IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Nurdan Oktan
16/01/2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection API with web scraping
  - Data wrangling
  - Exploratory data analysis using SQL
  - Exploratory data analysis with data visualization
  - Interactive visual analytics using Folium and Plotly Dash
  - Machine learning prediction
- Summary of all results
  - Exploratory data analysis result
  - Interactive analytics in screenshots
  - Predictive analytics result

Introduction

- Project background and context:
  - SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. If we can determine if the first stage will land, we can determine the cost of a launch. For this purpose, we will use machine learning pipelines to predict if the first stage will land successfully.
- Problems you want to find answers:
  - The factors effecting the landing process.
  - The interaction between the features have important impact on the success rate of a successful landing.
  - Orbits have the most success.
  - The best machine learning algorithm for this task.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected SpaceX rocket launches data using SpaceX API, helper functions and web scraping from Wikipedia.
- Perform data wrangling:
  - Created landing outcomes using pandas library and converted the into training labels with “1” means the booster successfully landed, “0” means it was unsuccessful.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

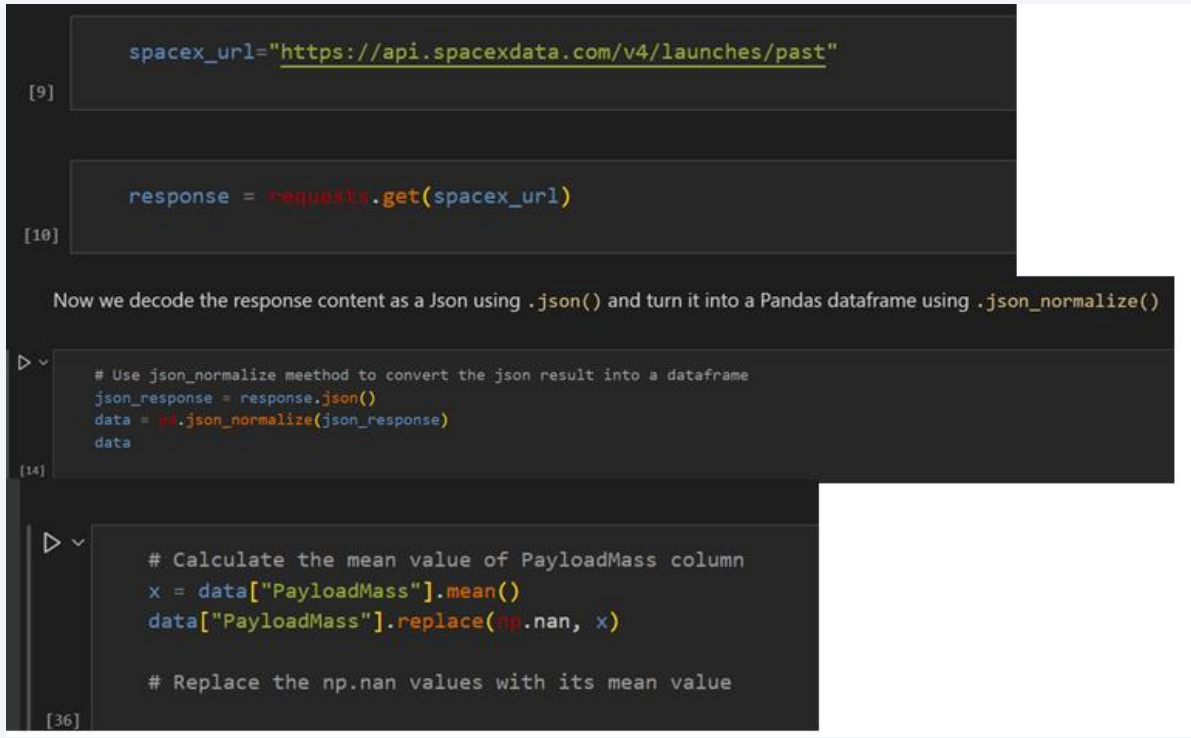
Data Collection

- Used get request to the SpaceX API.
- The response content was collected as a Json using json() function and turn it into pandas data frame using json_normalize() function.
- Cleaned the data and search missing values and fill in missing values if it is necessary.
- Web scraping from Wikipedia for Falcon 9 launch records using Beautiful Soup.
- The objective was to extract the launch records as HTML table, parse the table and convert it to pandas data frame for future analysis.

Data Collection – SpaceX API

- Get request function was used to get data, turned data into pandas data frame and did some necessary changes in data.

- https://github.com/lizedjo/master/blob/main/notebook_Data_Collection_API_HpqJpsIZS.ipynb



Data Collection - Scraping

- Did web scraping Falcon 9 launch records with Beautiful Soup library.
- Parsed the table and converted it into pandas data frame and extracted column name one by one.
- https://github.com/lizedjo/master/blob/main/notebook_Data_Collection_with_Web_Scraping_RD-i7nC.ipynb

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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

```
<Response [200]>
```

Create a BeautifulSoup object from the HTML response

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, "html.parser")
```

Print the page title to verify if the BeautifulSoup object was created properly

```
# Use soup.title attribute
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Next, we just need to iterate through the <th> elements and apply the provided extract_column_from_header() to extract column name one by one

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0 :
        column_names.append(name)
```

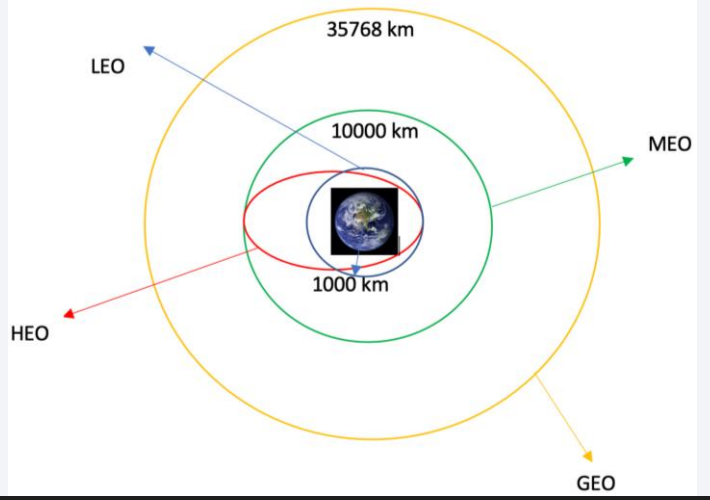
Check the extracted column names

```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

Data Wrangling

- Did exploratory data analysis to determine the training labels.
- Calculated the number of launches on each site.
- Calculated the number and occurrence of each orbit.
- Calculated the number and occurrence of mission outcome per orbit type.
- Created a landing outcome label from ‘Outcome’ column.
- Exported the results to csv.
- https://github.com/lizedjo/master/blob/main/EDA.ipynb



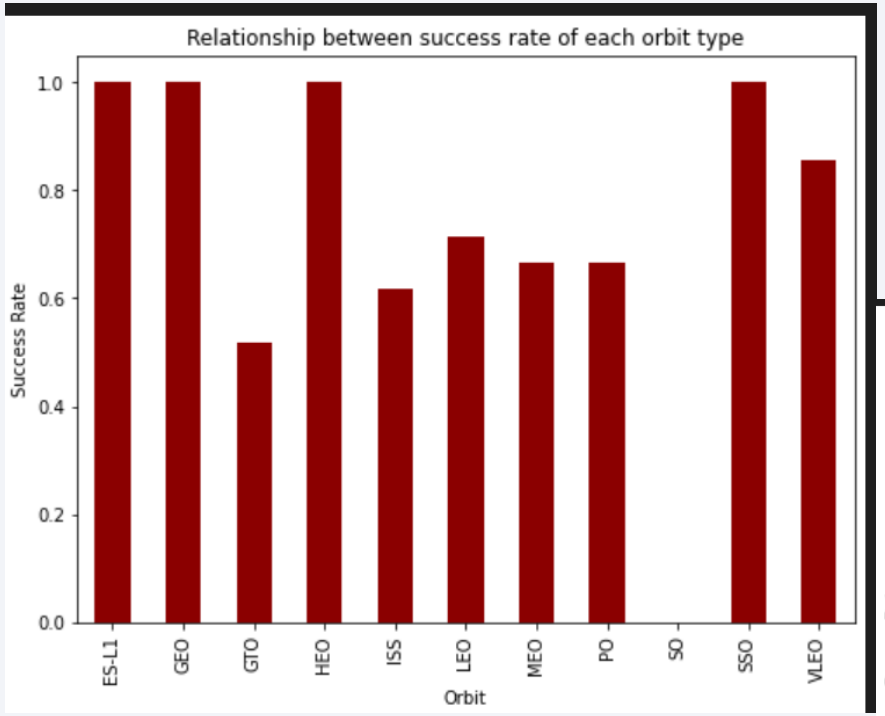
Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

