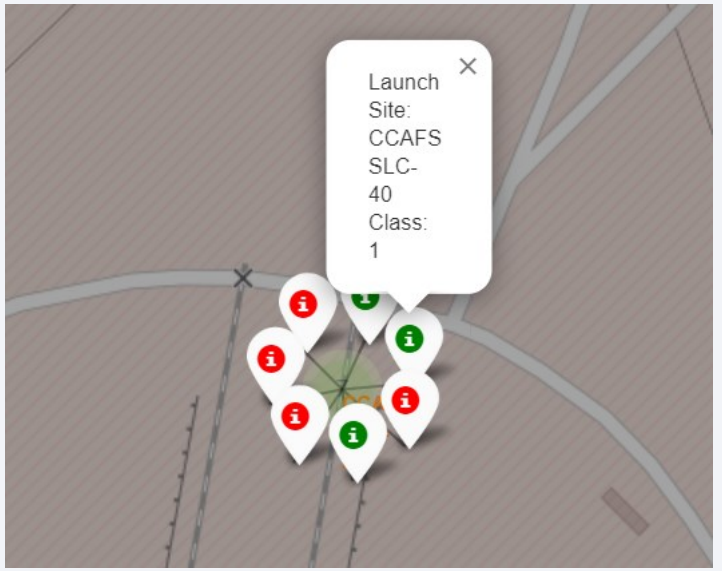
All Launch Site Locations

- All the launch sites are in USA. Particularly in the states of Florida and California.
- Launch sites are significantly close to the coasts.



Color Labelled Launch Outcomes

- Green popups represent successful outcomes.
- Red popups represent failed outcomes

