Winning Space Race with Data Science

Saranya SELVARADJOU
January 2023

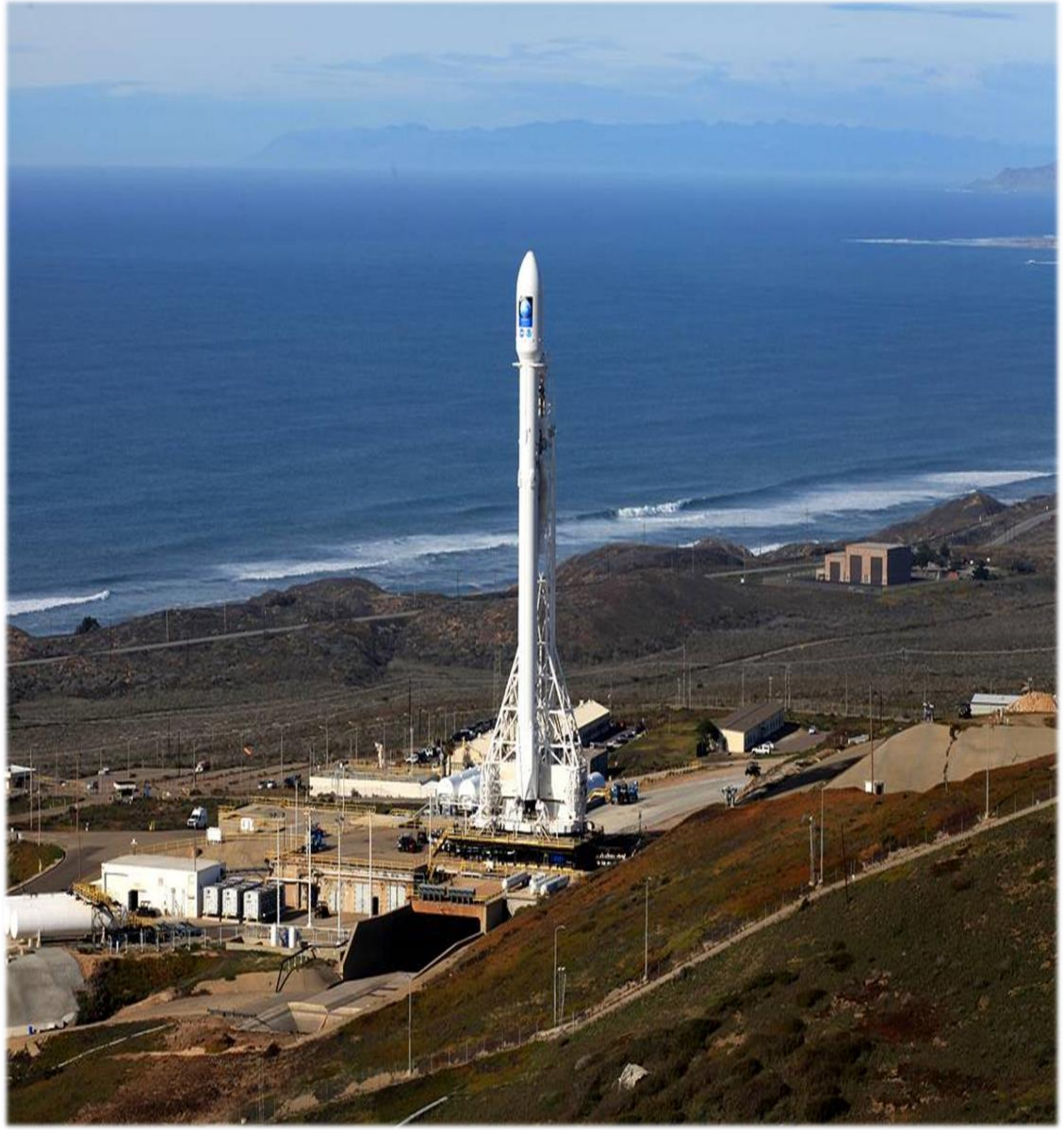
Outline

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

Executive Summary

- **Summary of Methodologies**
  - Data collection with API and webscraping
  - Data wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Map with Folium
  - Dashboard with Plotly Dash
  - Predictive Analysis (classification)

- **Summary of Results**
  - Exploratory Data Analysis results
  - Interactive Analytics results
  - Predictive Analysis results



Introduction

- **Project background and context**

SpaceX's Falcon 9 rocket has revolutionized the aerospace industry with its reusable first stage, which is designed to land back on Earth. SpaceX advertises Falcon 9 launches on its website at a cost of $62 million, which is significantly lower than the prices offered by other providers. Much of the cost savings for SpaceX comes from its ability to reuse the first stage of its rockets. We will predict the probability of successful landing for the first stage and provide valuable information to other companies that may want to compete with SpaceX in the rocket launch market.

- **Problems that need to be solved**

- What factors may affect the success or failure of landing attempts?
- What is the relationship of certain rocket variables on landing outcomes?
- What factors contribute to the highest success rate for landings?

Section 1

Methodology

Methodology

- Data Collection methodology :

  - Data was collected using SpaceX Rest API and web scraping wikipedia

- Perform Data Wrangling

  - One hot encoding was used to encode categorical variable

  - Dropped irrelevant columns

- Perform EDA using visualization and SQL

  - Matplotlib, seaborn were used to create plots and graphs that allow to see trends and patterns in the data

  - SQL was used to query the data and extract subsets of data for further analysis

- Perform interactive visual analytics

  - Folium and Plotly Dash visualization

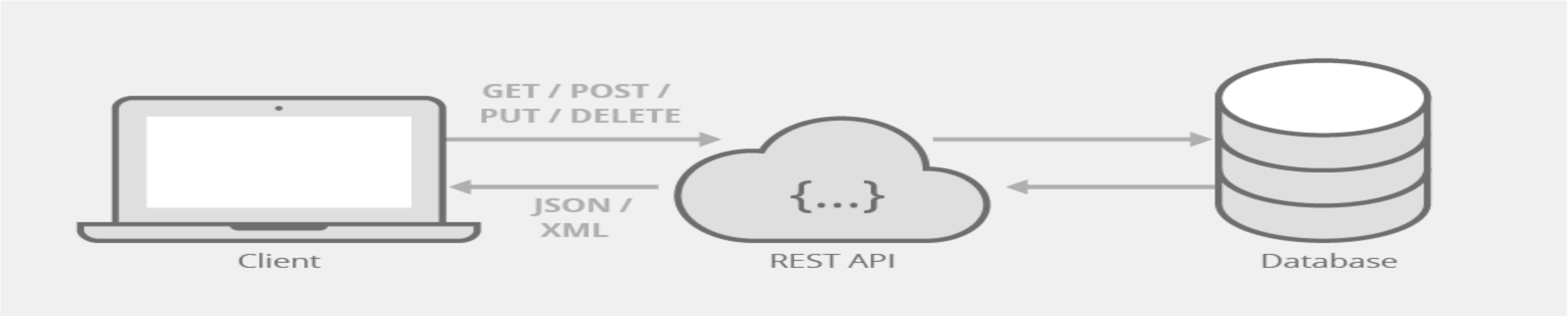
- Perform predictive analysis using classification model

  - Build, tune and evaluate classification models

Data Collection - Rest API and webscraping

- ## Rest API

The data collection process involved making GET requests to the SpaceX API and decoding the response content as a JSON object using the .json() function. The resulting data was then converted into a pandas dataframe using .json_normalize(). In order to ensure the data was clean and complete, we checked for missing values and filled in any missing data as needed.



- ## Webscraping

We used web scraping techniques with BeautifulSoup to extract launch records for the Falcon 9 rocket from Wikipedia and convert them into a pandas dataframe for further analysis.

