IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

S Srivatsan
16/01/2025

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- Project background and context

We are a new Space company called Space Y. We would like to perform Data Science task that can figure out if SpaceX falcon9 rocket will land or not. The data collected contains various features that can help us with the task,

- Problems you want to find answers

The problem that we need to find answer for is if a SpaceX falcon9 rocket will land or not.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - The data required is collected using 2 sources (SpaceX json Rest api(api.spacexdata.com), Wikipedia)
- Perform data wrangling
  - The data that was extracted is not in the correct format for the prediction. So, we convert categorical data into numerical values using one hot encoding.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
- We build different machine learning models to perform classification and we use different metrics to find the accuracies of these algorithms,

Data Collection

- Data needed for this project is collected from 2 sources.

1. SpaceX Rest Api is used to get data that is available in JSON format.

    https://api.spacexdata.com/

The data from SpaceX Rest Api is extracted using the requests module in python to communicate with SpaceX rest api and gets the in the JSON fromat. This data is then converted to a pandas DataFrame and then saved to a csv file.

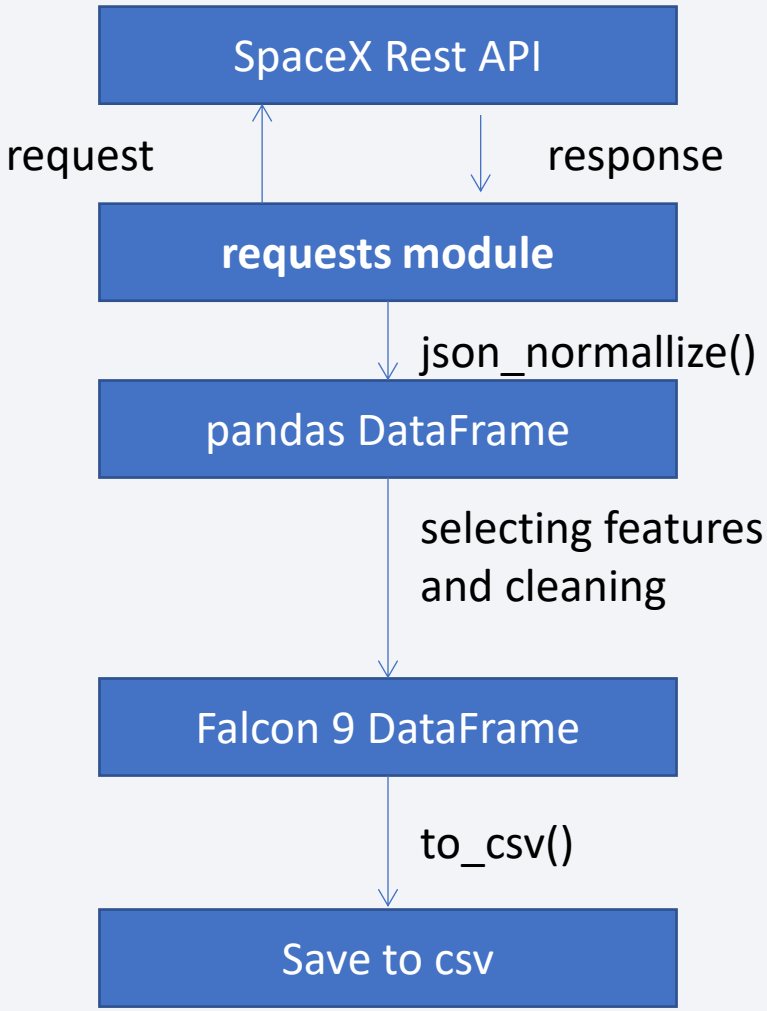
2. Wikepedia

    https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

The data in Wikipedia is a table in a html webpage. To extract this data, we need to use webscraping. First we use the requests module to communicate with Wikipedia to get the response in html format. Then the webscraping is done by using beatiful soup. And then the data is converted into a pandas DataFrame and saved to a csv file.

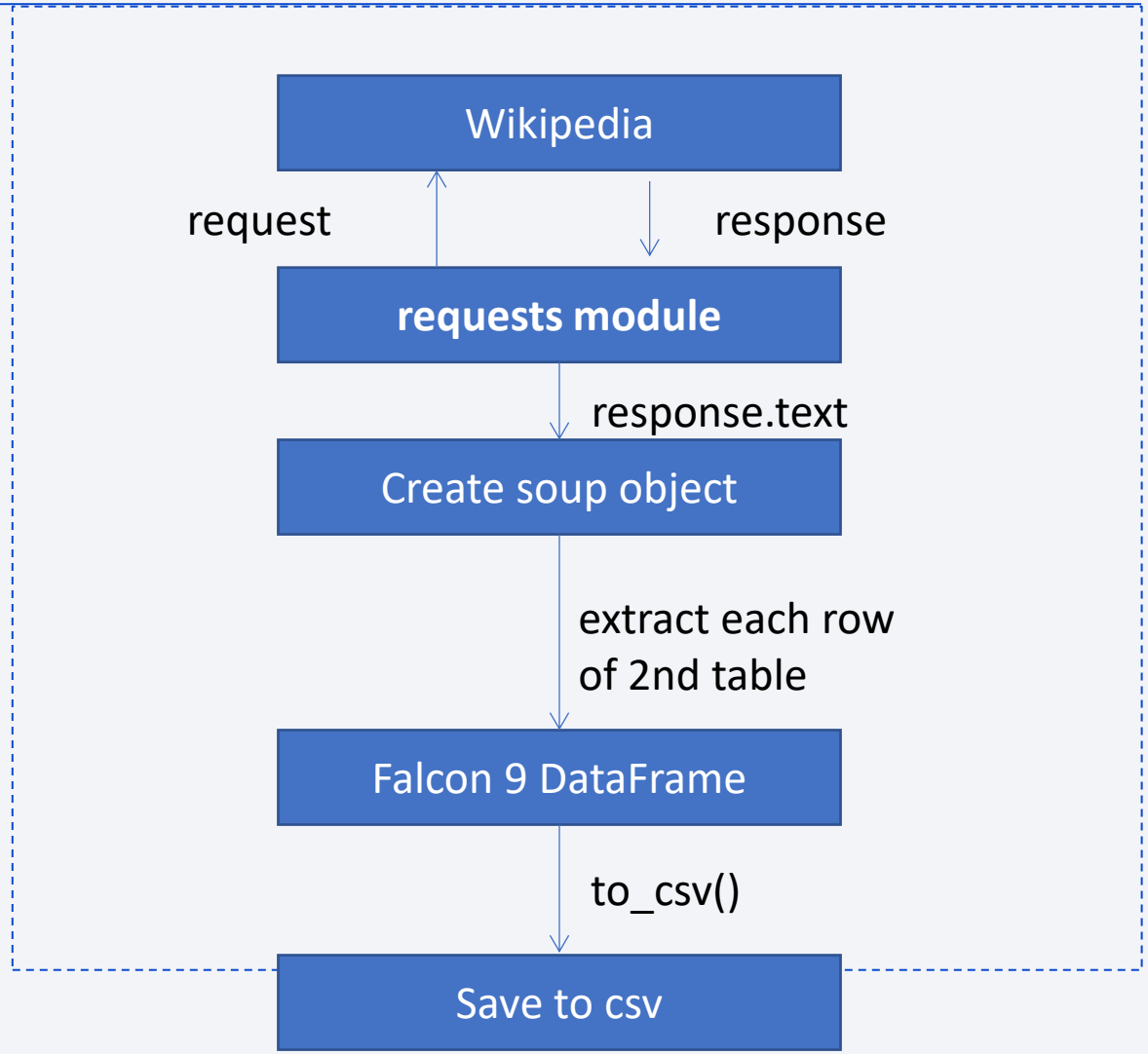
Data Collection – SpaceX API

- The data from SpaceX Rest Api is extracted using the requests module in python to communicate with SpaceX rest api and gets the in the JSON fromat. This data is then converted to a pandas DataFrame by performing json normalization to make sure we do not have any nested data.
- Then we select only certain features necessary for the analysis. We also remove all the falcon 1 related rows and only retain falcon 9 rockets
- After some more cleanup, we save the data to a csv file.
- You can lookat the data collection process in this github link.