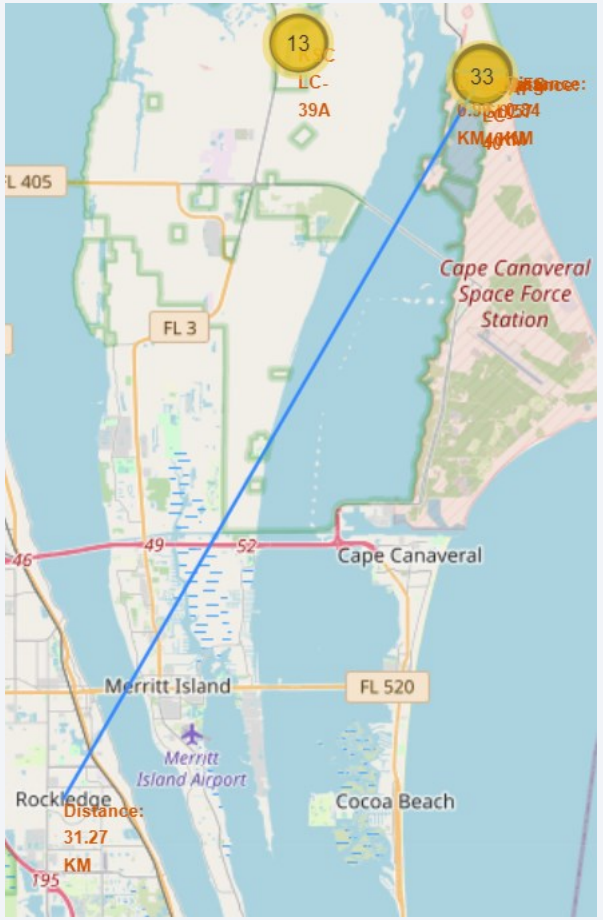


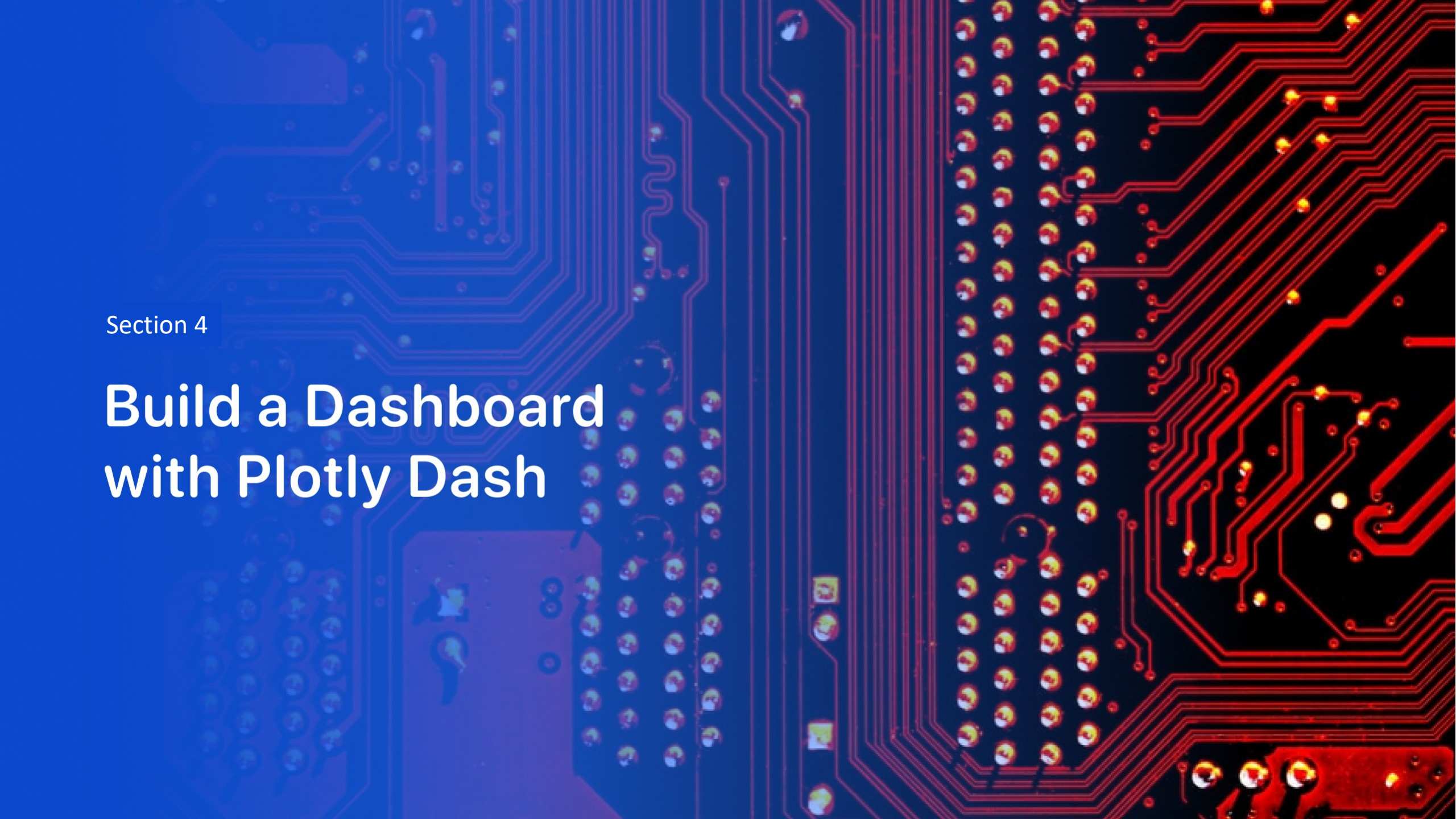
Launch Site Proximity

- Launch sites are close to railroads and coastlines.
- Launch sites aren't necessarily close to highways.