Data Collection - SpaceX API

1.Getting response from API

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

```
response = requests.get(spacex_url)
```

2.Converting response to a .json file

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

3.Apply custom functions to clean Data

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

4.Assign list to dictionary then DataFrame

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

```
# Create a data from launch_dict
data = pd.DataFrame(launch_dict)
```

5.Filter DataFrame and export to flat file (csv)

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```


Github : https://github.com/SaraSS1/IBM-Data-Science-Professional-Certificate/blob/main/10 Applied Data Science Capstone/1 spacex-data-collection-api.ipynb

Data collection - Webscraping

1.Retrieving response from HTML

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

```
requests.get(static_url)
# assign the response to a object
response = requests.get(static_url).text
```

2.Parse HTML using BeautifulSoup

```
soup = BeautifulSoup(response, 'html.parser')
```

3.Searching for tables in HTML

```
html_tables = soup.findAll("table")
```

4.Extracting the column names from the table

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table

temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

5.Creating a dictionary to store the data

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6.Adding data to appropriate keys in the dictionary

7.Converting dictionary to a pandas dataFrame

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df.head()
```

8.Exporting the dataframe to a CSV file

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

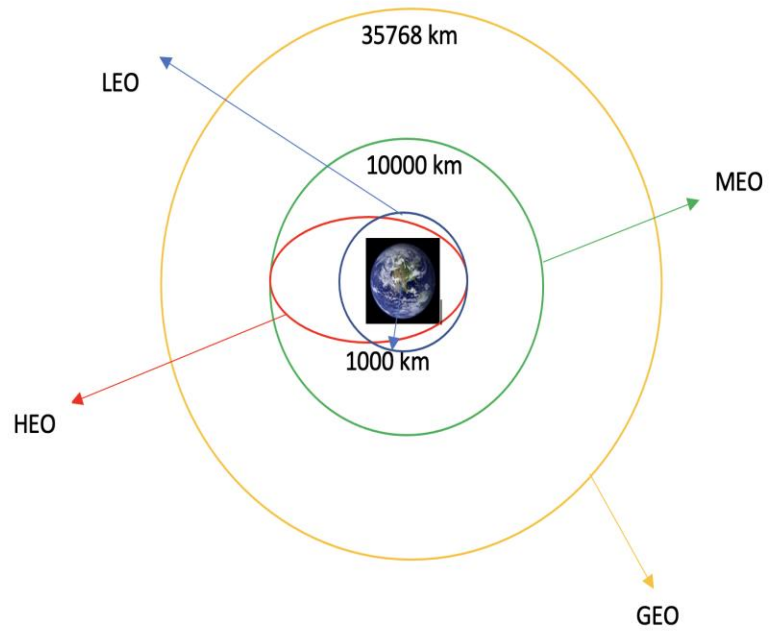
Github : https://github.com/SaraSS1/IBM-Data-Science-Professional-Certificate/blob/main/10 Applied Data Science Capstone/2 webscraping.ipynb