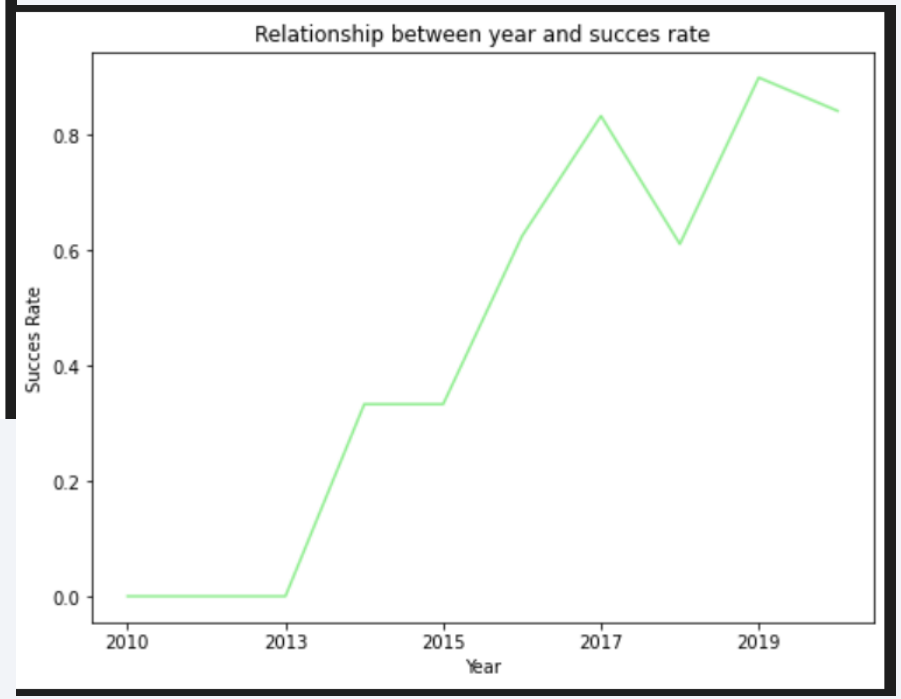
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for element in df["Outcome"]:
    if element in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
landing_class
```