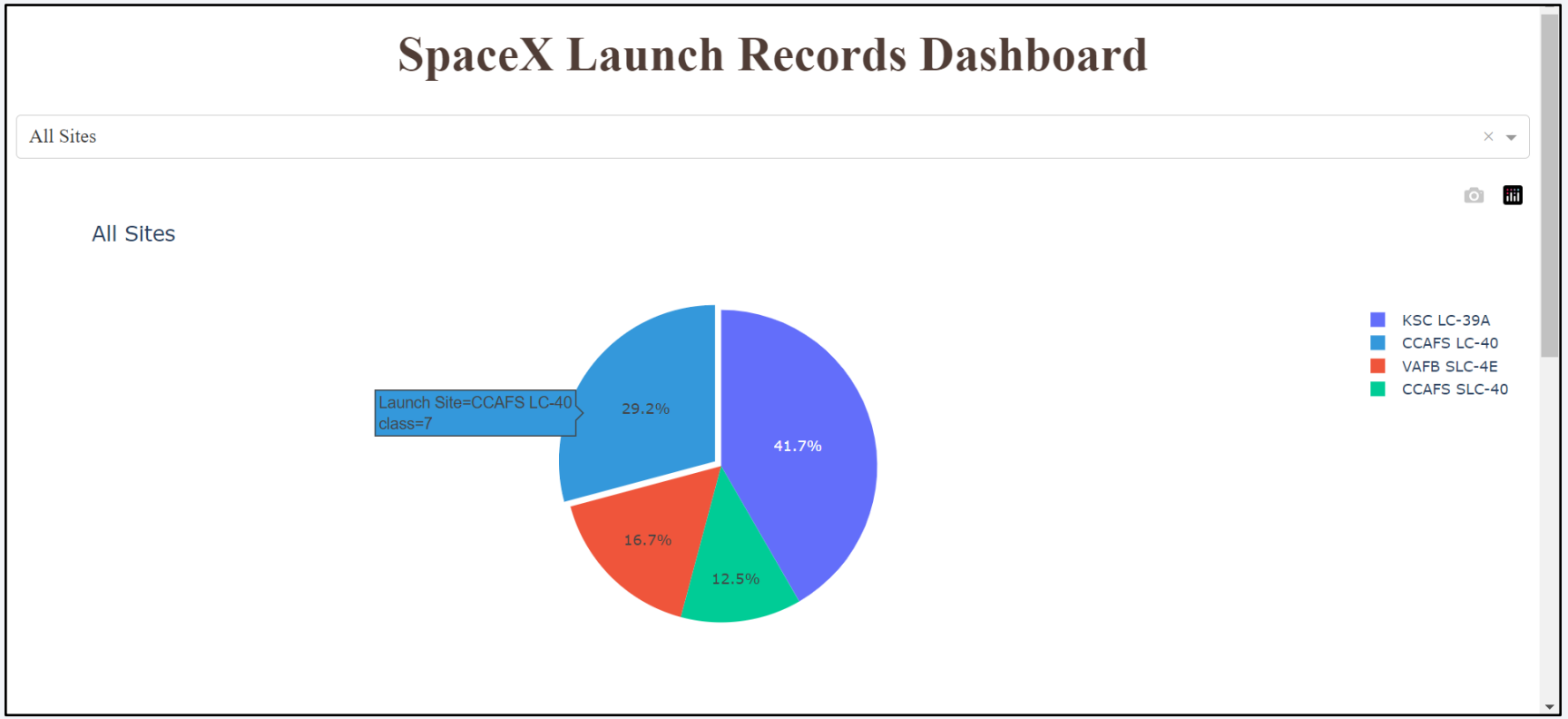
(Contrary to what's observed in these images, the launch sites In California are not in close proximity with Highways)

- All Launch sites are considerably away from cities.



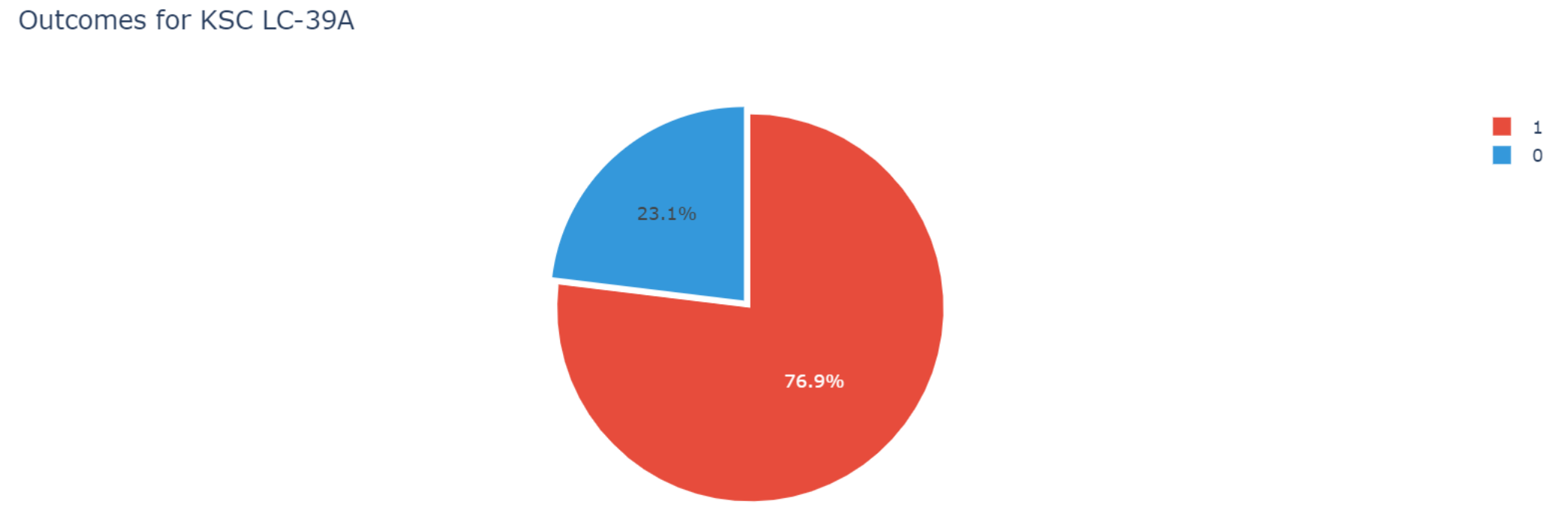

Section 4
Build a Dashboard with Plotly Dash

Total Success Launches for All Sites



- Launch sites CCAFS SLC – 40 and KSC LC-39 have the lowest and highest success rates respectively.