Data Wrangling

To effectively analyze and visualize SpaceX data, it is necessary to clean, transform and manipulate the data. The following steps are part of this process:

- **Calculate the number of launches at each site.** This helps us understand the distribution of launches across different locations.
- **Determine the frequency of each orbit type**. This allows us to comprehend the types of orbits that are most commonly used by SpaceX.
- **Calculate the number and frequency of mission outcomes by orbit type.** This helps us identify the success rate of missions for different orbit types.
- **Save the cleaned and transformed data in a CSV file**, which is a common format for storing and sharing data.
- **Create a new landing outcome label from the outcome column**.
- **Calculate the success rate for every landing in the dataset using cleaned data.** This can help us understand the overall success rate of SpaceX landings.

Each launch aims to a dedicated orbit, and here are some common orbit types



Github : https://github.com/SaraSS1/IBM-Data-Science-Professional-Certificate/blob/main/10 Applied Data Science Capstone/3 Data wrangling.ipynb

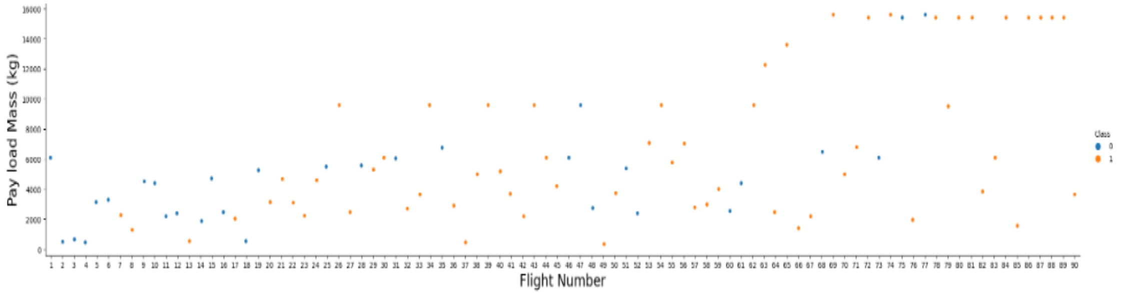
Exploratory Data Analysis with SQL

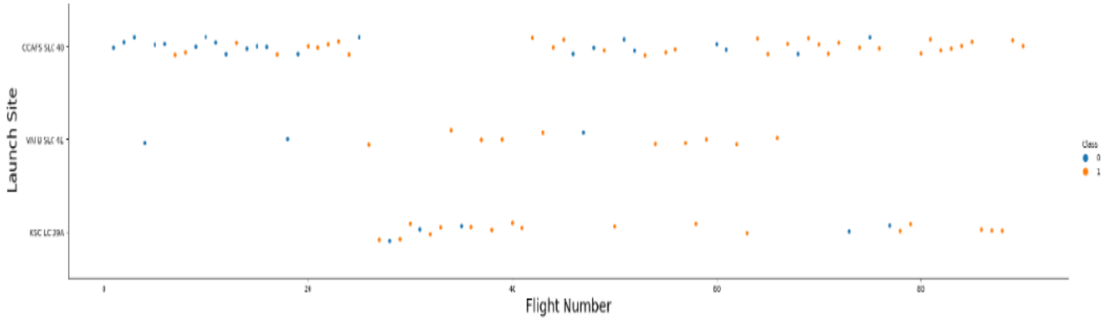
We used SQL queries to gather information from the dataset and gain a deeper understanding of it. The specific queries that we performed were:

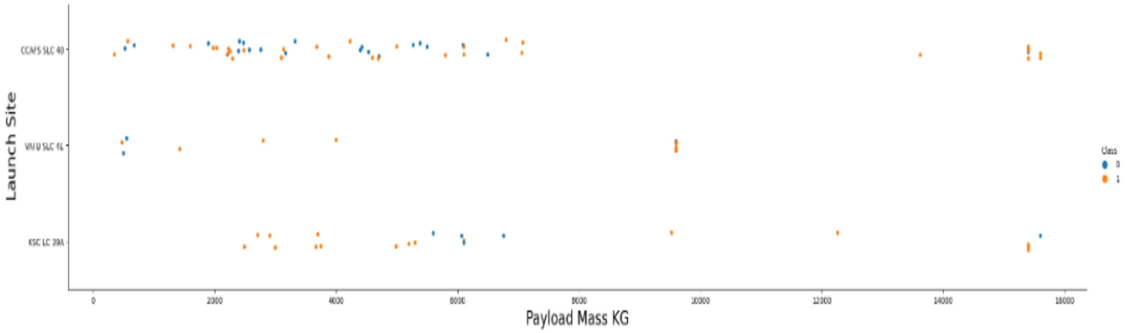
- Displaying the names of the unique launch sites in the space mission.
- Display 5 records where launch sites begin with the string ‘CCA’.
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9v1.1
- Listing the date where the successful landing outcome in drone ship was achieved
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000.