```
Output exceeds the size limit. Open the full output data in a text editor
[0,
 0,
 0,
 0,
 0,
 0,
 1,
```

EDA with Data Visualization

- Visualized:
- Flight number and launch site,
- Payload and launch site,
- Success rate of each orbit type,
- Flight number and orbit type,
- The launch success yearly trend to see the relation between them.
- https://github.com/lizedjo/master/blob/main/EDA%20with%20Data%20Visualization.ipynb





EDA with SQL

- Wrote many SQL queries to understand the data, those queries provide the results of these:
  - The names of the unique launch sites in the space mission.
  - 5 records where launch sites start with the string “CCA”
  - The total payload mass carried by boosters launched by “NASA (CRS)”.
  - Average payload mass carried by booster version F9 v1.1.
  - The date when the first successful landing outcome in ground pad.
  - The names of the boosters which have success in drone ship and have payload mass between 4000 - 6000 kg.
  - The total number of “successful” and “failure” mission outcomes.
  - The names of the booster versions which have carried the maximum payload mass.
  - The records which will display the month names, failed landing outcomes in drone ship, booster versions, launch site for the months in year 2015.
  - Rank the count of successful landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order.
- https://github.com/lizedjo/master/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Got the latitude and longitude coordinates at each launch site and added a circle marker for each launch site with a name label to visualize all launch sites on interactive map.
- Then marked success/failed launches for each site on map and assigned them as 0 for failed or 1 for success launches.
- Added marker colors (red for 0, green for 1) for these classes using MarkerCluster().
- Calculated distances between a launch site to its proximities and found answer for these questions:
  - How close the launch sites to railways, highways and coastlines?
  - How close the launch sites to nearby cities?
- https://github.com/lizedjo/master/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

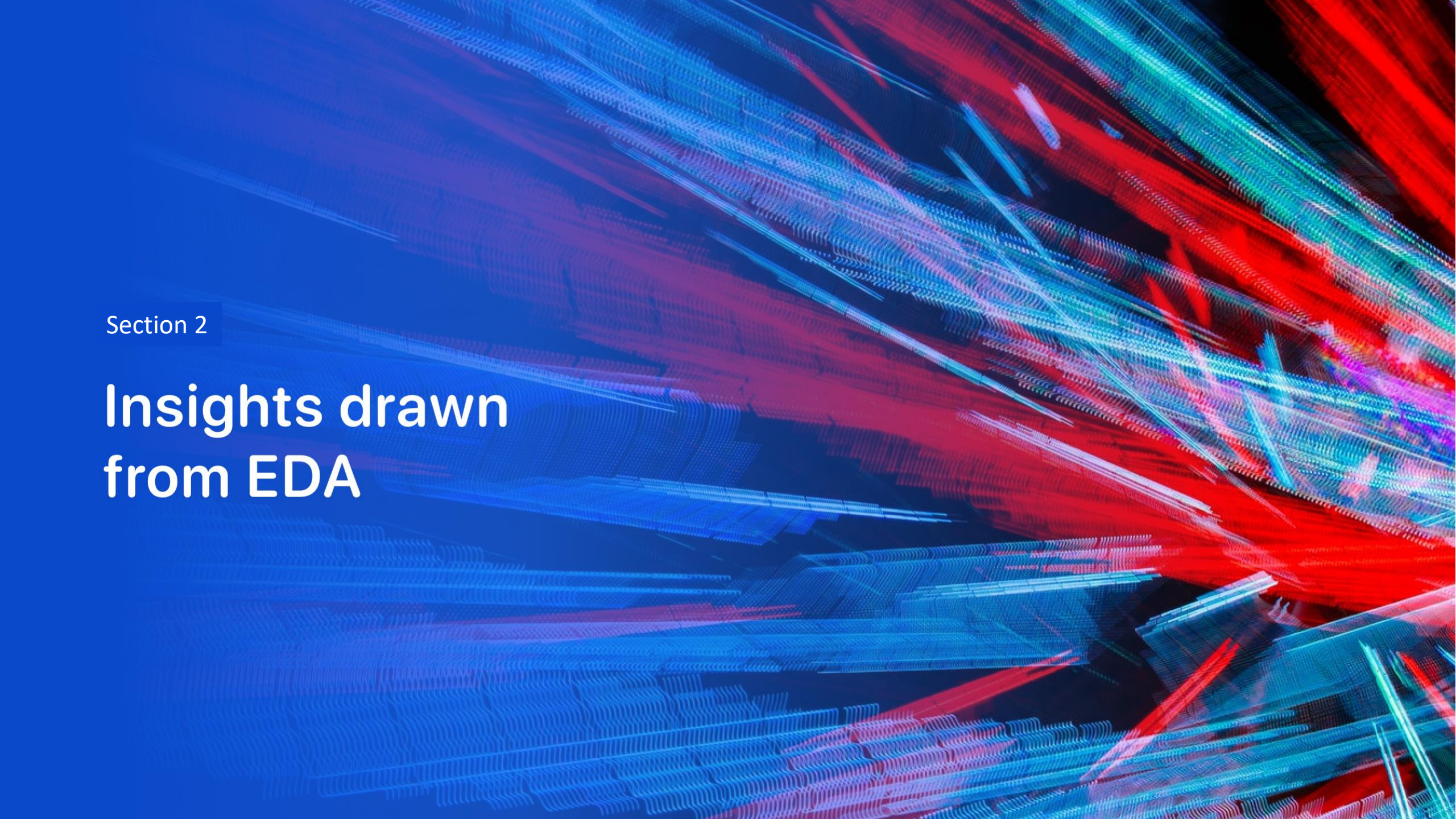
- Created an interactive dashboard with Plotly dash.
- Created pie charts showing the total launches by a certain site.
- We then plotted scatter graph showing the relationship with “Outcome” and “Payload Mass (kg)” for the different booster version.
- https://github.com/lizedjo/master/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Loaded data into a dataframe.
- Created numpy array using ‘Class’ column from the dataframe.
- Normalized data.
- Split data into train and test data.
- Created Logistic Regression, Support Vector Machine, Decision Tree and K-Nearest Neighbors classifier object then created a GridSearchCV object for each machine learning algorithm object and fit the object for each algorithm to find the best parameters.
- Calculated the accuracy for each machine learning algorithm to find the best algorithm for this project.
- https://github.com/lizedjo/master/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

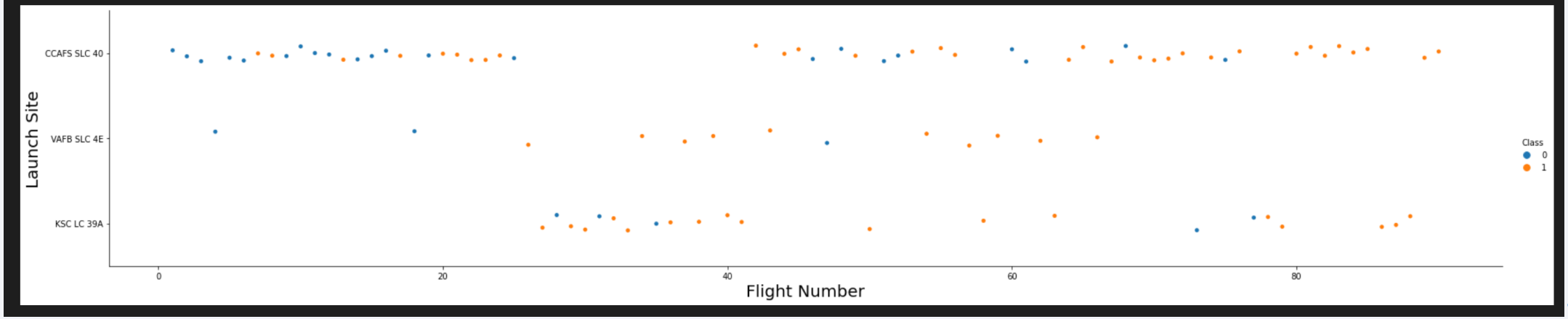
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

- This scatter plot shows that the higher flight number on launch site gives more success rate.

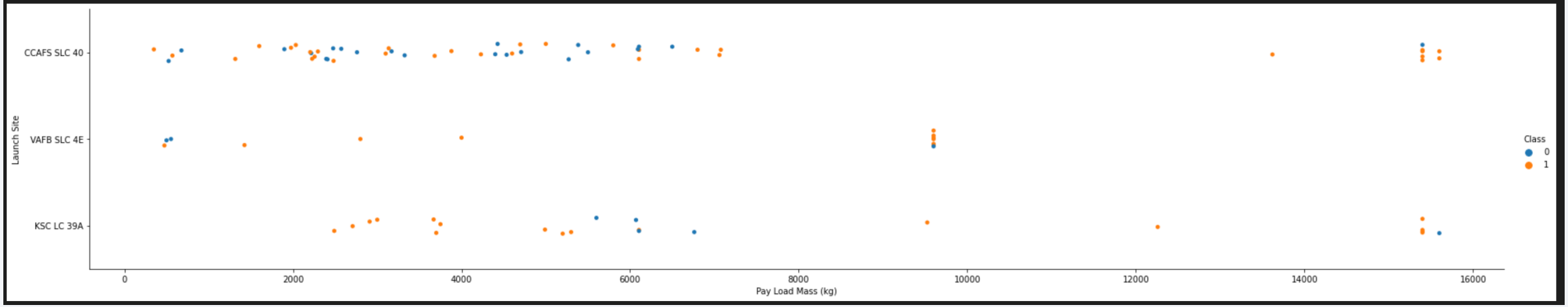
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y = 'LaunchSite', x="FlightNumber", hue="Class", data =df,aspect=5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel ("Launch Site", fontsize =20)

plt.show()
```



Payload vs. Launch Site

- Greater than 9000 kg pay load mass gives mostly high success rate. However, there is no a specific pattern in this graph.

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect=5)
plt.xlabel("Pay Load Mass (kg)")
plt.ylabel("Launch Site")
plt.show()
```

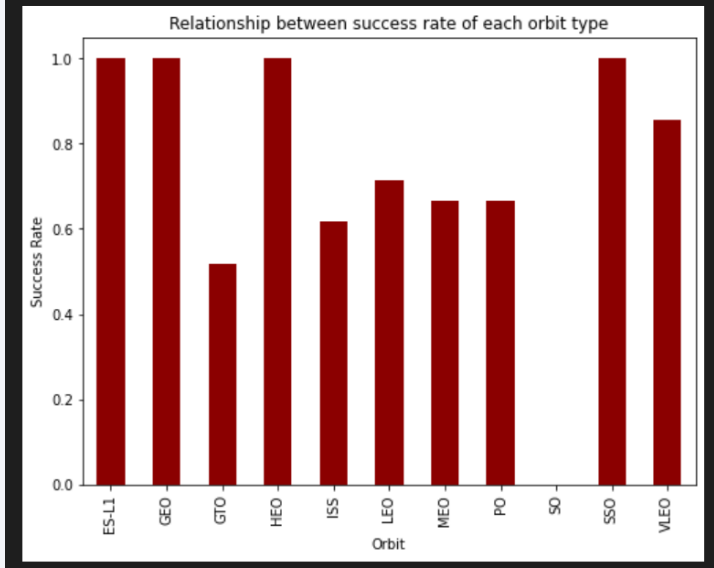


Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, SSO, VLEO orbits has higher success rate than 80% while other orbits are around 60%.
- SO orbit has the least success rate which is 0%.