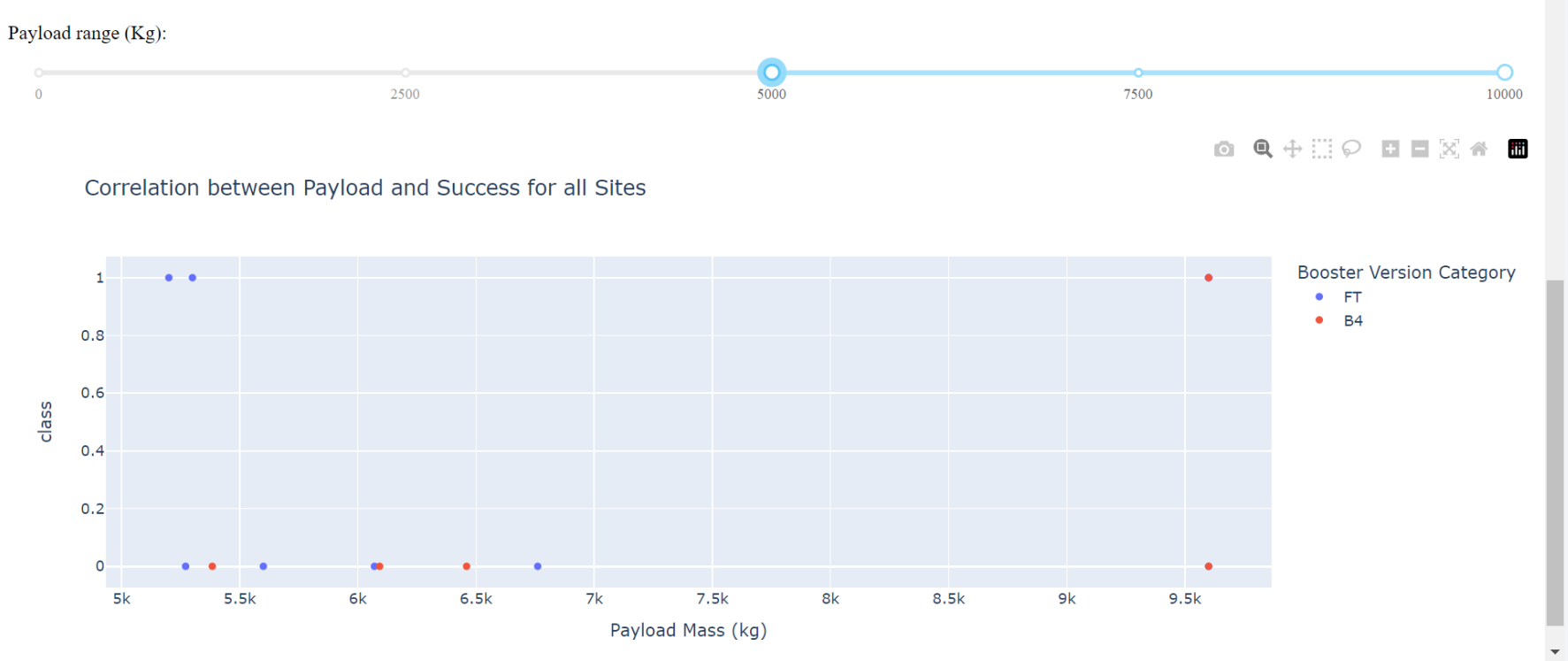
Total Launches for Site KSC LC-39A



- This site has a success rate of more than 75%.

Payload Mass VS Launch Outcome Scatter Plot

- Success rate is less with heavy payload masses.