- Listing the total number of successful and failure mission outcomes.
- Listing the names of the booster_versions which have carried the maximum payload mass.
- Listing the failed landing_outcomes in drone ship, their booster versions, and launch site names for the year 2015.
- Ranking the count of landing outcomes (Such as Failure drone ship or Success ground pad between the date 04-06-2010 annd 20-03-2017 in descending order).

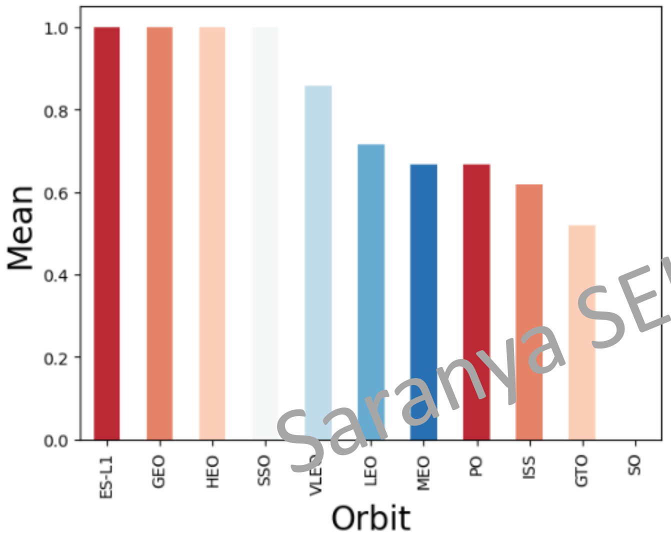
Github : https://github.com/SaraSS1/IBM-Data-Science-Professional-Certificate/blob/main/10 Applied Data Science Capstone/4 EDA sql-coursera_sqllite.ipynb

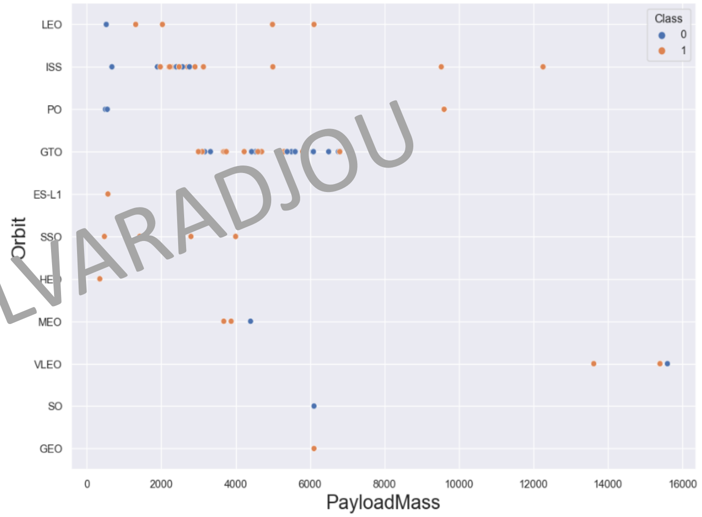
EDA with Data Visualization

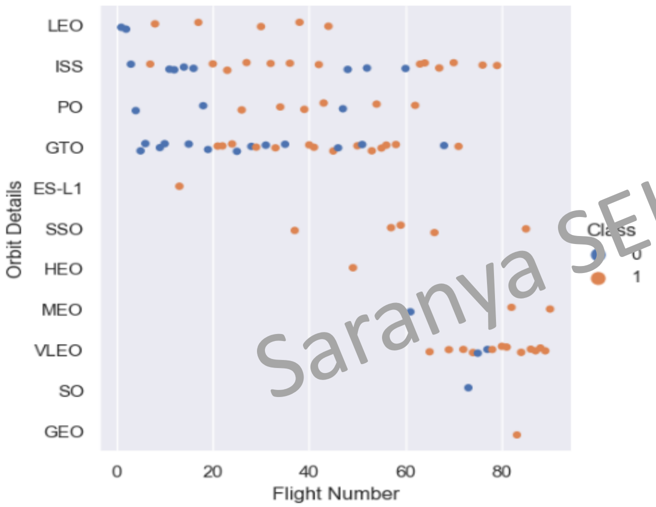


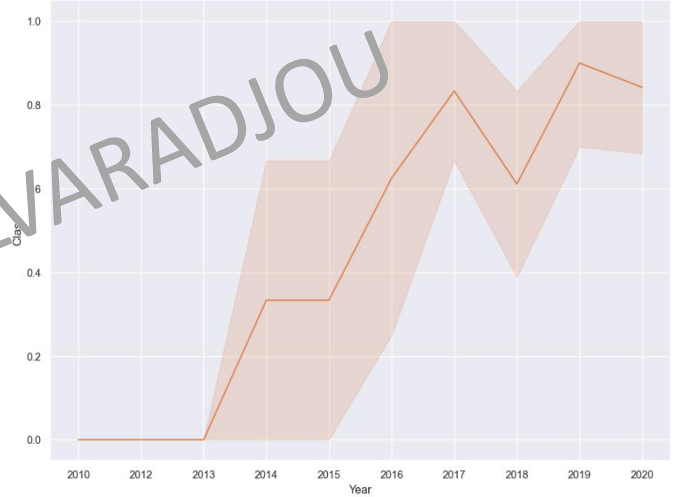












Github : https://github.com/SaraSS1/IBM-Data-Science-Professional-Certificate/blob/main/10 Applied Data Science Capstone/5 eda-dataviz.ipynb

Build an interactive map with Folium

Folium is a Python library that allows users to create interactive maps using the Leaflet JavaScript library. With Folium, users can visualize data that has been processed in Python on an interactive map in a web browser. This can be useful for exploring and understanding geographical data, and for communicating that data to others. We used the latitude and longitude coordinates for each launch site and added a circle marker around each site, labeled with the name of the site. We also used a marker cluster to group and display the launch outcomes (failures and successes) as green and red markers on the map, respectively.

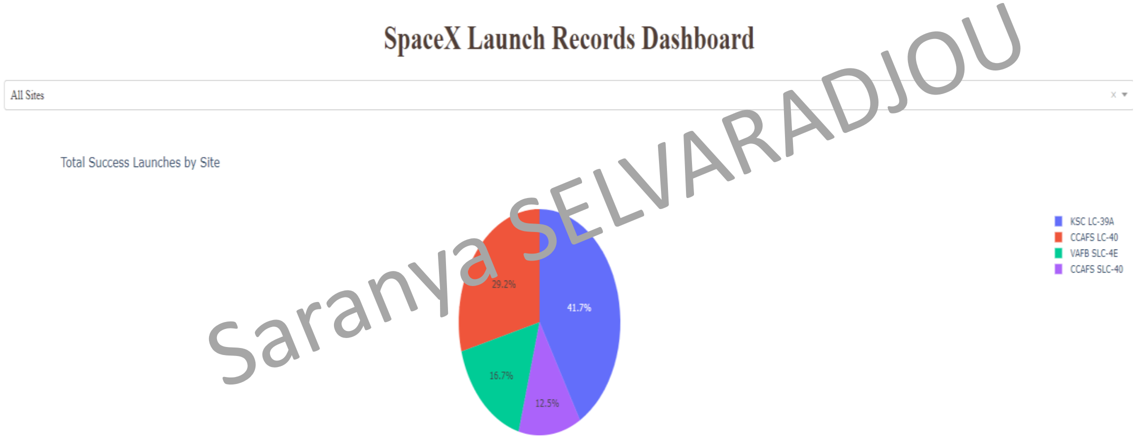
Map Object	Use
Map marker	A simple marker that displays a pin at a specific location on the map
Icon marker	A marker that displays a custom icon at a specific location on the map