```
# HINT use groupby method on Orbit column and get the mean of Class column
dfg = df.groupby(["Orbit"]).mean()
dfg_class = dfg["Class"]
dfg_class.plot(kind ="bar", title="Relationship between success rate of each orbit type", xlabel= "Orbit", ylabel= "Success Rate",  figsize=(8,6), color="darkred")
```

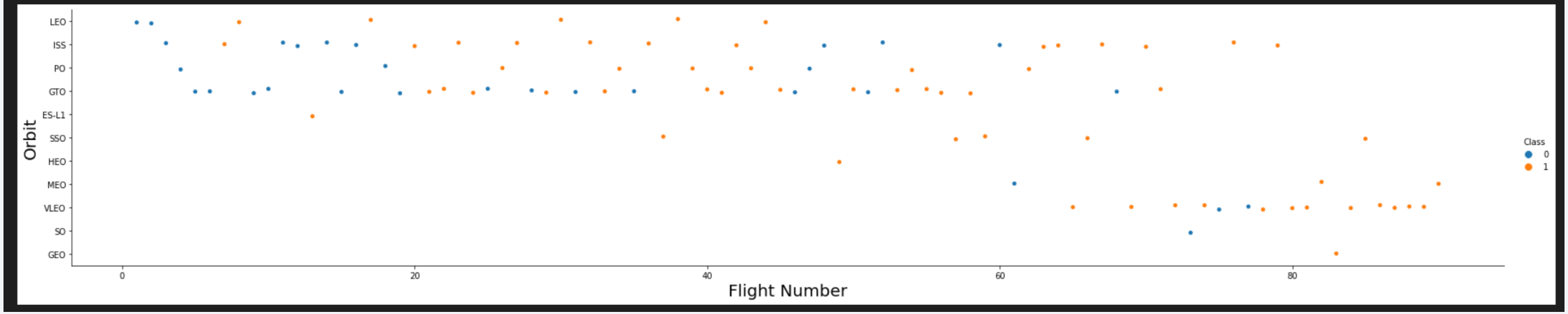
```
<AxesSubplot:title={'center':'Relationship between success rate of each orbit type'}, xlabel='Orbit', ylabel='Success Rate'>
```



Flight Number vs. Orbit Type

- This plot shows that the larger the flight number on each orbits, the greater the success rate especially for the orbit LEO.

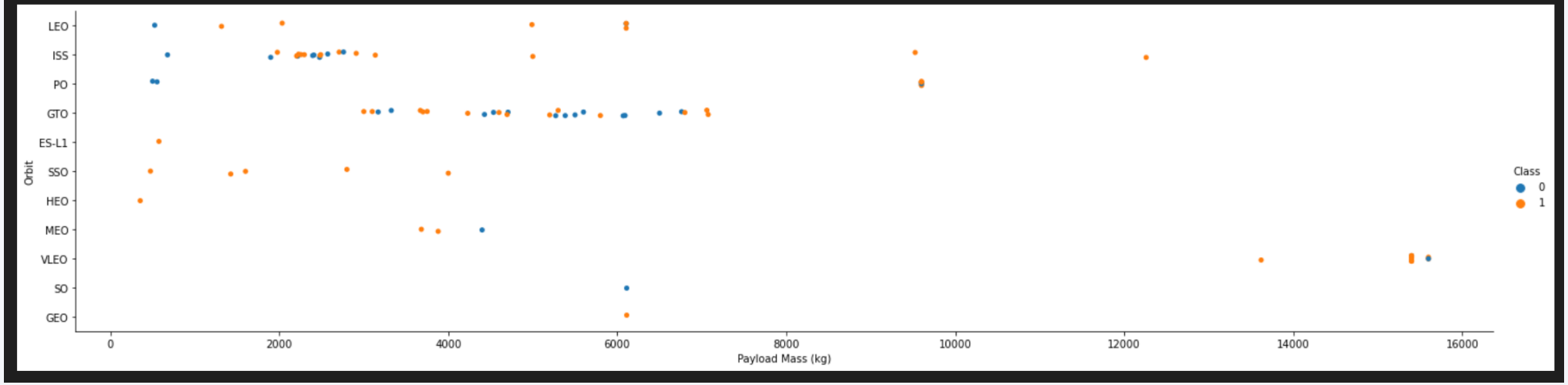
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Payload vs. Orbit Type

- Higher payload mass has positive impact on LEO, ISS and PO orbits. However, it has negative impact on MEO and VLEO orbit while GTO orbit shows no relation.

```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(x="PayloadMass", y="Orbit", hue="Class", data=df, aspect=4)
plt.xlabel("Payload Mass (kg)")
plt.ylabel("Orbit")
plt.show()
```

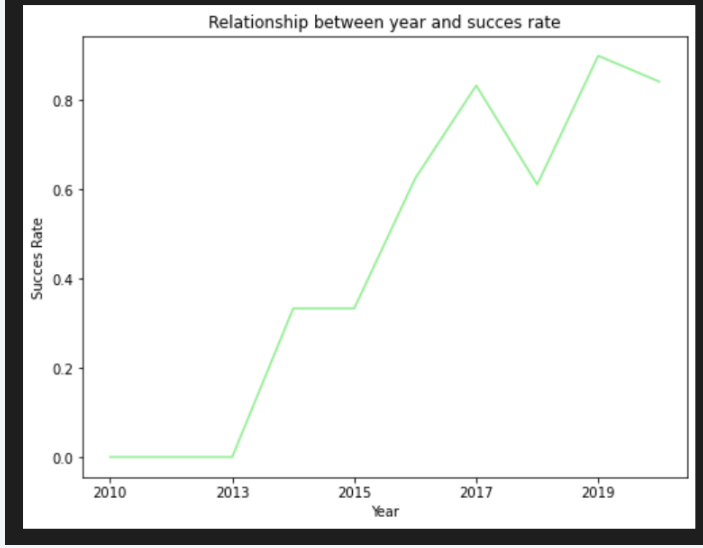


Launch Success Yearly Trend

- Success rate is showing mostly positive impact by year.

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
dfe = pd.DataFrame(Extract_year(df["Date"]))
df["Date"] = dfe
dfi = df.groupby(["Date"]).mean()

dfi_class = dfi["Class"]
dfi_class.plot(kind="line", xlabel="Year", ylabel="Succes Rate", figsize=(8,6), title="Relationship between year and succes rate", color="lightgreen")
```

```
<AxesSubplot:title={'center':'Relationship between year and succes rate'}, xlabel='Year', ylabel='Succes Rate'>
```



All Launch Site Names

```
%sql SELECT DISTINCT "LAUNCH_SITE" FROM SPACEXTBL
```

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

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

- Using DISTINCT in the query lets us to remove duplicated LAUNCH_SITE.

Launch Site Names Begin with 'CCA'

- “WHERE … LIKE …” filters launch sites starting with “CCA”. LIMIT 5 let us to display 5 records from that filter.

```
%sql SELECT * FROM SPACEXTBL WHERE "LAUNCH_SITE" LIKE '%CCA%' LIMIT 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

+ Code + Markdown

Total Payload Mass

```
%sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "CUSTOMER" = 'NASA (CRS)'
```

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

SUM("PAYLOAD_MASS__KG_")
45596

- The SUM() function is used to calculate total payload mass carried by boosters launched by “NASA (CRS)”.

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "BOOSTER_VERSION" LIKE '%F9 v1.1%'
```

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

AVG("PAYLOAD_MASS__KG_")
2534.6666666666665

- The AVG() function is used to calculate average payload mass where the booster version “F9 v1.1”.

First Successful Ground Landing Date

```
%sql SELECT MIN("DATE") FROM SPACEXTBL WHERE "Landing _Outcome" LIKE '%Success%'
```

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

MIN("DATE")
01-05-2017

- The MIN() function is used to find the date where the first landing outcome was successful.

Successful Drone Ship Landing with Payload between 4000 and 6000

- This query gives the booster version where the landing was successful and the payload mass is between 4000 - 6000 kg.

```
%sql SELECT "BOOSTER_VERSION" FROM SPACEXTBL WHERE "LANDING _OUTCOME" = 'Success (drone ship)' AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000;
```

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

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

Total Number of Successful and Failure Mission Outcomes

- The COUNT() function returns the number of records returned by a select query.
- Created 2 subqueries. The first one counts the number of “success” mission outcome, the other one counts the number of “failure” mission outcome.

```
%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
```

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

SUCCESS	FAILURE
100	1

Boosters Carried Maximum Payload

```
%sql SELECT DISTINCT "BOOSTER_VERSION" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") FROM SPACEXTBL)
```

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

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

- The MAX() function returns the largest value of the selected column.
- This query shows that the names of the booster versions which have carried the maximum payload mass.