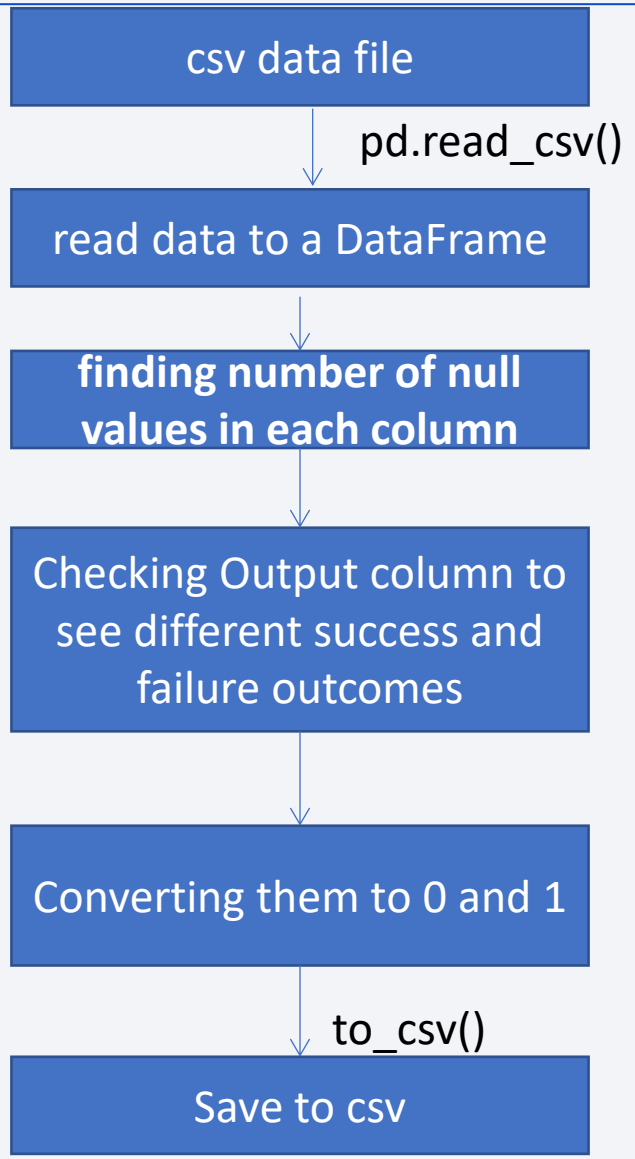
Data Collection - Scraping

- The data from Wikipedia is extracted using the Web Scraping using beautiful Soup in python. We first use request to grab the html page. Then we create a beautiful soup object.
- We then loop through each row of the 2nd table and get columns that is necessary for us and convert it to a DataFrame
- Finally we save the data into a csv file.
- You can look at the data collection process in this github link.



Data Wrangling

- We start by reading the data that we stored in the csv file using pd.read_csv(). When we look at the data, we have a few columns have categorical data including the outcome column. Which needs to be changed to numerical values. The independent columns are converted to numerical values after some EDA is finished. For now, we find out how many null values exists in each column and convert the outcome column to numerical values and save it into a new csv file.
- You can find the jupyter notebook for the Data Wrangling process in this Github Link



EDA with Data Visualization

We start by plotting multiple scatter plots anc catplot to find out relationships between different columns over the class.

- x="FlightNumber", y="PayloadMass", hue="Class"
- x='FlightNumber', y='LaunchSite', hue = 'Class'
- x = 'PayloadMass', y = 'LaunchSite', hue = 'Class'
- x = 'FlightNumber', y = 'Orbit', hue = 'Class'
- x = 'PayloadMass', y = 'Orbit', hue = 'Class',

We then create a bar plot that shows the average success rate for each orbit

We create a line plot between Year of launch vs success rate. This is because we have time series data.

Finally, we perform one hot encoding to convert all the independent columns with categorical data into numerical data.

You can find the jupyter notebook for the EDA with Data Visualization in this Github Link

EDA with SQL

- In this section, we will use the sql magic command in jupyter notebook to perform EDA with the data.
- We start by reading data from csv and saving it into a database file.
- We use different columns to understand different things like total number of times each outcome is occuring, Launch Sites for Failed DroneShip landings, getting booster versions with largest payload mass, date of the first succesful landing, total and average payload masses and much more
- We use select commands to grab the data and different concepts like where clauses, group by, sub queries, ordering and aggregate functions to expore data to get useful insights
- You can find the jupyter notebook for the SQL EDA in this Github Link

Build an Interactive Map with Folium

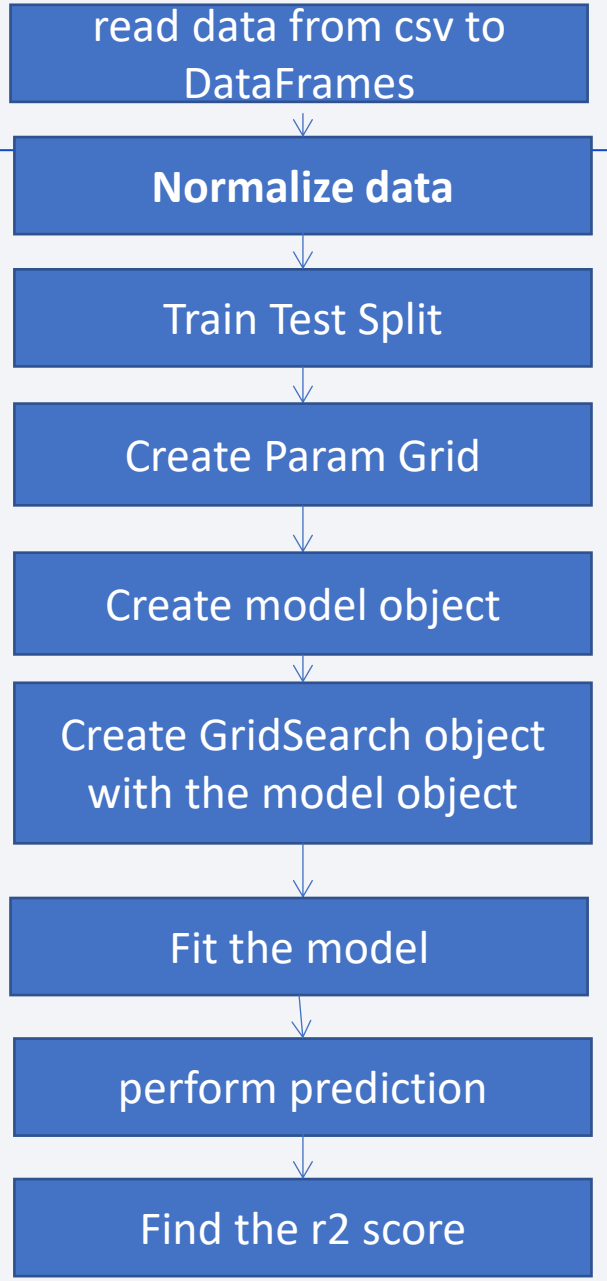
- In this section we create a USA map and we place markers for the launch sites based on the coordinates. We use different kinds of map objects like circle, marker, marker cluster, PolyLine.
- The Marker is used to specify the exact location of an Launch. A LaunchSite can have multiple launch locations.
- A Circle object representes the LaunchSite.
- A marker cluster clusters the markers and shows how many launch locations are in an area depending on the zoom level.
- The PolyLine is a line between two locations. This is used to show the distances between the launch site and nearby places.
- You can find the jupyter notebook for the Folium Interactive map in this Github Link

Build a Dashboard with Plotly Dash

- In this section we built a plotly dash application. The dash app contains a dropdown and a range slider and 2 graphs.
- The dropdown contains options for each LaunchSite. We can also select if we want to display graph for all LaunchSites.
- The range slider is to represent payload mass.
- The first graph shows pie chart of success rate for the launch site that we lect using the dropdown.
- The second graph shows scatter plot of payload Mass vs class. The data is filtered based on the selected LaunchSite from the dropdown and payload mass withing the range selected from the RangeSlider.
- You can find the python file for the Plotly Dash App in this Github Link

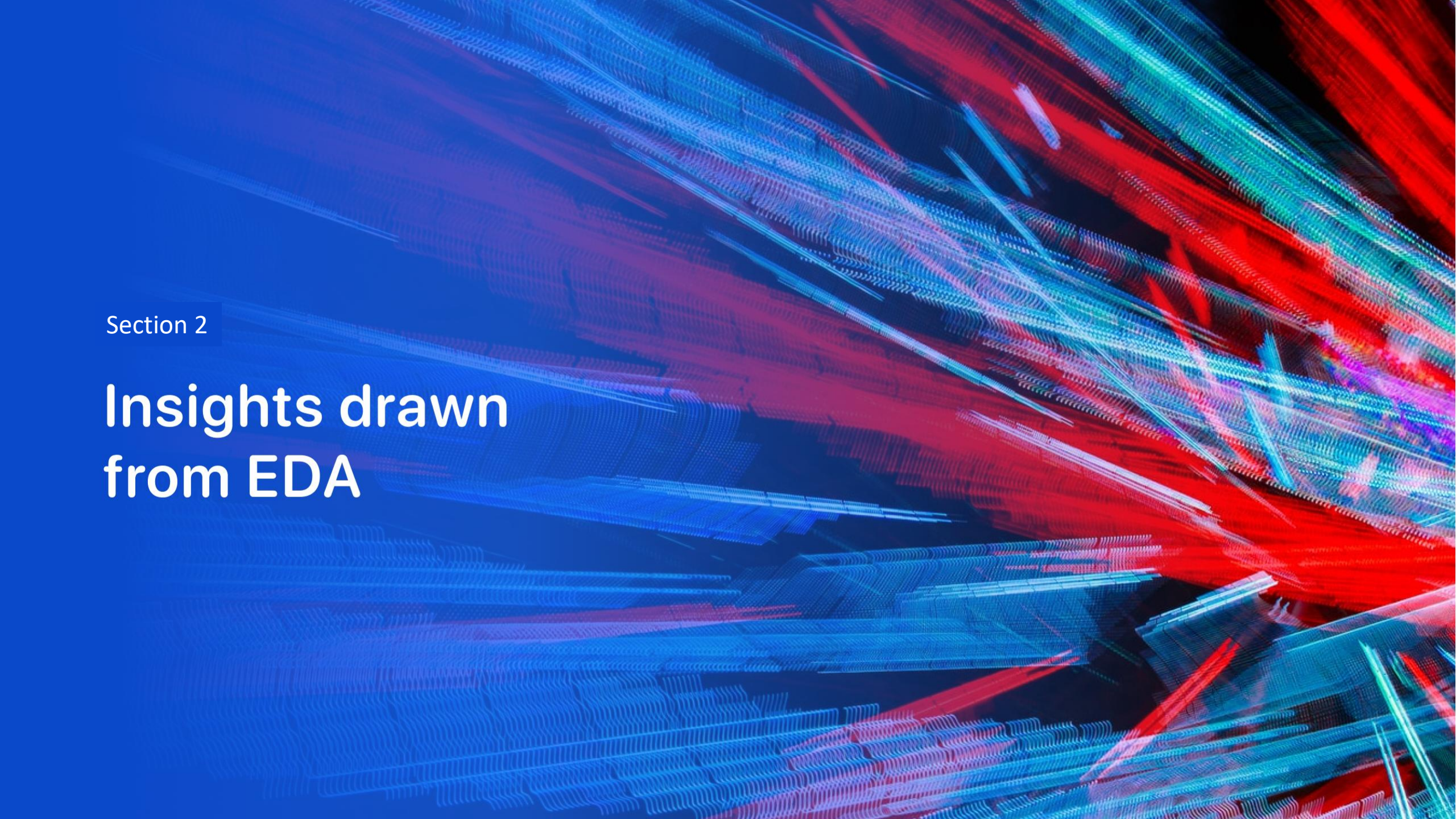
Predictive Analysis (Classification)