Circle marker	A marker that displays a circle at a specific location on the map, with a customizable radius and color
Polyline	A line that connects multiple points on the map
Marker cluster	A group of markers that are combined and displayed as a single marker, with a count of the number of markers

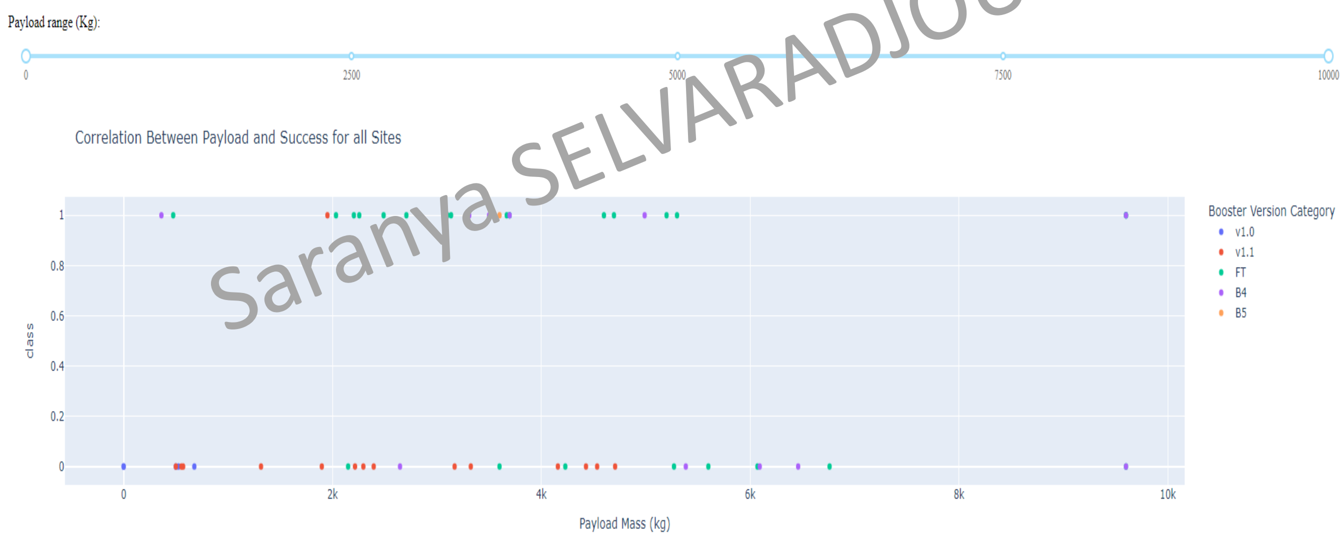
Github : https://github.com/SaraSS1/IBM-Data-Science-Professional-Certificate/blob/main/10 Applied Data Science Capstone/6 location folium.ipynb

Build a dashboard with Plotly Dash

Interactive visualizations of the data were created on the Plotly Dash dashboard by adding pie chart and scatter graph plots:

- A pie chart is used to illustrate the number of successful launches per site, allowing for easy comparison of launch success among different sites.
- A scatter graph is used to display the correlation between outcome (success or failure) and payload mass for different booster versions.





Github : https://github.com/SaraSS1/IBM-Data-Science-Professional-Certificate/blob/main/10 Applied Data Science Capstone/7 Interactive Dash.ipynb

Predictive Analysis - Classification

Building model

- Load data.
- Transform data.
- Split data into train and and test set using train_test_split.
- List down machine learning algorithms we want to use.
- Set our parameters and algorithms to GridSearchCV.
- Fit our datasets into GridSearchCV and train our model.

Evaluating Model

- Check accuracy for each model.
- Check hyperparameters for each type of algorithms.
- Plot confusion matrix

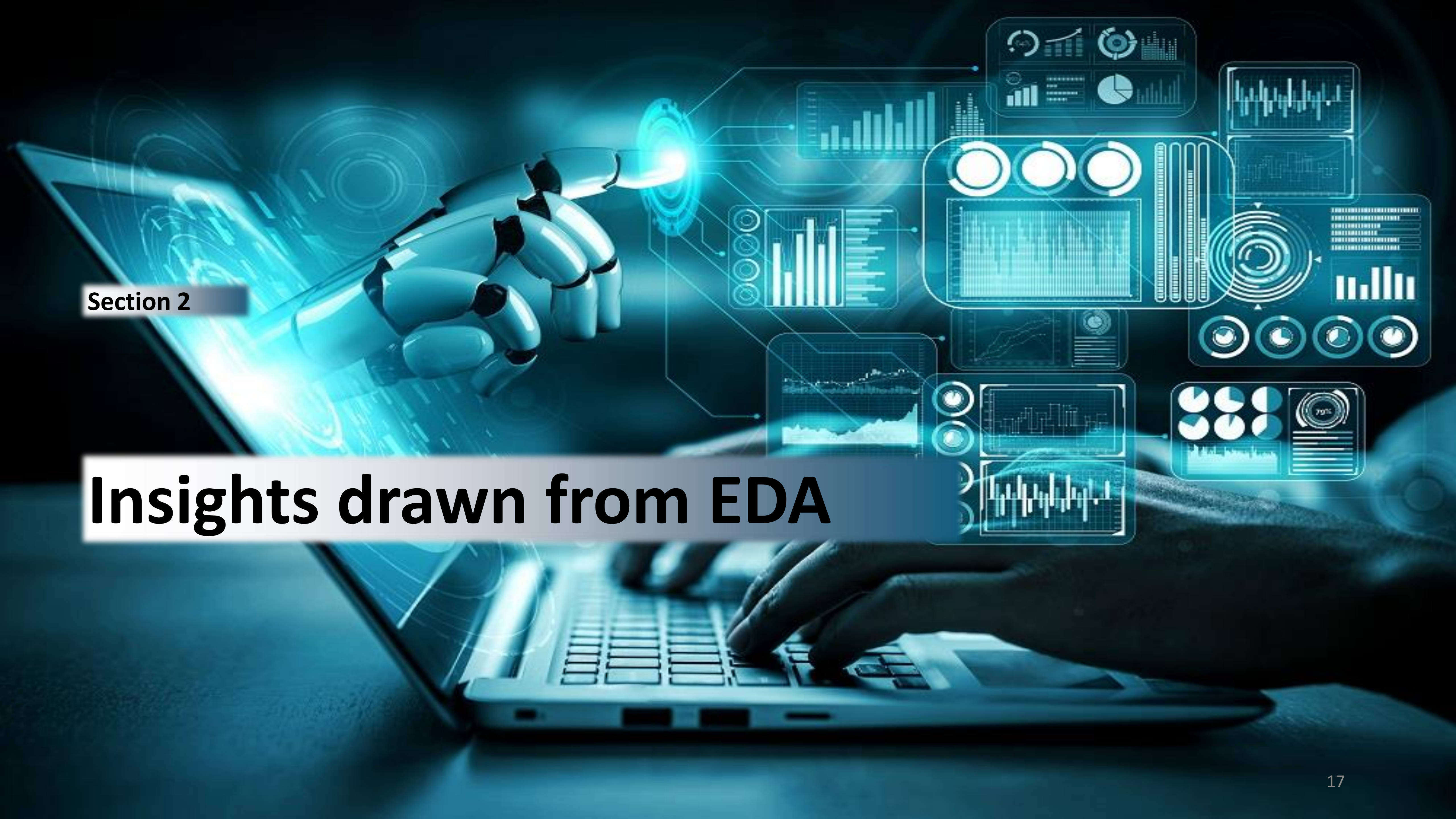
Finding the best classification model

- The model with best accuracy score is the best performing model

Github : https://github.com/SaraSS1/IBM-Data-Science-Professional-Certificate/blob/main/10 Applied Data Science Capstone/7 Interactive Dash.ipynb