2015 Launch Records

- This query shows that the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015.
- SQLLite doesn't support monthnames. Therefore, we used substr(Date, 4, 2) as month to get the months and substr(Date,7,4)= '2015' for year.

```
%sql SELECT substr("DATE", 4, 2) AS MONTH, "BOOSTER_VERSION", "LAUNCH_SITE" FROM SPACEXTBL WHERE "LANDING _OUTCOME" = 'Failure (drone ship)' and substr("DATE",7,4) = '2015'
```

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

MONTH	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

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

- This query returns landing outcomes and their count where mission was successful between the date 04-06-2010 and 20-03-2017.
- GROUP BY groups results by landing outcome
- ORDER BY… DESC is used to sort the data in descending order.

```
%sql SELECT "LANDING _OUTCOME", COUNT("LANDING _OUTCOME") FROM SPACEXTBL WHERE "DATE" >= '04-06-2010' and "DATE" <= '20-03-2017' and "LANDING _OUTCOME" LIKE '%Success%' GROUP BY "LANDING _OUTCOME" ORDER BY COUNT("LANDING _OUTCOME") DESC ;
```

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

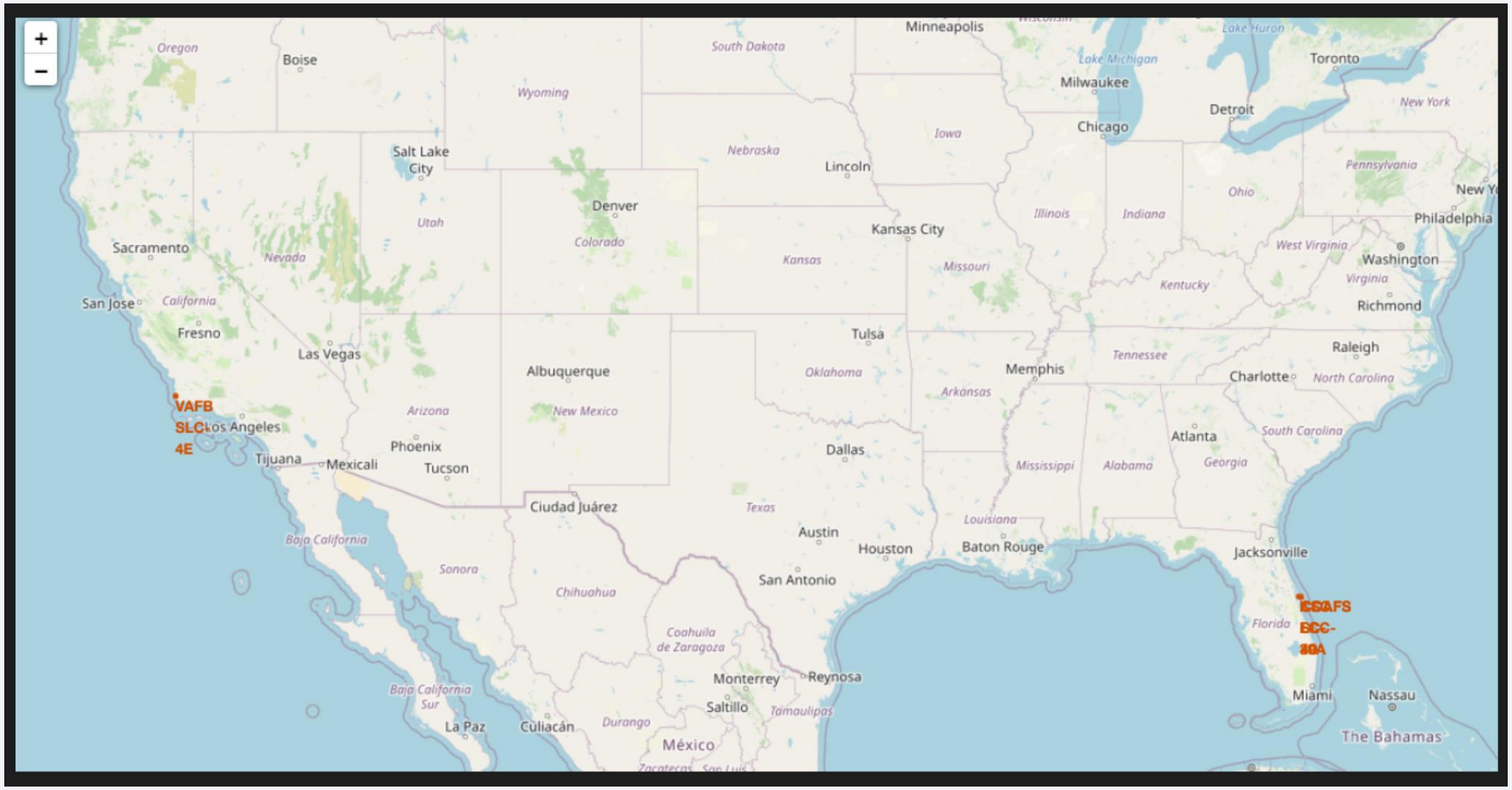
Landing _Outcome	COUNT("LANDING _OUTCOME")
Success	20
Success (drone ship)	8
Success (ground pad)	6