- In this section, we build different models like LogisticRegression, SVM, DecisionTreeClassifier, K nearest neighbors using GridSearch to find the best estimator for each model.
- We then use these models for training and testing.
- Then we fing the r2_score.
- Based on the r2_score, we can say SVM is the best model among others.
- You can find the jupyter notebook for the predictions in this Github Link



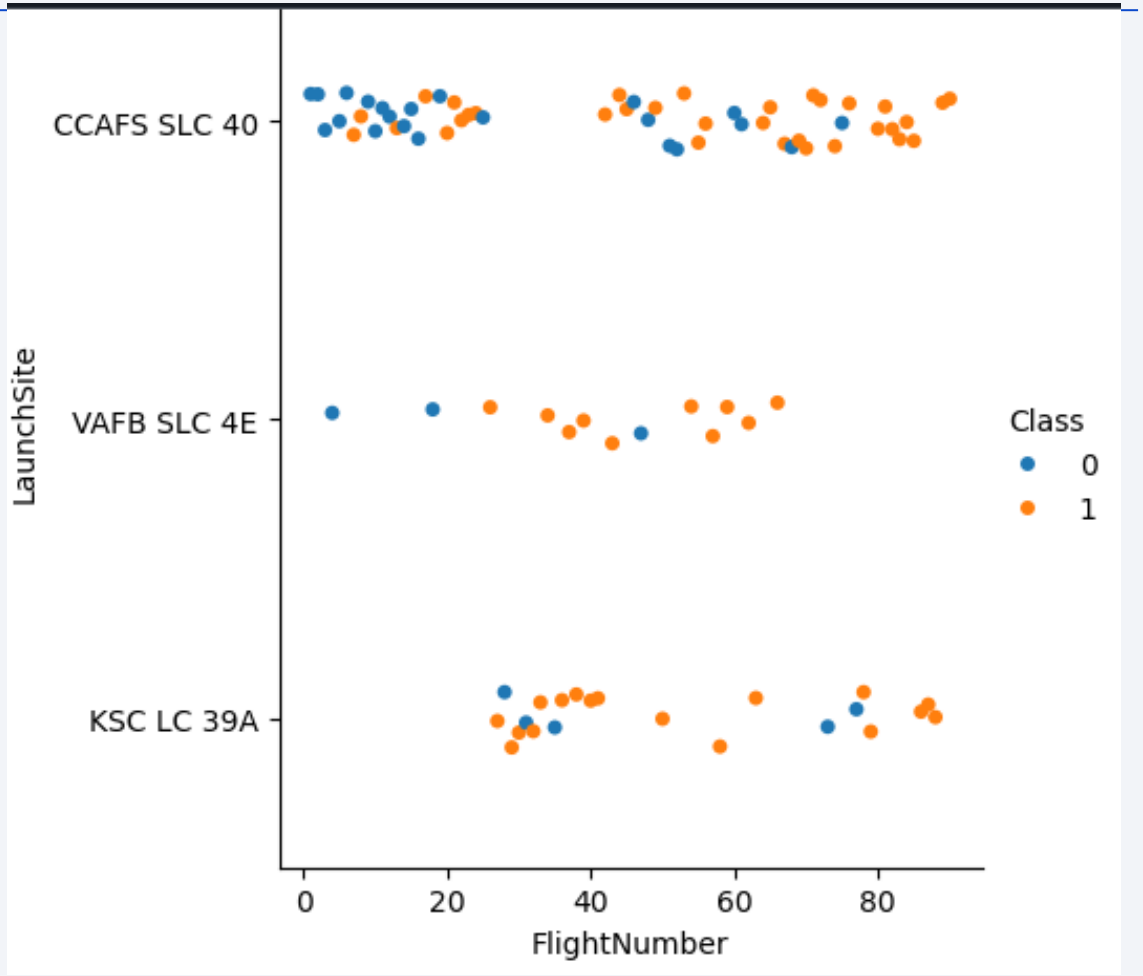
Results

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


Section 2
Insights drawn from EDA

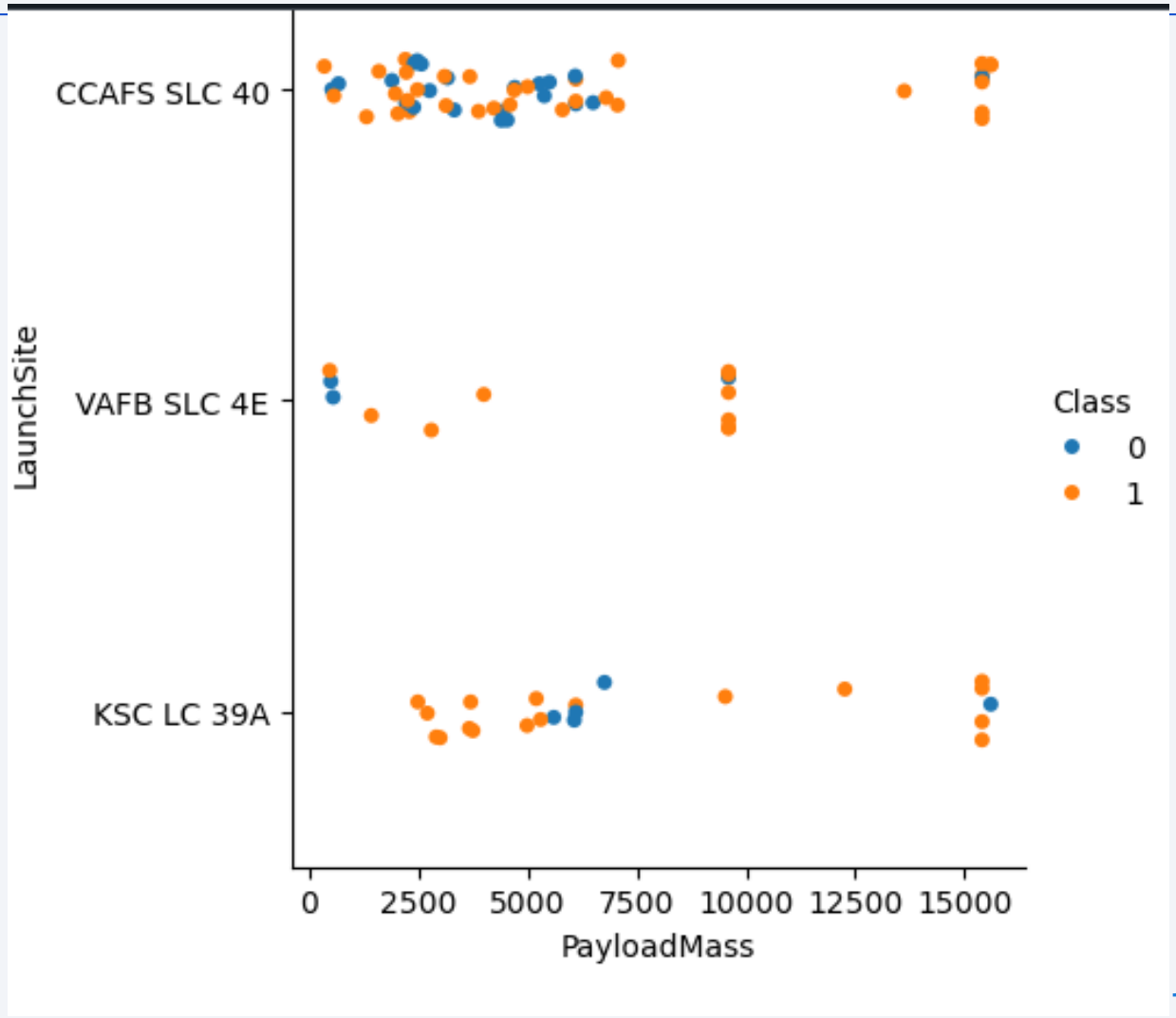
Flight Number vs. Launch Site

- Here is a Scatter plot that shows the relationship between The FlightNumber and LaunchSite, Class
- Insight: CCAFS SLC 40 has way more failures compared to the remaining Landing Sites. Also lower flight numbers have more failures