Results

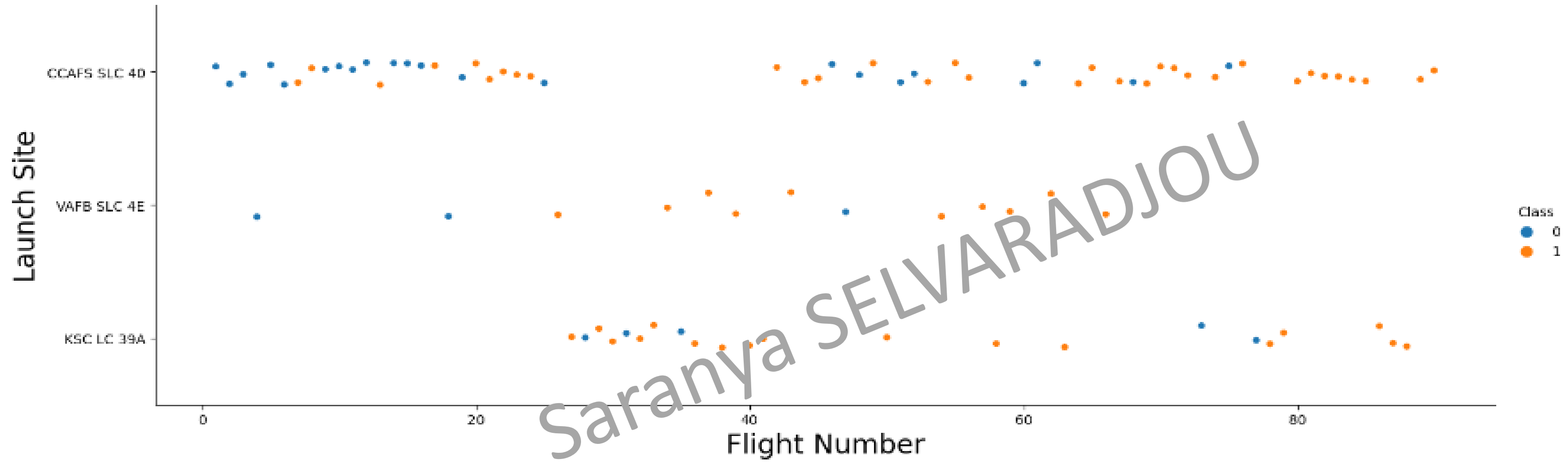
- **Exploratory Data Analysis Results**
- **Interactive Analytics Results**
- **Predictive Analysis Results**



Section 2

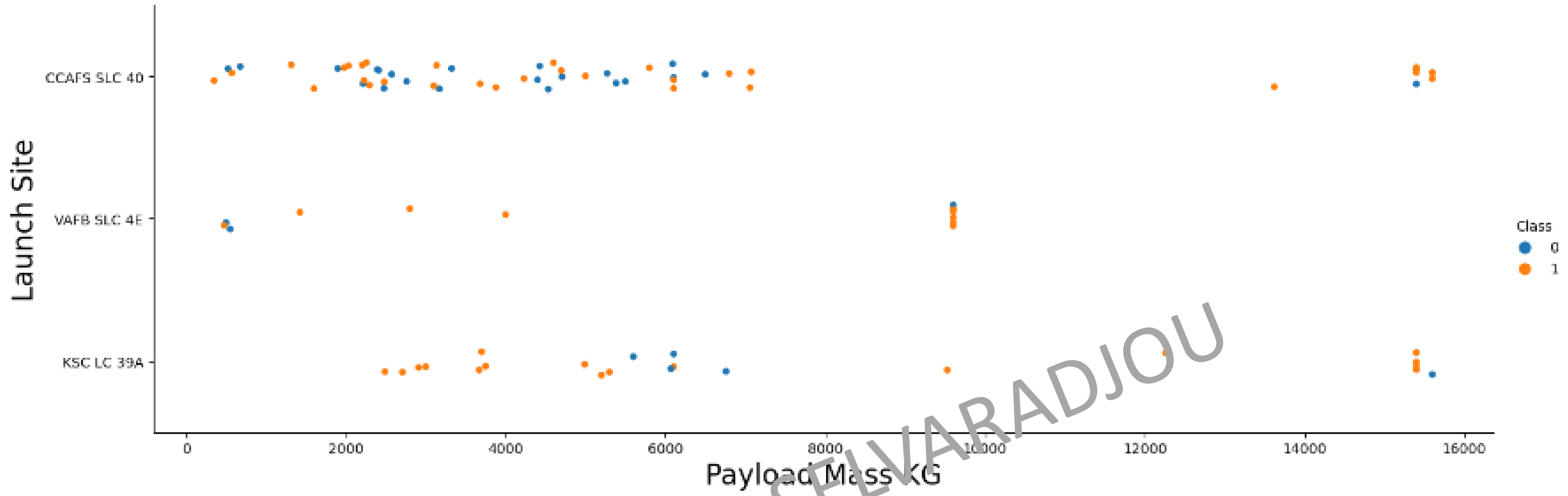
Insights drawn from EDA

Flight Number vs. Launch Site



- We can see that overall the success rate increases with time.
- Most of the earlier flights were launched from CCAFS SLC 40.
- Two earlier flights were launched from VAFB SLC 4E and none from KSC LC 39A.

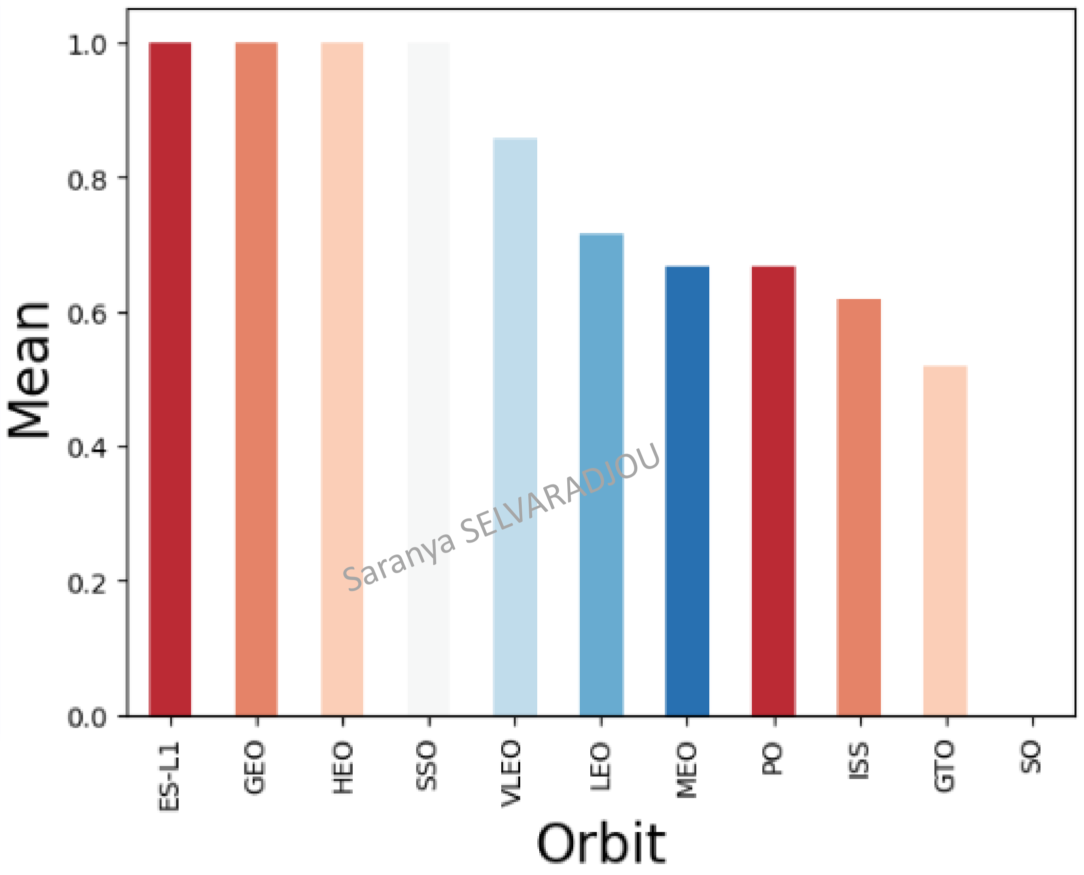
Payload vs. Launch Site



- The scatter plot shows that a higher payload mass is associated with more successful launches.
- Payload Mass more than 9000 KG have a high success rate

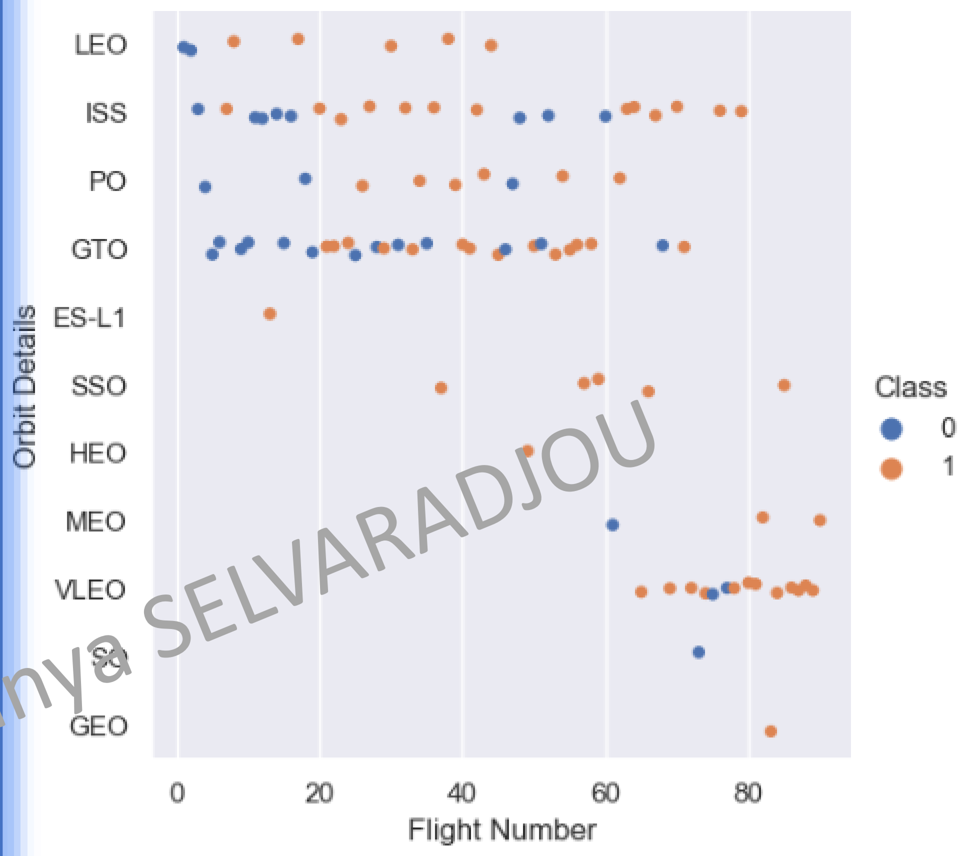
Success rate vs. Orbit type

- The following orbits have the best success rate:
  1. ES-L1
  2. GEO
  3. HEO
  4. SSO
- Orbit SO has the lowest success rate



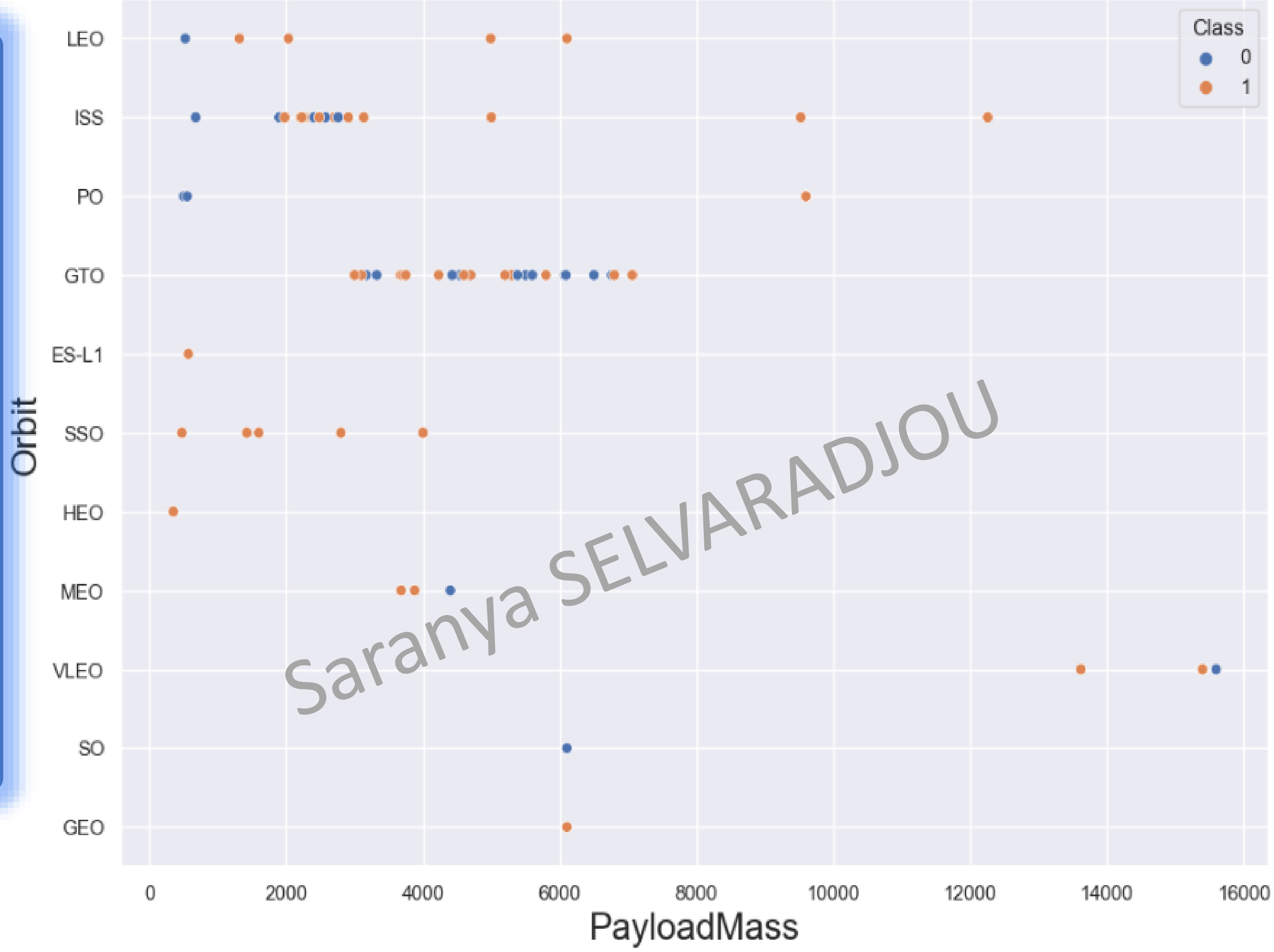
Flight number vs. Orbit type

- We can see that the orbit ES-L1, HEO and GEO had only one launch (successful) in total. This explains why they had the highest success rate.
- All 5 SSO launches were successful.
- Similarly, orbit SO also had only one launch(failed) in total. This explains why it had the lowest success rate.
- In general, sucess rate increases with flight number for all orbits



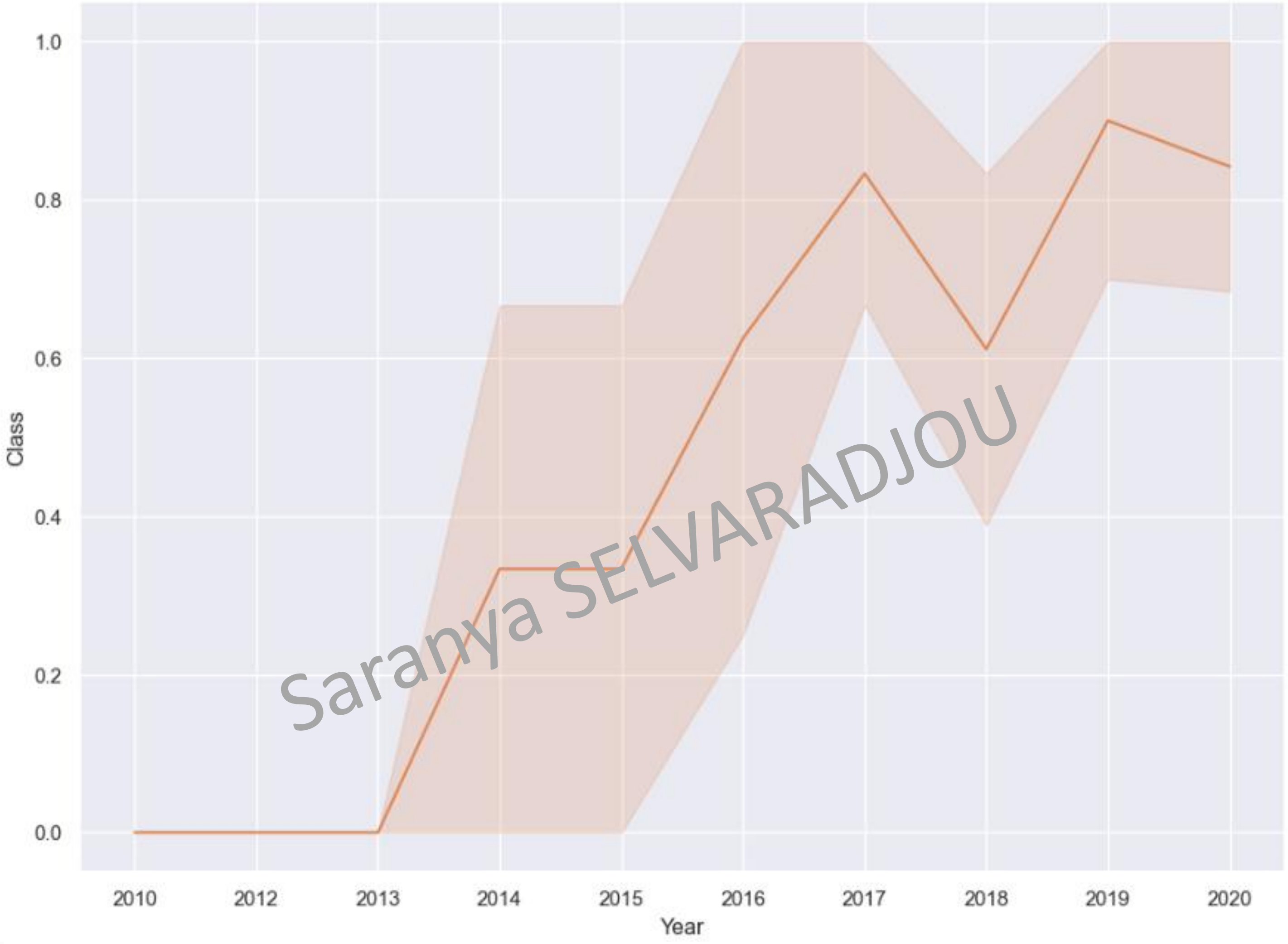
Payload vs. Orbit type

- The success rate of the orbits PO, LEO and ISS increases with Payload mass.
- We can't see a clear relationship between PayloadMass and success rate for the GTO orbit.



Launch success Yearly trend

- From 2010 to 2013 all landings were unsuccessful.
- The success rate started increasing from 2013 though there is a minor decrease in 2018.



All Launch Site names

- Find the name of the unique launch sites.