Section 3

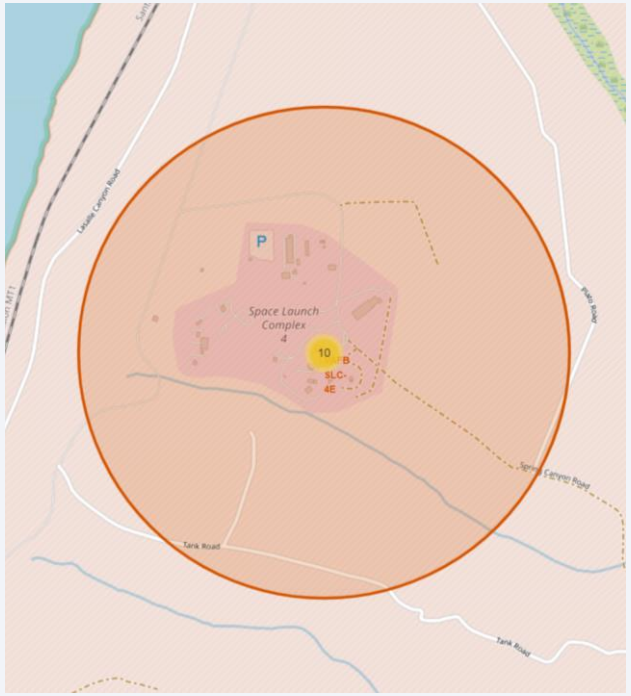
Launch Sites Proximities Analysis

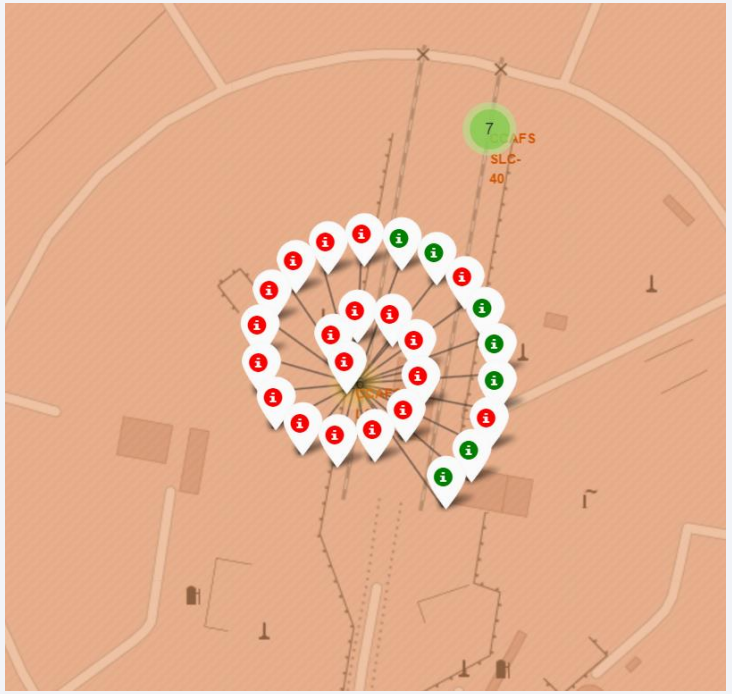
Location of the Launch Sites

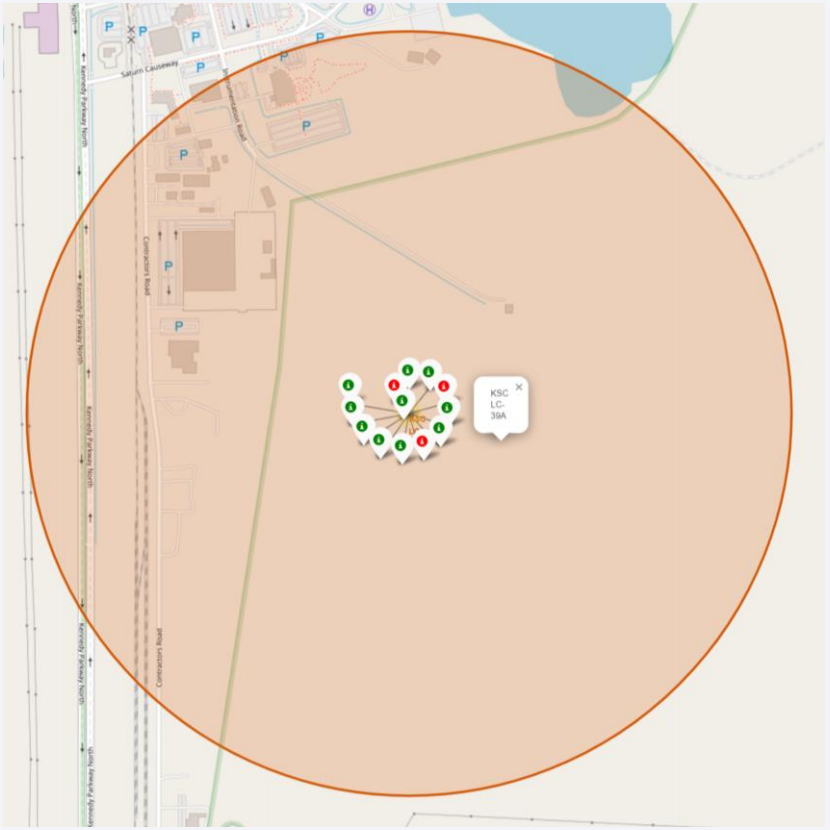
- See that all the SpaceX launch sites are located in the United States (California and Florida).



Markers Showing Launch Sites with Color Labels



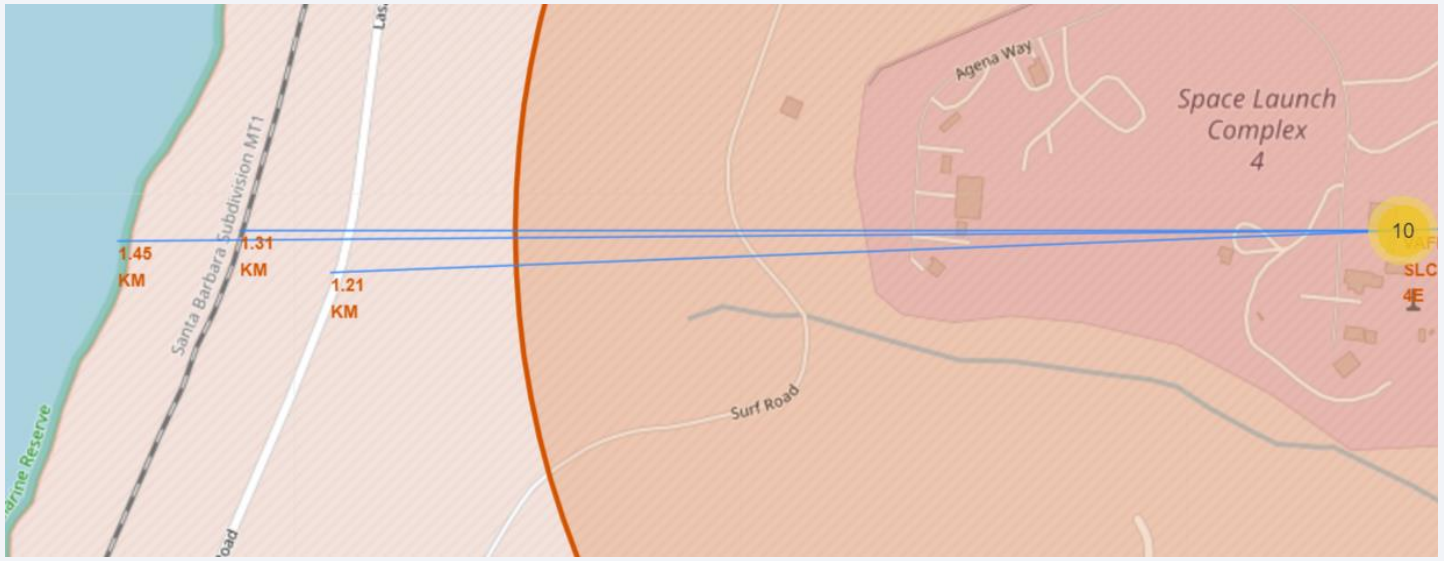




- Green marker represents successful launch, red marker represents failed launch.
- Thanks to folium library we can identify which launch sites have a good success rates.
- See that KSC LC-39A has the highest success rate.

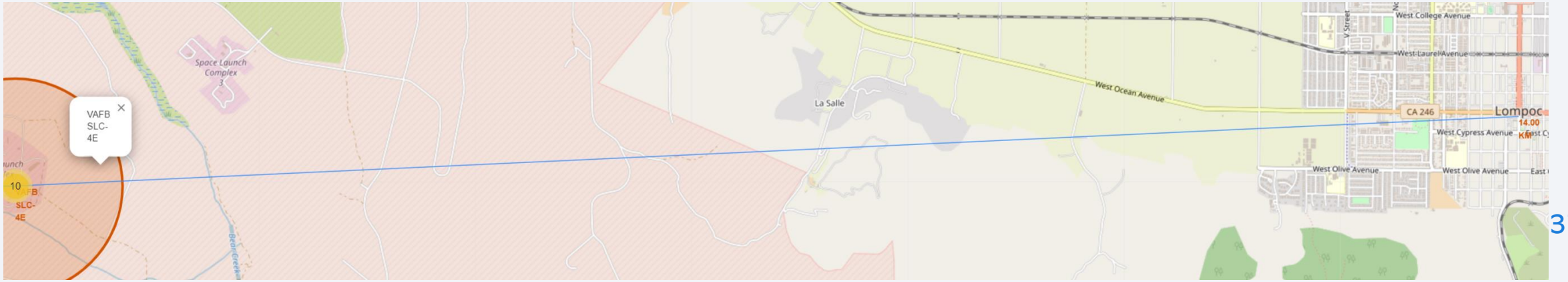
Distance the Launch Site VAFB SLC-4E and Its Proximities