Payload vs. Launch Site

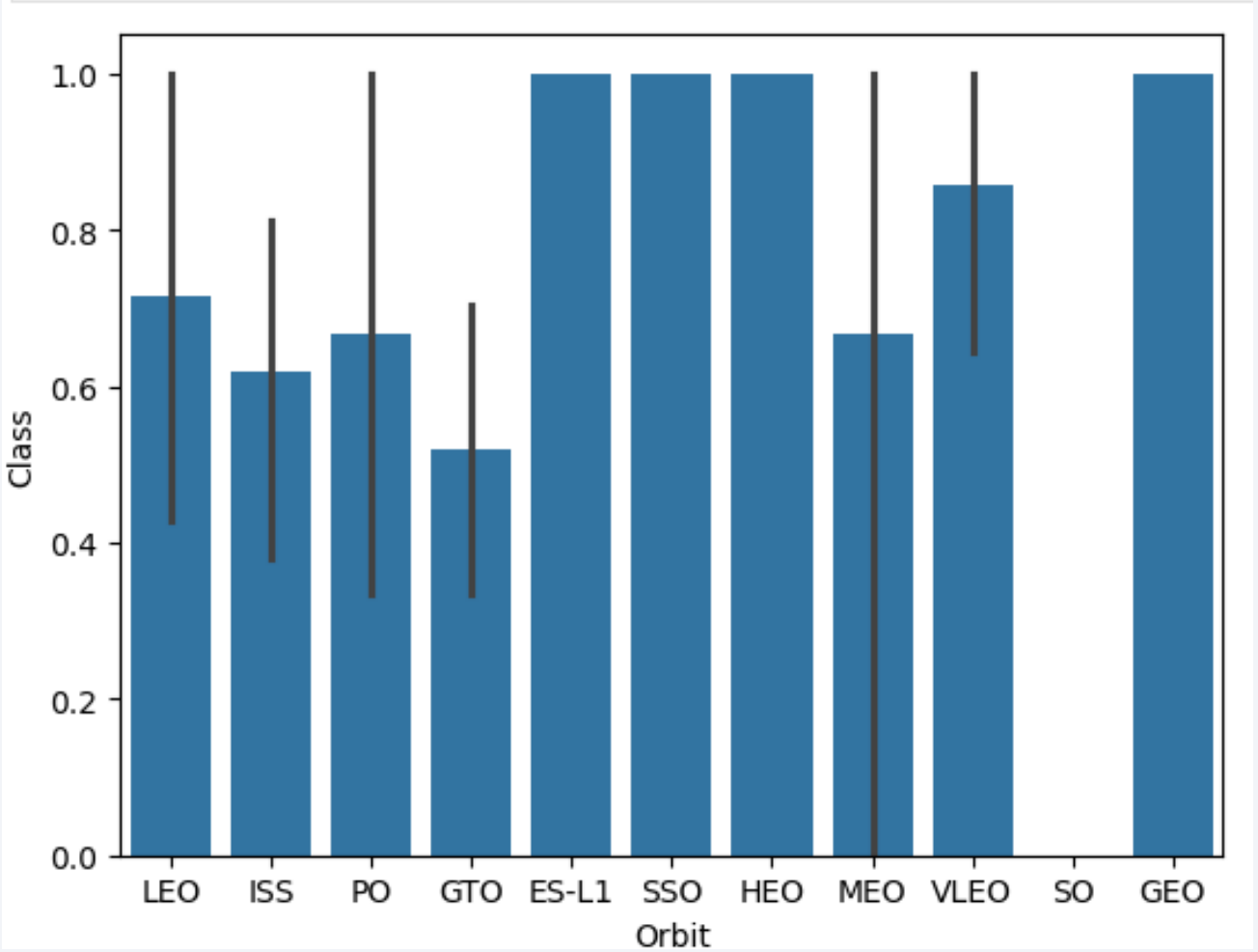
- Here is a Scatter plot that shows the relationship between Payload and Launch Site, Class
- If we observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).



Success Rate vs. Orbit Type

- Here is a bar chart for the success rate of each orbit type

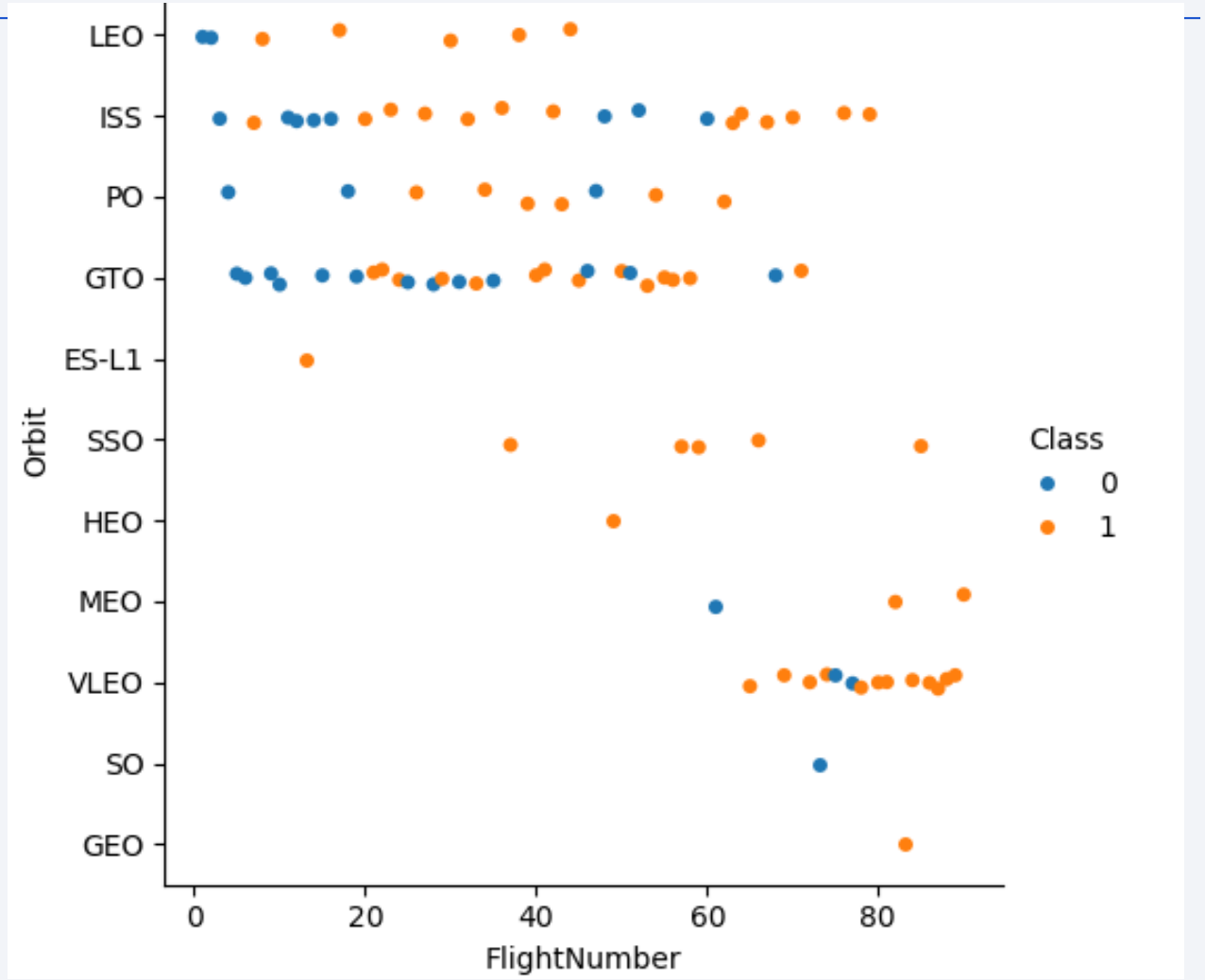
- We can see that ES-L1, SSO, HEO, GEO have the highest success rates.