```
SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1;
```

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

Selects the unique values from the LAUNCH_SITE column in the SPACEXTBL table and orders the results by the LAUNCH_SITE column.

Launch Site names begin with ‘CCA’

- Find 5 records where launch sites begin with ‘CCA’.

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

Selects all columns (*) from the SPACEXTBL table and filters the results by rows where the LAUNCH_SITE column begins with 'CCA'.

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

Total Payload Mass

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

```
SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE
Customer = 'NASA (CRS)'
```

SUM(PAYLOAD_MASS__KG_)
45596

Selects the sum of the PAYLOAD_MASS__KG_ column from the SPACEXTBL table and filters the results by rows where the Customer column is equal to 'NASA (CRS)'.

Average Payload Mass By F9 v1.1

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

SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.1%'

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

Selects the average of the PAYLOAD_MASS__KG_ column from the SPACEXTBL table and filters the results by rows where the Booster_Version column begins with 'F9 v1.1'.

First successful ground landing date

- List the date when the first succesful landing outcome in ground pad was achieved

```
SELECT MIN(DATE) FROM SPACEXTBL WHERE "LANDING _OUTCOME" = 'Success (ground pad)';
```

MIN(DATE)
01-05-2017

Selects the minimum value of the DATE column from the SPACEXTBL table and filters the results by rows where the LANDING _OUTCOME column is equal to 'Success (ground pad)'.

Successful droneship landing

- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE
PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 AND "LANDING
_OUTCOME" = 'Success (drone ship)';
```

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

Selects the unique values from the BOOSTER_VERSION column in the SPACEXTBL table and filters the results by rows where the PAYLOAD_MASS__KG_ column is between 4000 and 6000 and the LANDING _OUTCOME column is equal to 'Success (drone ship)'.

Total number of mission outcomes

- List the total number of successful and failure mission outcomes

```
SELECT MISSION_OUTCOME, COUNT(*) AS Total FROM SPACEXTBL
GROUP BY MISSION_OUTCOME
```

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

Selects the MISSION_OUTCOME column and a count of the rows in the SPACEXTBL table, grouped by MISSION_OUTCOME.

Boosters carried maximum payload

- List the names of the booster_versions which have carried the maximum payload mass.

```
SELECT BOOSTER_VERSION,PAYLOAD_MASS__KG_ FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (SELECT
MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

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

Selects the BOOSTER_VERSION and PAYLOAD_MASS__KG_ columns from the SPACEXTBL table and filters the results by rows where the PAYLOAD_MASS__KG_ column is equal to the maximum value of the PAYLOAD_MASS__KG_ column in the SPACEXTBL table.

2015 Launch records

- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

```
SELECT substr(Date, 4, 2) AS Month, "Landing _Outcome",
Booster_Version, Launch_Site FROM SPACEXTBL WHERE "Landing
_Outcome" = 'Failure (drone ship)' AND substr(Date, 7, 4) = '2015'
```

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

Selects the month from the Date column, the Landing _Outcome column, the Booster_Version column, and the Launch_Site column from the SPACEXTBL table, and filters the results by rows where the Landing _Outcome column is equal to 'Failure (drone ship)' and the year of the Date column is equal to '2015'.

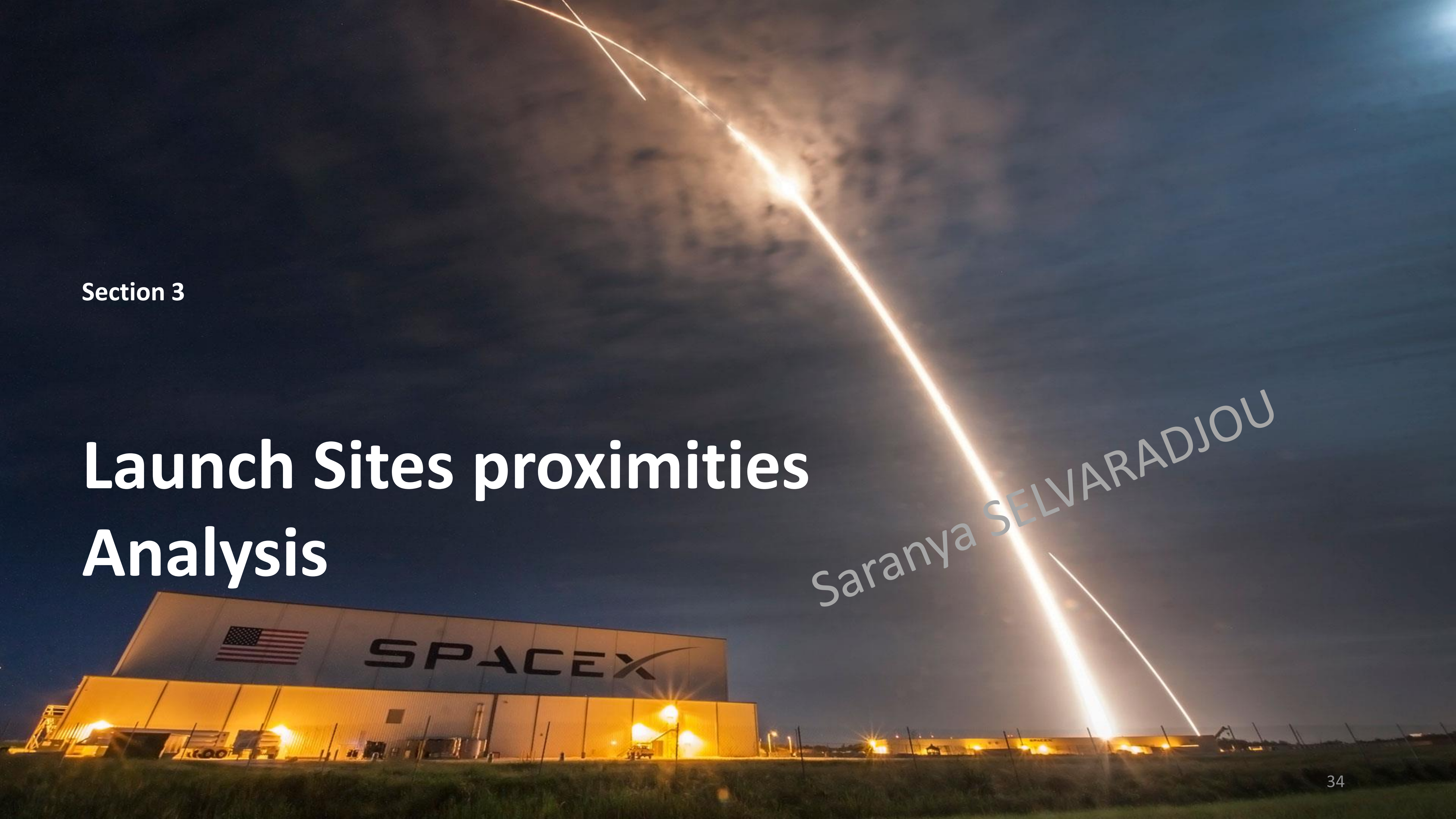
Rank landing outcomes

- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
SELECT "Landing _Outcome", Count(*) AS QTY FROM SPACEXTBL
WHERE "Landing _Outcome" LIKE '%Success%' AND Date BETWEEN
'04-06-2010' AND '20-03-2017' GROUP BY "Landing _Outcome"
```