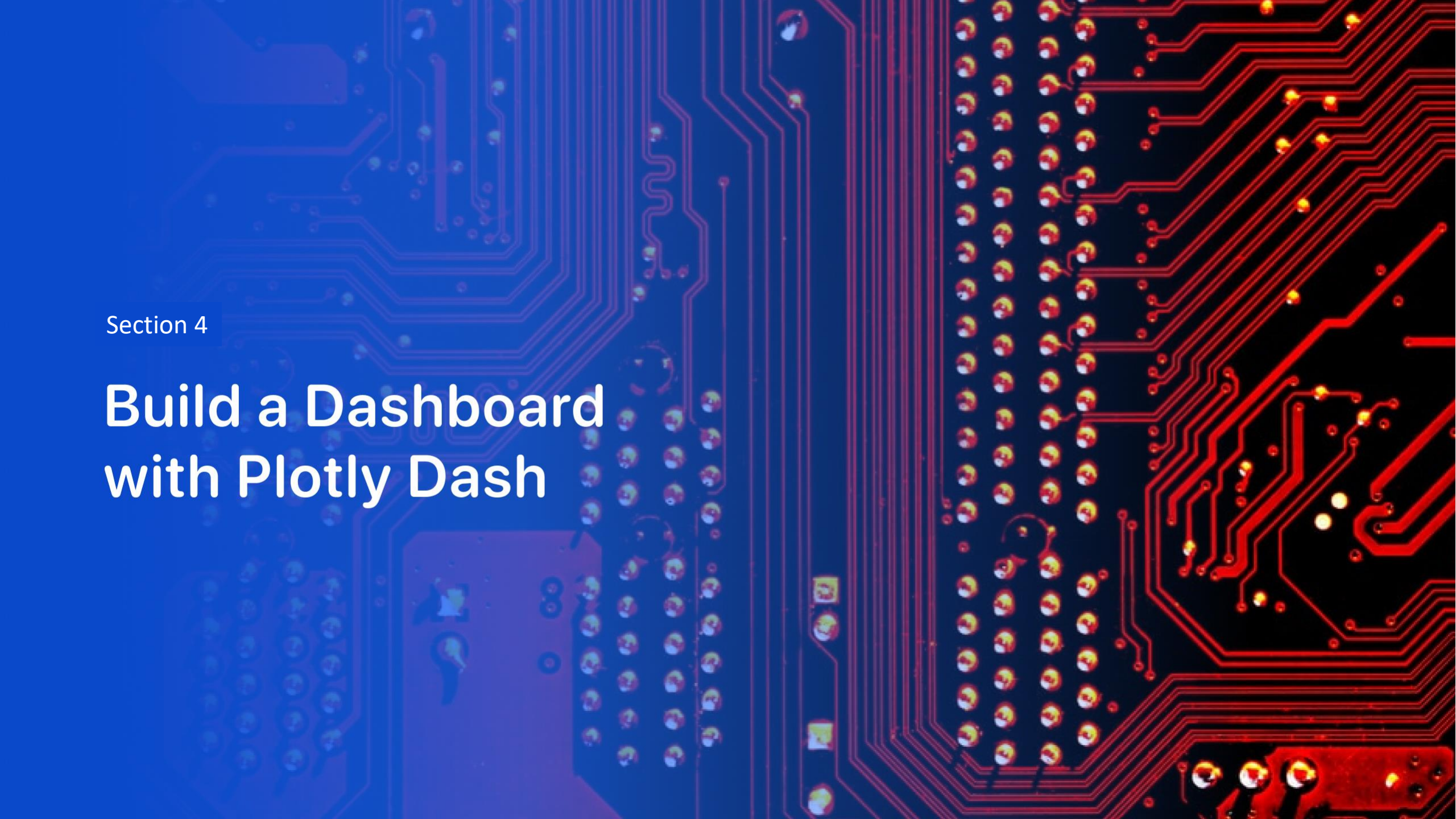
See how launch site close to coastline, railway and highway:



- We see that launch site VAFB SLC-4E is close to the coastline, the railway and the highway while it is far away from the city Lompoc.

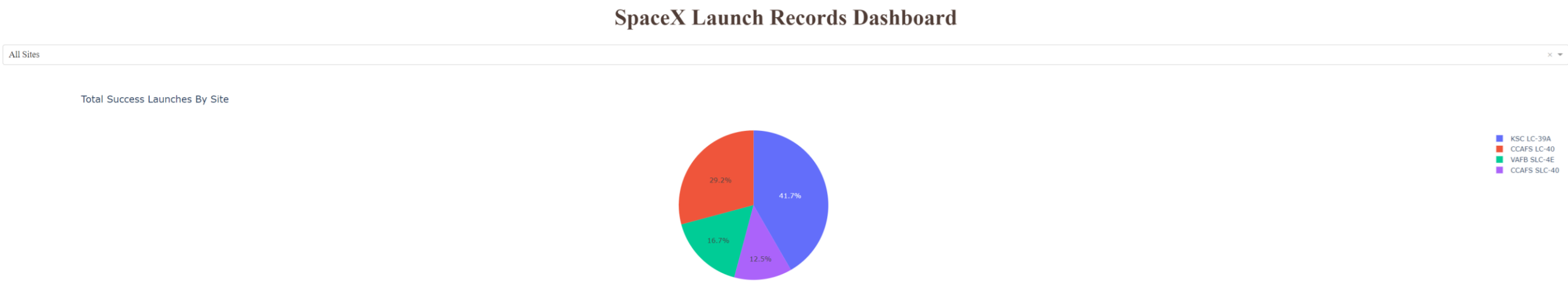
See how launch site close to the city:




Section 4
Build a Dashboard with Plotly Dash

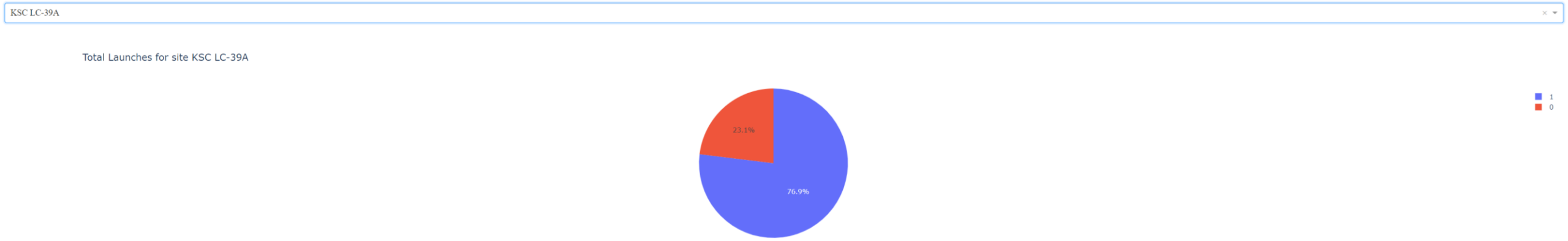
Total Success Launches by Site

- KSC LC-39A did the most successful launches while CCAFS SLC-40 did the least.



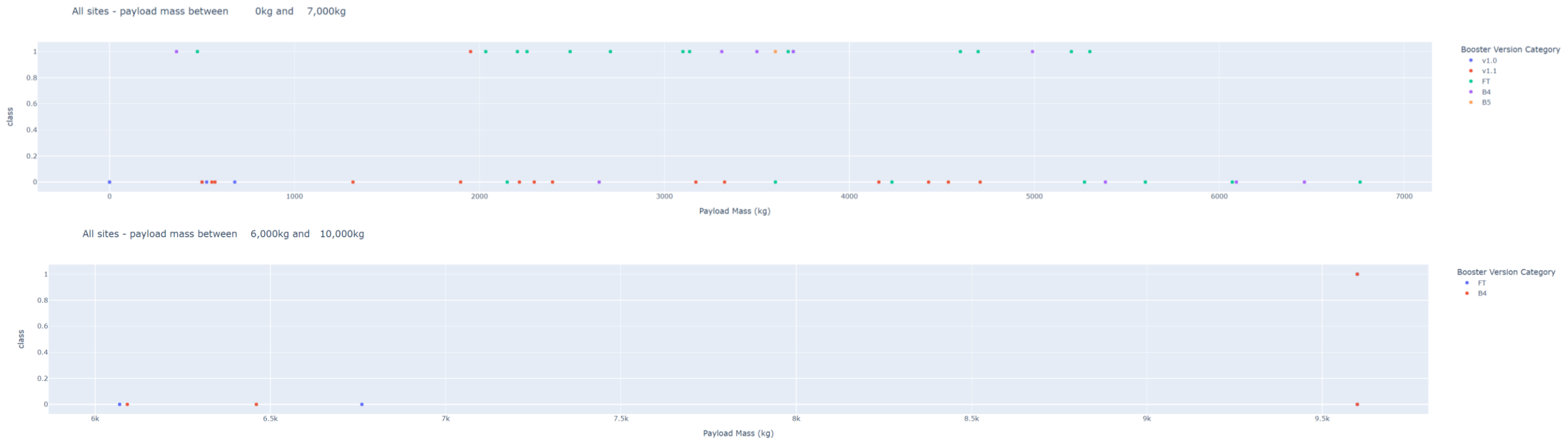
Success Percentage of KSC LC-39A

- This site had 76.9% successful launches and 23.1% failed launhces.