Flight Number vs. Orbit Type

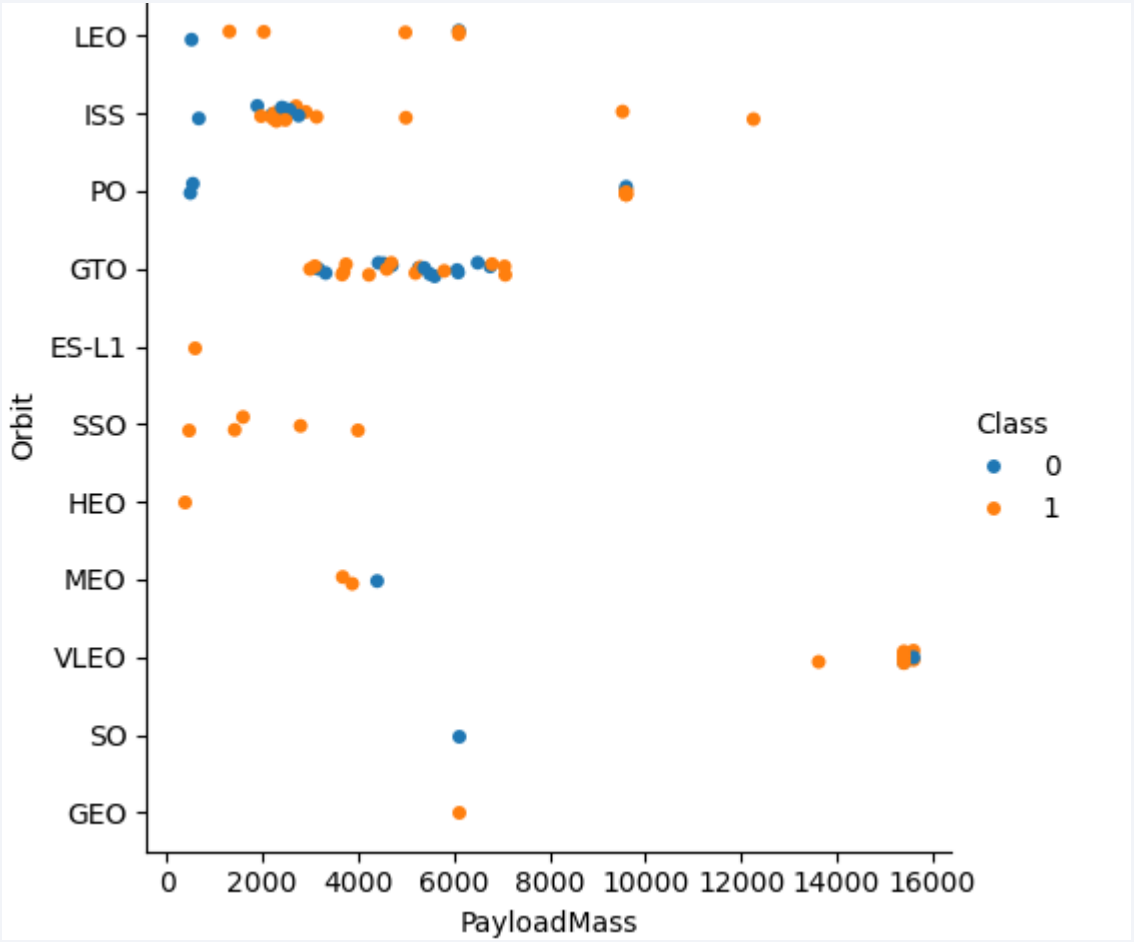
- Here is a Scatter plot for Flight number vs. Orbit type vs Class

- We can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.



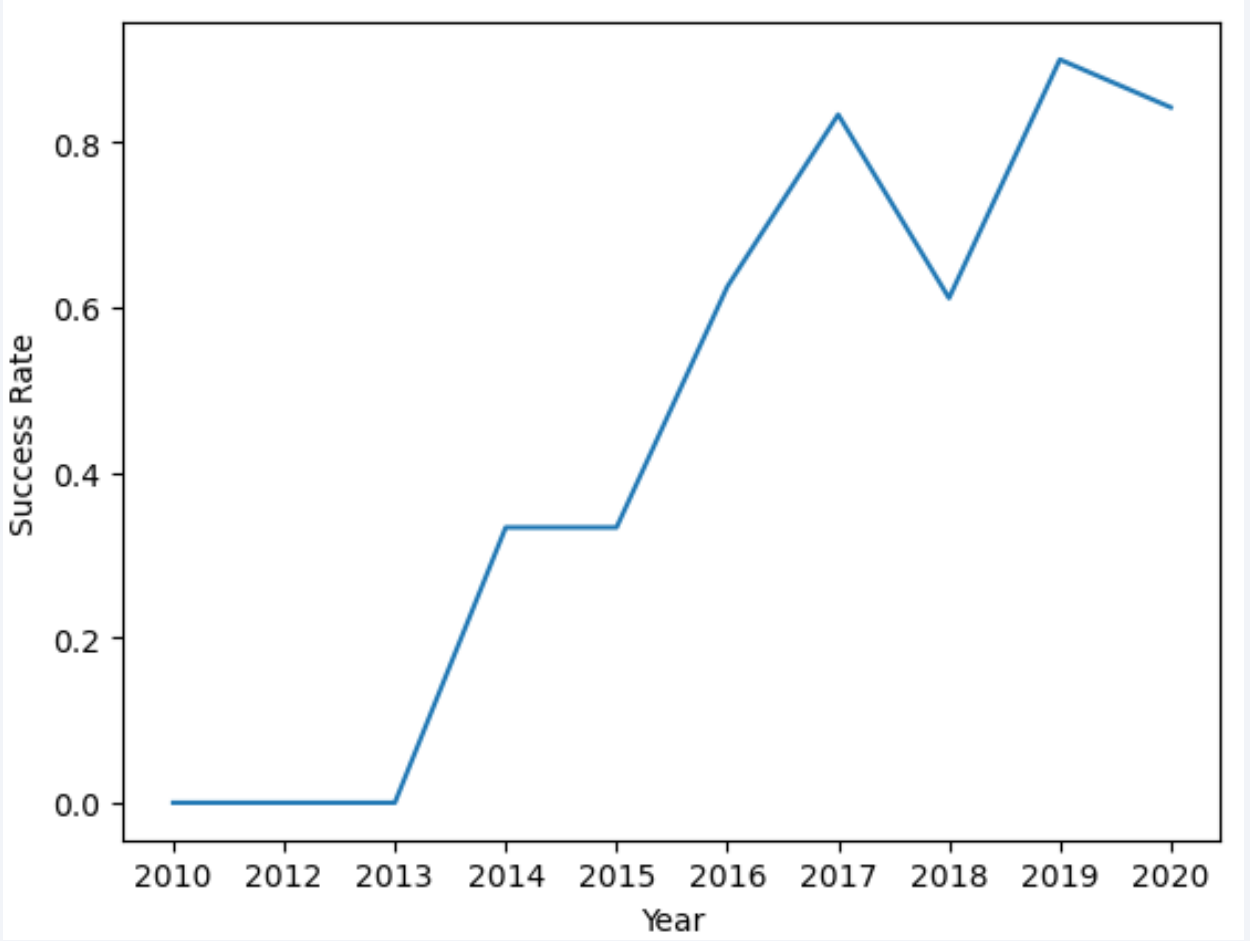
Payload vs. Orbit Type

- Here is a scatter plot of payload vs. orbit type vs class
- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.



Launch Success Yearly Trend

- Here is a line plot that shows the Success Rate over the years.
- you can observe that the sucess rate since 2013 kept increasing till 2020



All Launch Site Names

- Find the names of the unique launch sites
- Query : select distinct "Launch_Site" from SPACEXTABLE;
- The query uses distinct keyword to get unique values from the column Launch_Site.

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- select "Launch_Site" from SPACEXTABLE
- where "Launch_Site" like 'CCA%'
- limit 5;
- This query uses the where clause to filter the data that contains first 5 Launch_Sites that starts with CCA

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

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- select Customer, sum(PAYLOAD_MASS__KG_) as TOTAL_MASS
- from SPACEXTABLE
- group by(Customer)
- having Customer = 'NASA (CRS)';
- To get the total payload mass from NASA, we need to perform a groupby with a sum() aggregate function. To filter the data to contain only NASA, use the having clause.

Customer	TOTAL_MASS
NASA (CRS)	45596

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- select "Booster_Version", AVG(PAYLOAD_MASS__KG_) as AVG_MASS
- from SPACEXTABLE
- group by("Booster_Version")
- having "Booster_Version" = 'F9 v1.1';
- To get the average payload mass of F9 v1.1 booster we need to perform a groupby with a avg() aggregate function. To filter the data to contain only F9 v1.1, use the having clause.

Booster_Version	AVG_MASS
F9 v1.1	2928.4

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

```
select "Landing_Outcome", min(date) as "First Success Landing"
from SPACEXTABLE
        where "Landing_Outcome" = 'Success (ground pad)';
```

Landing_Outcome	First Success Landing
Success (ground pad)	2015-12-22

- We select the columns Landing_Outcome and date columns. We use the min function to find the first date. We filter the data that has only Successful Ground pad landing using the where clause.

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
select "Booster_Version", "PAYLOAD_MASS__KG_", "Landing_Outcome"  from SPACEXTABLE

where "Landing_Outcome" = 'Success (drone ship)'

and "PAYLOAD_MASS__KG_" between 4000 and 6000;
```

- We select the columns Booster_Verson, Payload_Mass_KG, Landing_Outcome columns. and we use where clause to get data that contains only Successful droneship Landing that has payload mass between 4000 and 6000

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
select "Mission_Outcome", count("Mission_Outcome") "Count" from SPACEXTABLE
group by "Mission_Outcome";
```

For this query we are grouping the Mission_Outcome columns and we are using a count aggregate function to find the Success and failure missions.

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- select distinct "Booster_Version", "PAYLOAD_MASS__KG_" from SPACEXTABLE
- where "PAYLOAD_MASS__KG_" = (
- select max("PAYLOAD_MASS__KG_") from SPACEXTABLE
-)

With this query we are using a sub query . First the internal query is using to get a max Payload_Mass_Kg_ column. then we can use that in the where clause of the main query. In there where cluase we are comparing if the Payload Mass column is equal to result of the sub query.