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The models that were built included: Logistic Regression, Support Vection Machine, Decision Tree, K-Nearest Neighbours.
- All Models showed the same level of accuracy for test data.(~83%)
- Decision tree showed highest accuracy on Training data. (~86%)

```
accuracy = logreg_cv.score(X_test, Y_test)
print("Accuracy on Test Data:", accuracy)
```

```
Accuracy on Test Data: 0.8333333333333334
```

```
accuracy = svm_cv.score(X_test, Y_test)
print("Accuracy on Test Data:", accuracy)
```

```
Accuracy on Test Data: 0.8333333333333334
```

```
accuracy = tree_cv.score(X_test, Y_test)
print("Accuracy on Test Data:", accuracy)
```

```
Accuracy on Test Data: 0.8333333333333334
```

```
accuracy = knn_cv.score(X_test, Y_test)
print("Accuracy on Test Data:", accuracy)
```

```
Accuracy on Test Data: 0.8333333333333334
```

```
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
```

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 4,
'min_samples_split': 10, 'random_state': 0, 'splitter': 'random'}
accuracy : 0.8625
```

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

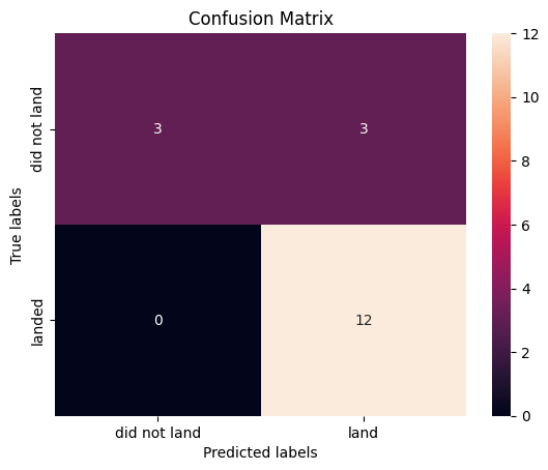
```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

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

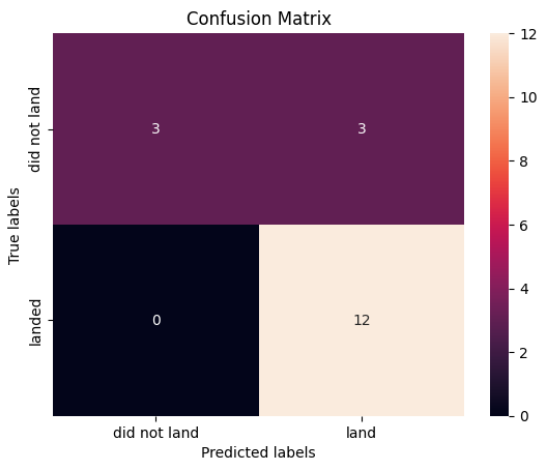
```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

Confusion Matrix

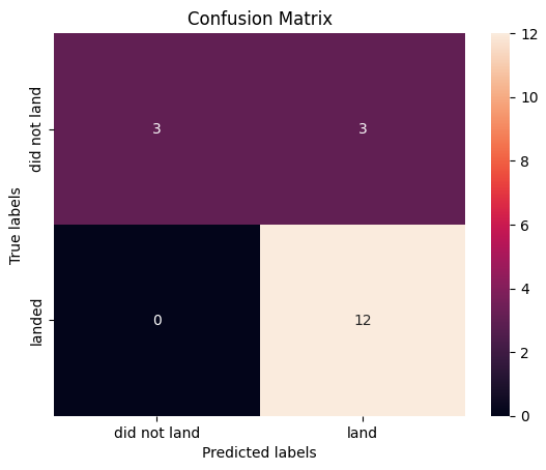
- Confusion Matrix for all Models is the same as they have the same accuracy



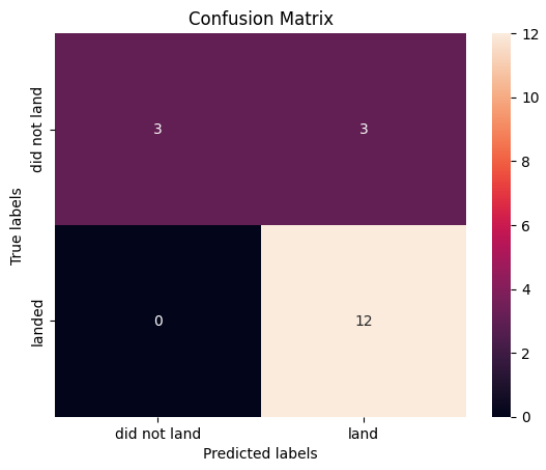
Logistic Regression



Support Vector Machine



Decision Tree



K-Nearest Neighbours

Conclusions

There are a few conclusions to that can be made:

- Launch Site KSC LC-39A has the highest success rate in all launch sites.
- Launch sites are general in close proximity with coastlines, railroads, highways, and are far away from cities.
- Success rate of launches increases with the increase in flight number.
- Orbit SSO is the most favorable orbit for a successful outcome as all of it's launches have been successful.
- Decision Tree model works best our data for predictive analysis. The model has an adequately acceptable accuracy of approximately 86%.

Thank you!