Payload vs. Launch Outcome

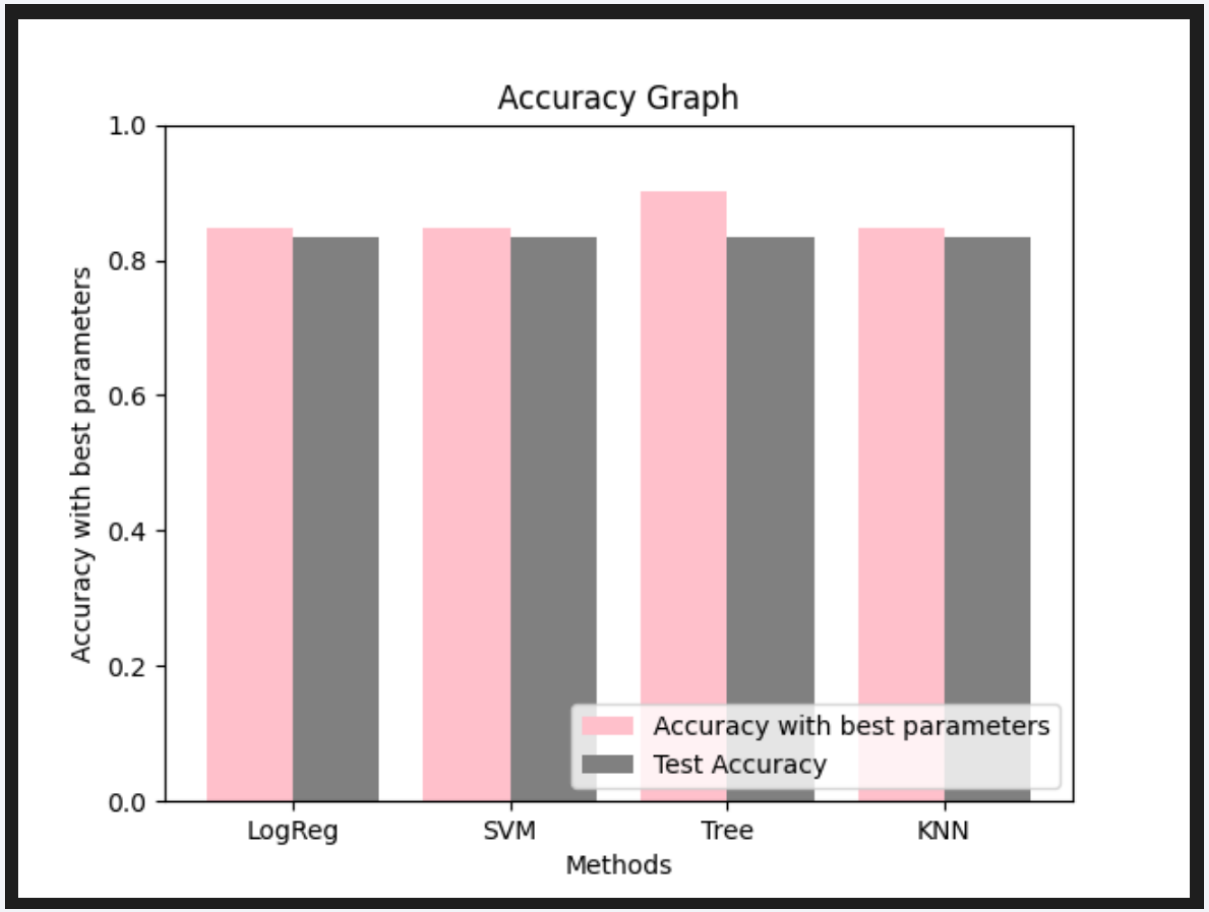
- Payloads under 6000 kg has more success. Especially payloads under 6000 kg with FT booster has a great success.



Section 5

Predictive Analysis (Classification)

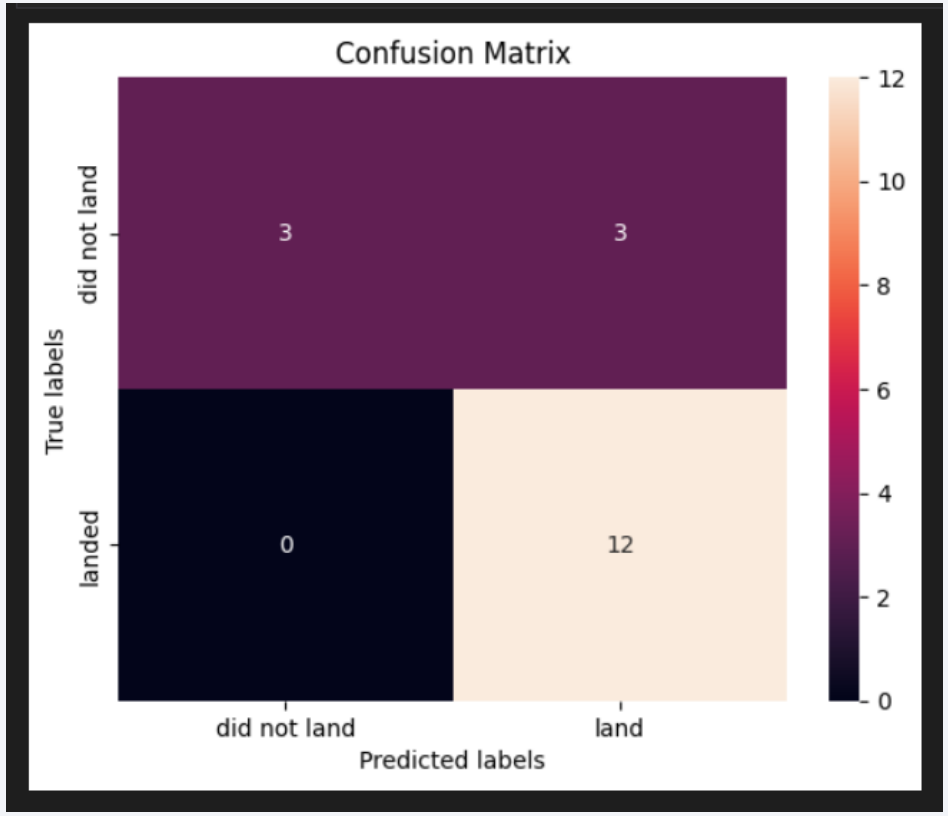
Classification Accuracy



- Test accuracy was same for all methods but the Decision Tree classifier has the best accuracy which is 0.9017 with best parameters.

Confusion Matrix

- Confusion matrix of the Decision Tree classifier is shown below.



- It was ‘land’ for the algorithm while it was ‘did not land’ in real. It means that the main problem for this algorithm was false positivies.

Conclusions

- A higher number of flights at launch site results in a greater success rate at launch site.
- Generally, low-weight payloads perform better than heavy ones.
- Launch success rate was stable between 2010 - 2013 then started to increase in 2013 until 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO has good success rates.
- KSC LC-39A is the most successful launch site.
- The accuracy showed that the Decision Tree classifier is the best algorithm for this specific task.

Appendix

- Confusion matrix is given below for reference.

		Predicted	
		Negative (**N**) **-**	Positive (**P**) **+**
Actual	Negative **-**	True Negatives (T**N**)	False Positives (F**P**) **Type I error**
	Positive **+**	False Negatives (F**N**) **Type II error**	True Positives (T**P**)

Thank you!