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

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

select substr(Date, 6,2) as month, "Landing_Outcome", "Booster_Version", "Launch_Site" from SPACEXTABLE

where "Landing_Outcome" = 'Failure (drone ship)' and substr(Date,0,5)='2015'

This query uses the where clause that checks if out Landing_Outcome is Failed Drone Ship and if the date is 2015. to grab the year, we are using the substr function.

The columns we are selecting month(using substring on the date), Landing_Outcome, Booster_Version, Launch_Site.

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

```
select "Landing_Outcome", count("Landing_Outcome") as "Count"
    from SPACEXTABLE
where Date between '2010-06-04' and '2017-03-20'
    group by "Landing_Outcome"
    order by "Count" desc;
```

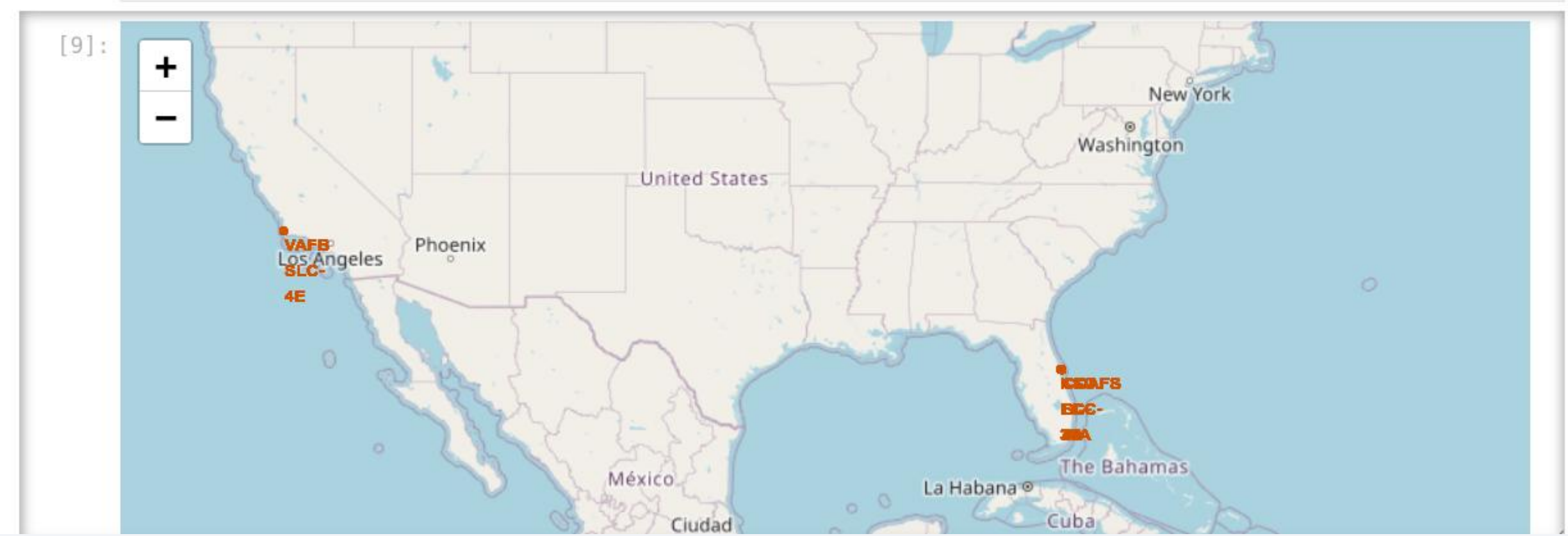
- In this query, we are using the columns Landing_Outcome, Count of the Landing_Outcome . To perform this we need to use the groupby on the landing_Outcome. We filter the data using the where clause which get only the data that contains Date between '2010-06-04' and '2017-03-20' . Finally we sort the data using the Count

Landing_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 Sites markers on a Folium Map.



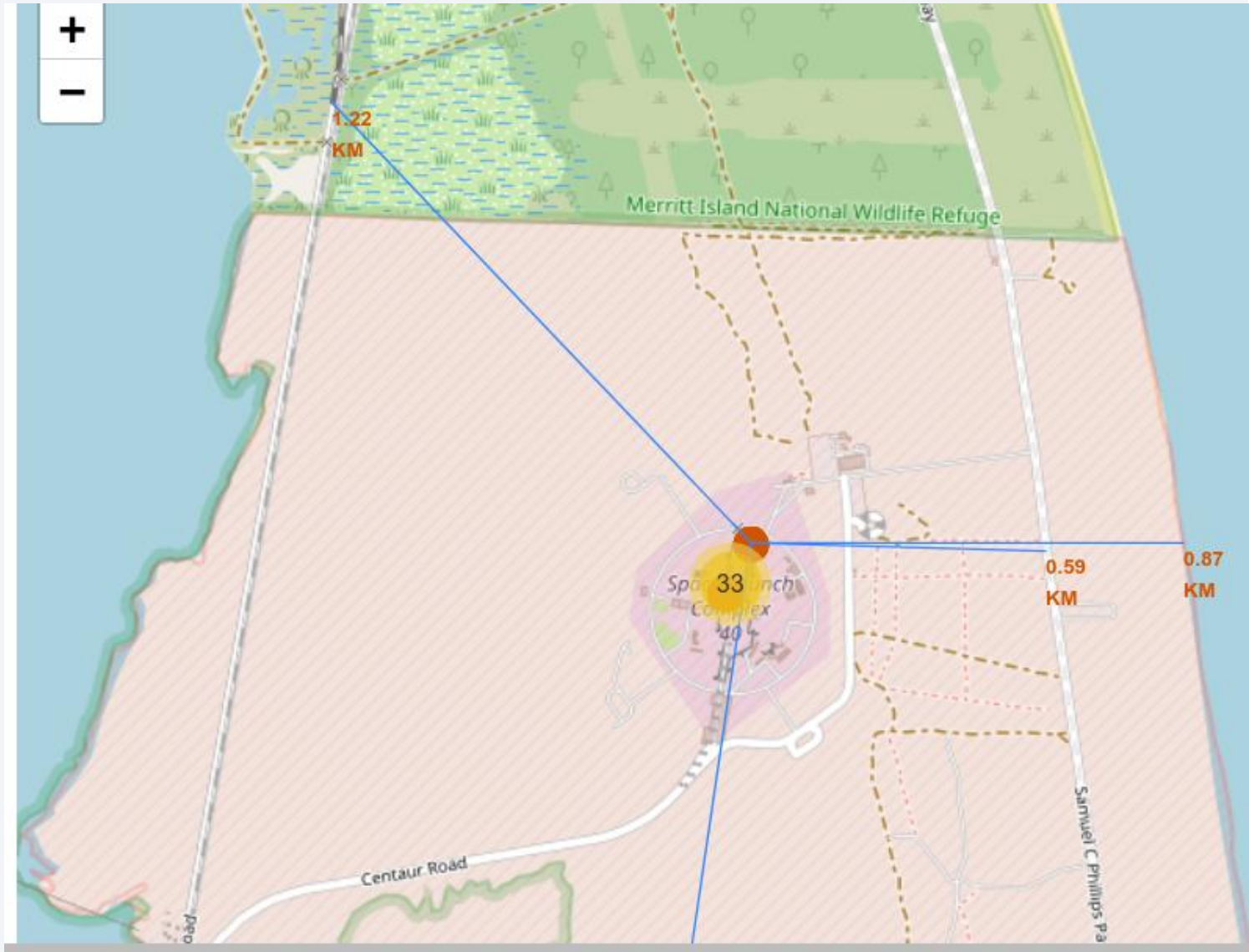
- We can see that the launch sites are near the costal area

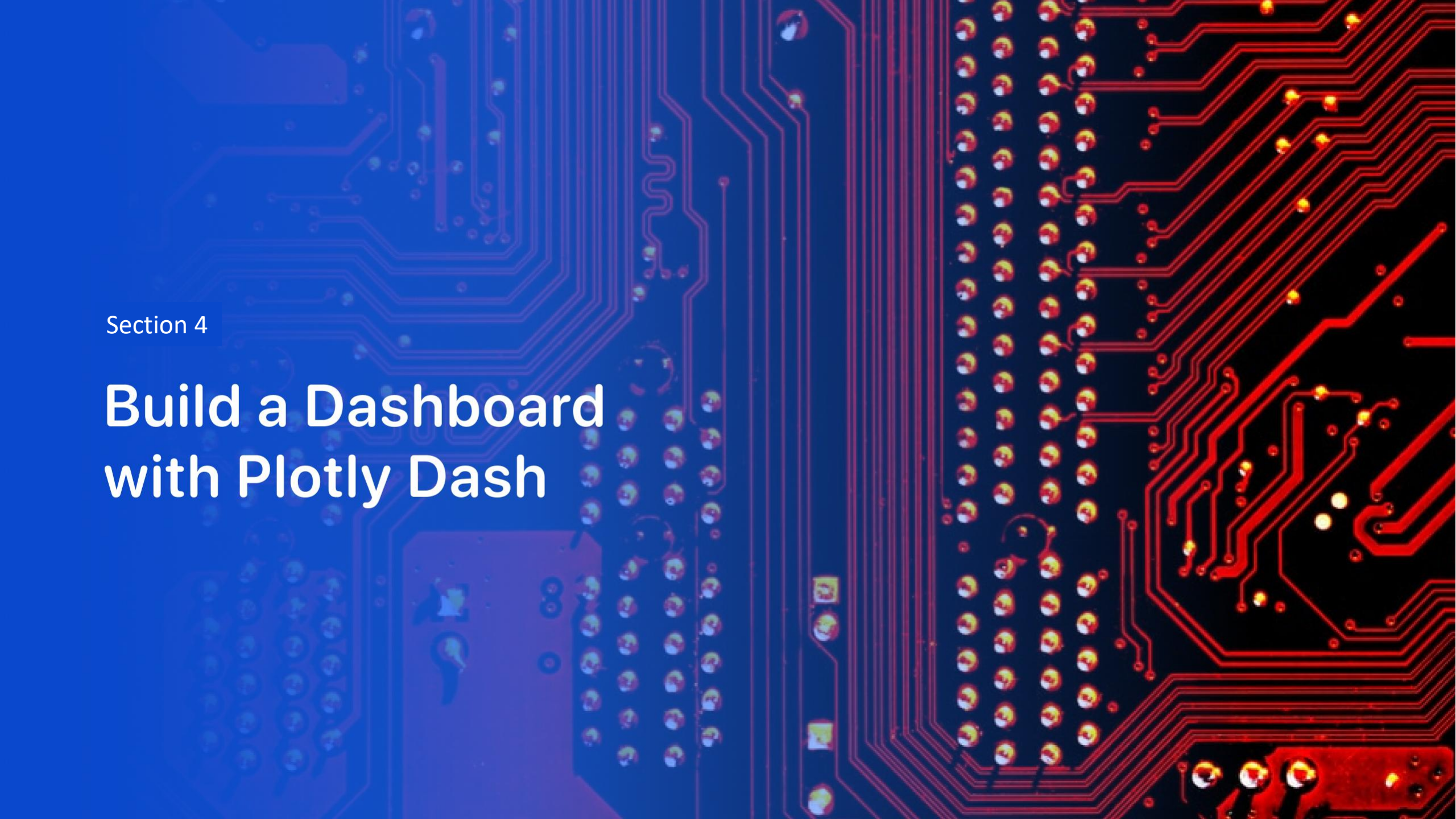
Folium Map with all Launch Sites with Marker Cluster

- The marker clusters helps to cluster the markers in an area and shows us how many markers are in an area depending on the zoom level.

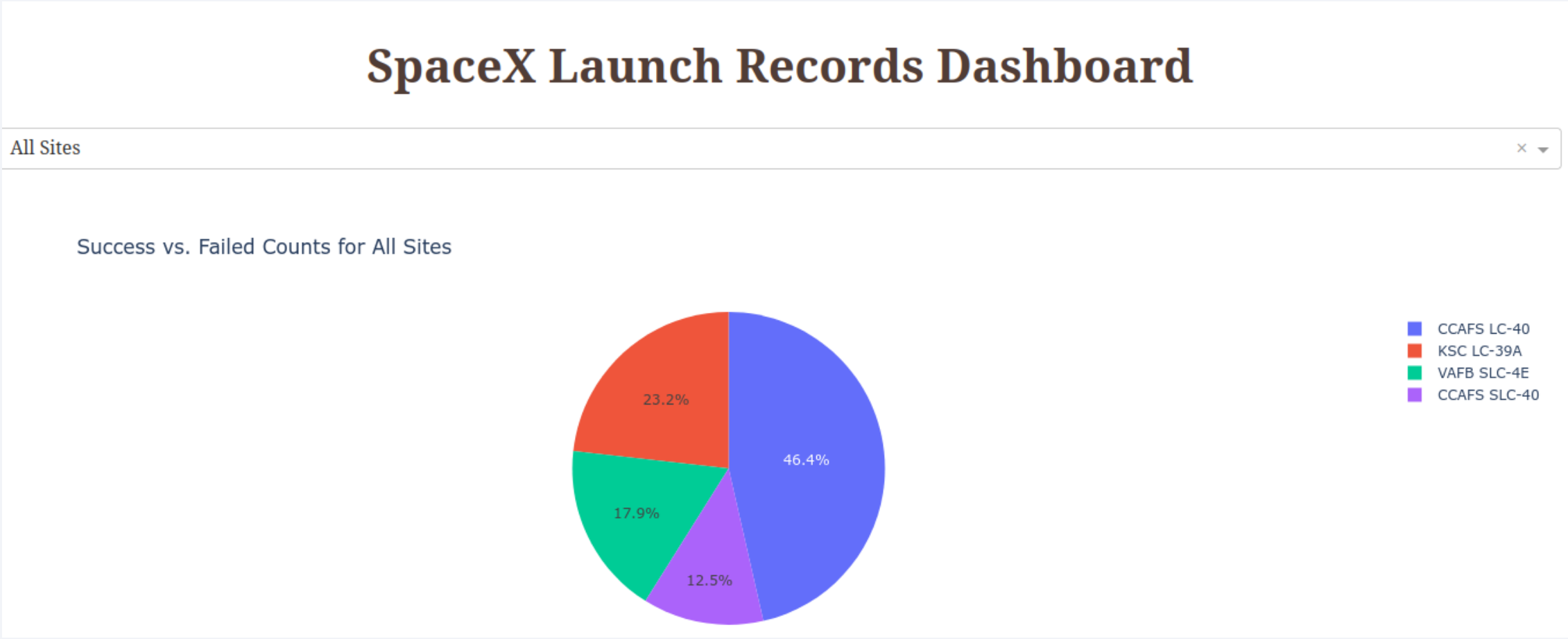
Marking distance of Launch Site from Highway, Railway Track, Coastline , City

- Along with the marker cluster, we placed markers for a Higway, Rail Track , Coast line, City by using specific latitude and longitude.
- We calculate distance between the marker and the launch site and use the distance as the marker text.
- Then we use polyLines to draw the lines from the launch site to each marker.
- We can see that The launch Site is close to roads, coastline, Railway but its far from the city.



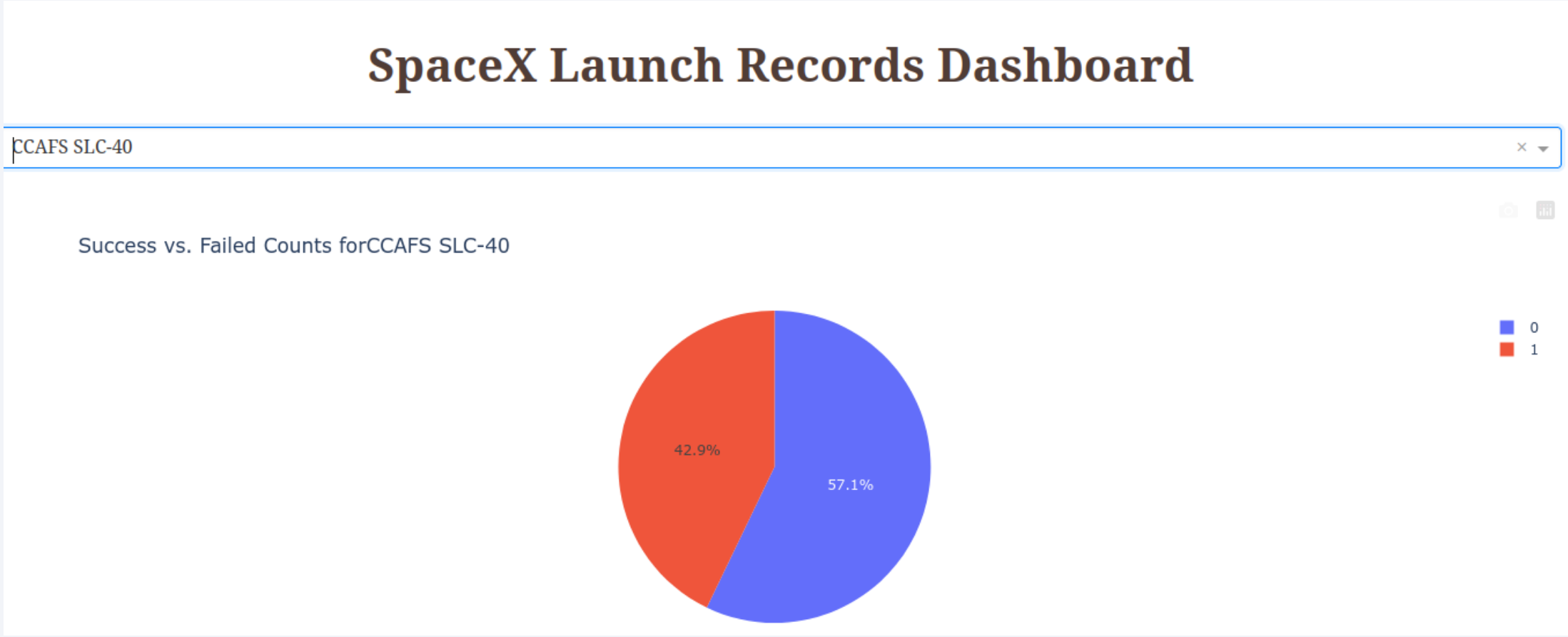

Section 4
Build a Dashboard with Plotly Dash

Dash App that shows the success rates of all the launch sites



- To create this, we first setup a dropdown in the layout of the app and a callback function that captures its inputs. then based on the input provided, we use plotly exppress to create the pie chart.
- From the pie char above CCAFS SLC-40 has the highest success rate among all launch Site with 46.4%

Pie chart that shows the Launch Site with the highest Success to failure ratio.



We can see that the to number of successful landing is almost the same as number of failures but there are more failures.

Scatter plot for Payload vs Success Rate

- To create this, we started with a RangeSlider component. Then we used the range as input for the calback function. We use this for filtering the Payload Mass. The range selected above is (0- 6000)
- We can see majority of the FT has class 1.

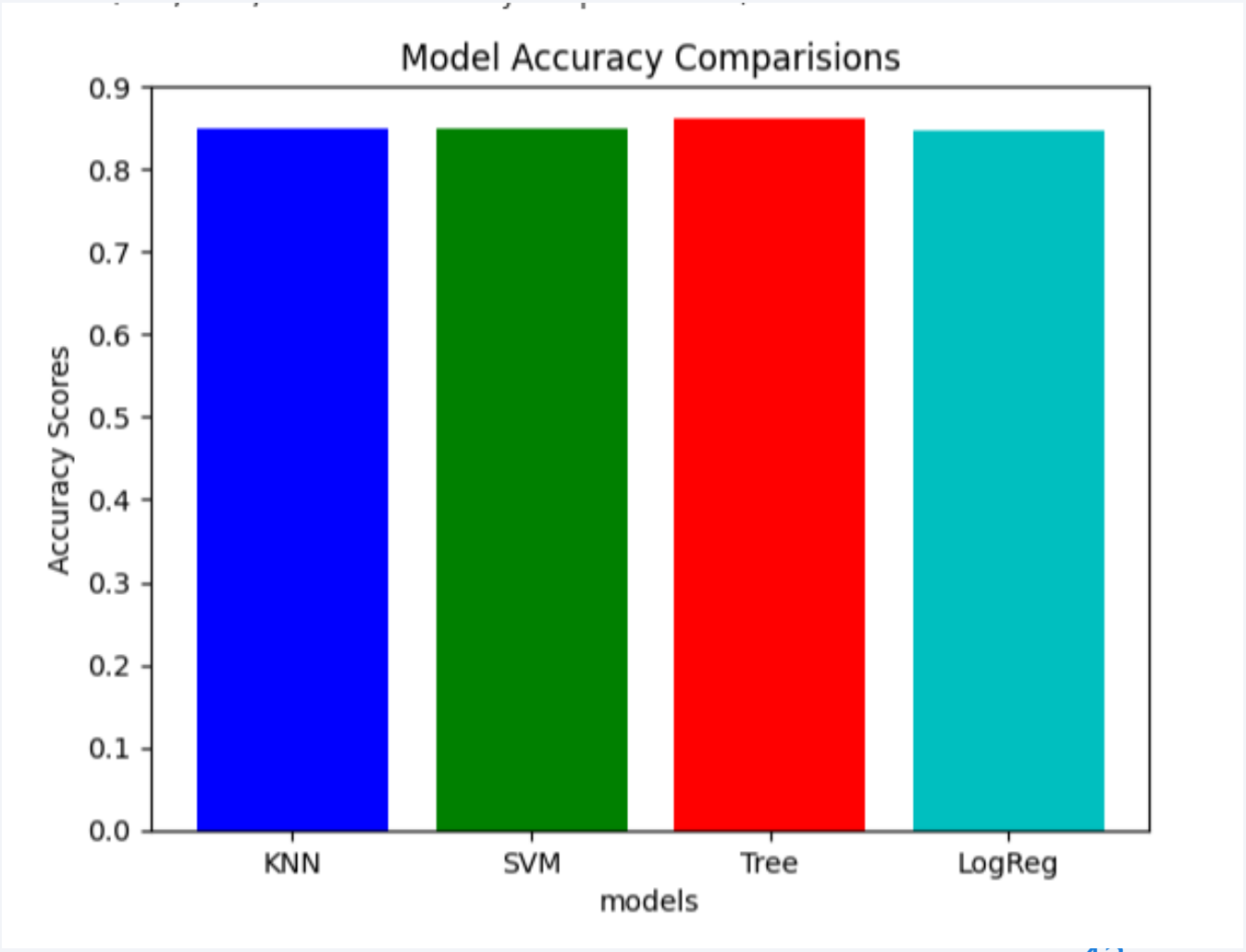
Section 5

Predictive Analysis (Classification)

Classification Accuracy

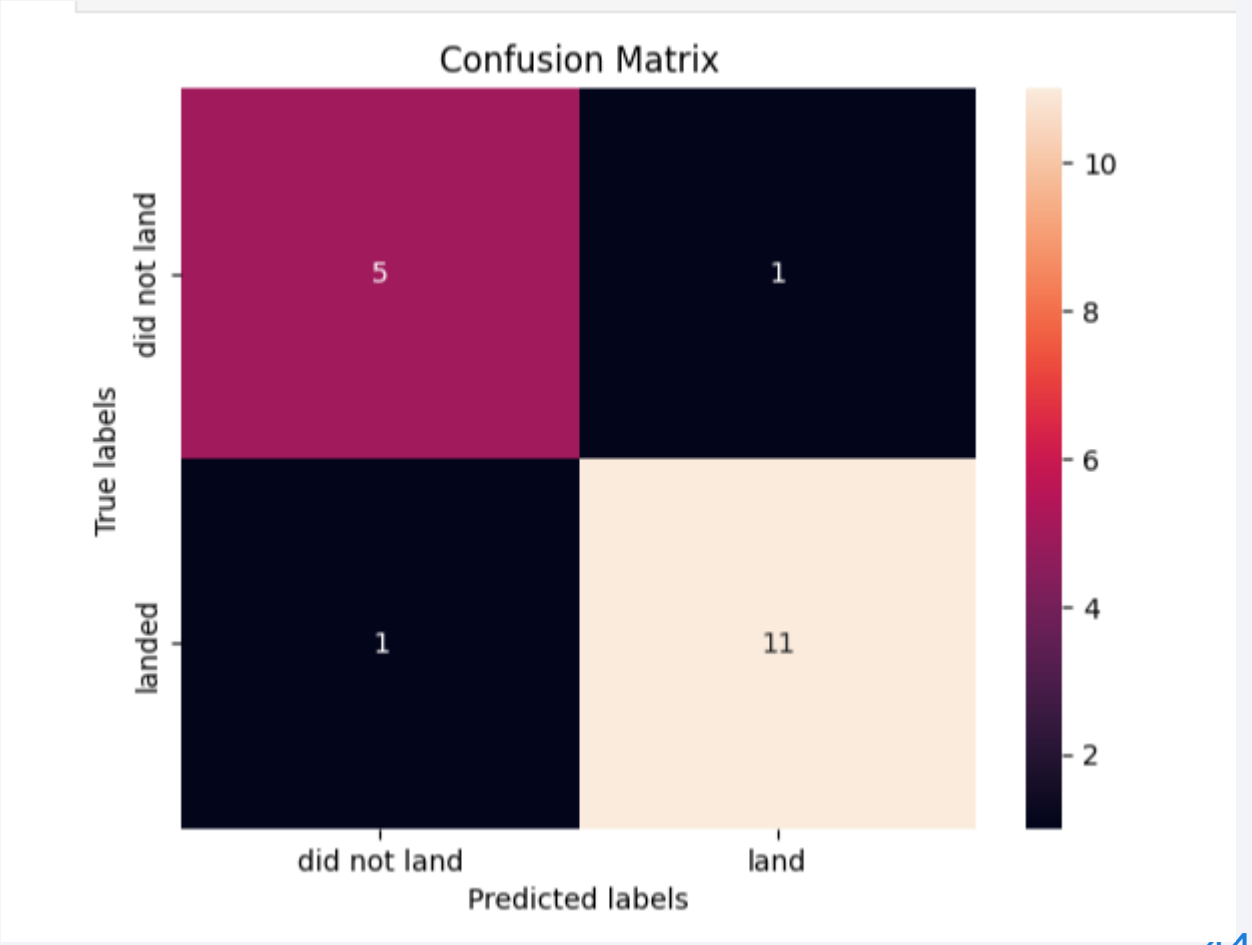
We built 4 models with GridSearch for parameter tuning and we plotted the bars for the accuracies.

We can see that the Decision Tree has the highest accuracy score.



Confusion Matrix

- This is the Confusion matrix for the Decision tree algorithm which has the best accuracy.
- We can see the model has 16 Correct predictions and 2 wrong predictions. which reflects in the accuracy score (0.86) which is slightly higher than other models.



Conclusions

- The first groud pad landing was on 2015-12-22.
- CCAFS SLC-40 has 46% success rate.
- CCAFS SLC-40 has higher Success Rate compared to the remaining Launch Sites.
- VAFB-SLC launchsite has no rockets launched for heavy payload mass(greater than 10000).
- ES-L1, SSO, HEO, GEO orbits have the highest success rates.
- Decision Tree is the best model for this problem.

Appendix

- Code Example for the training process:
- X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 2)
- parameters ={'C':[0.01,0.1,1], 'penalty':['l2'], 'solver':['lbfgs']}
- parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
- lr=LogisticRegression()
- logreg_cv = GridSearchCV(lr, cv = 10, param_grid = parameters)
- logreg_cv.fit(X_train, Y_train)
- print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
- print("accuracy :",logreg_cv.best_score_)

Thank you!