Landing _Outcome	QTY
Success	20
Success (drone ship)	8
Success (ground pad)	6

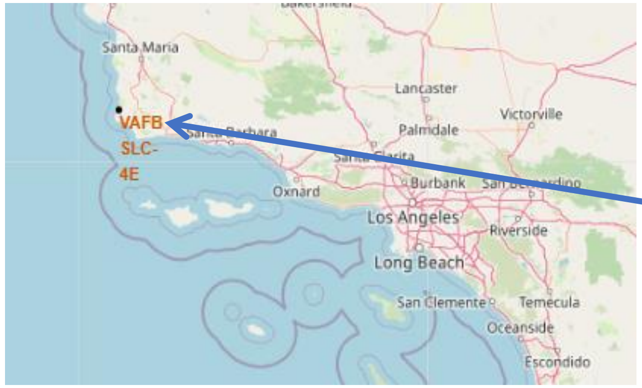
Selects the Landing _Outcome column and a count of the rows in the SPACEXTBL table, grouped by Landing _Outcome, and filters the results by rows where the Landing _Outcome column contains the word 'Success' and the Date column is between '04-06-2010' and '20-03-2017'.

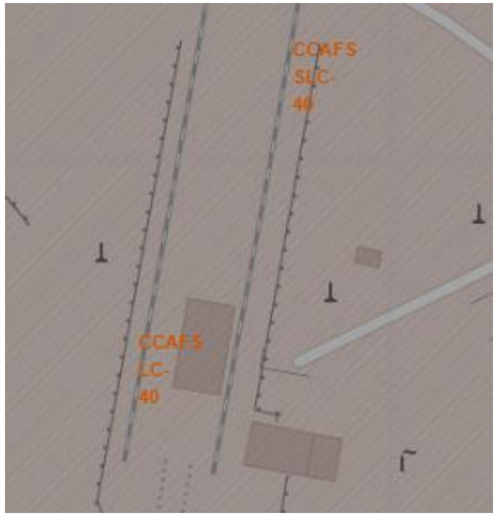


Section 3

Launch Sites proximities Analysis

All Launch sites on Folium map





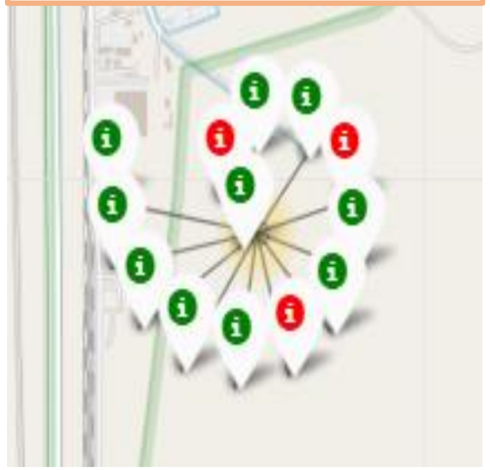
We can see that all SpaceX launch sites are near the coasts of United States of America, specifically in the States of Florida and California.

Success/failure of launches by launch site

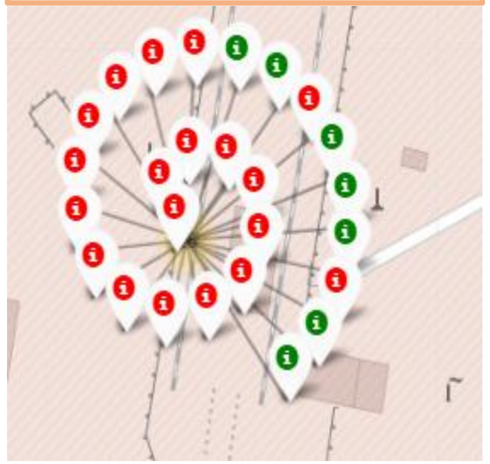
Green marker represents successful launches and **red** marker represents failed launches.

VAFB SLC-4E

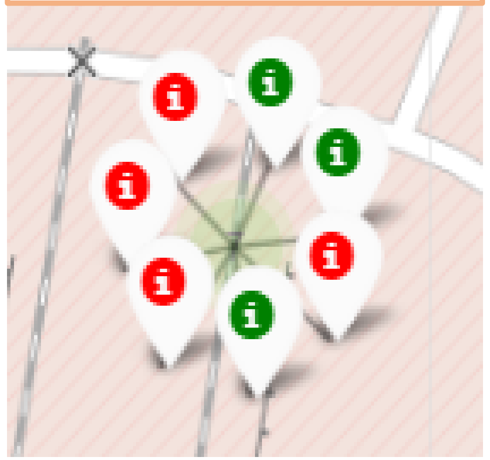
KSC LC 39-A



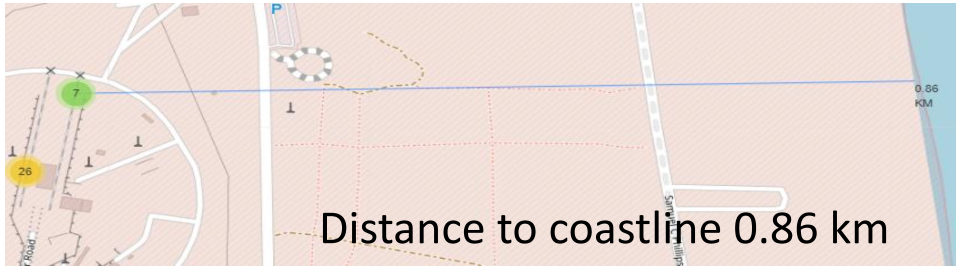
CCAFS LC-40



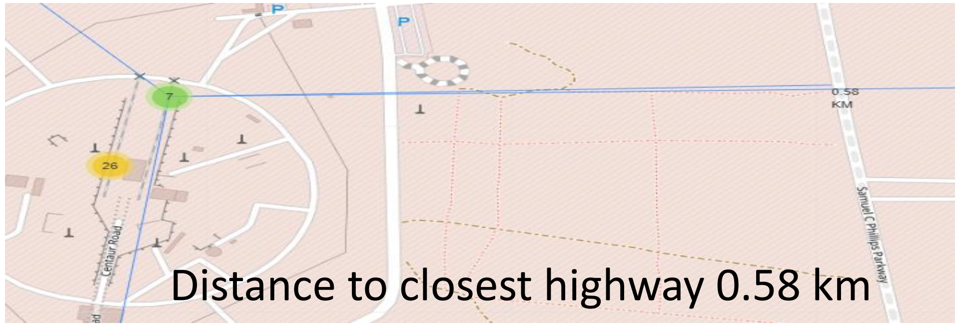
CCAFS SLC-40



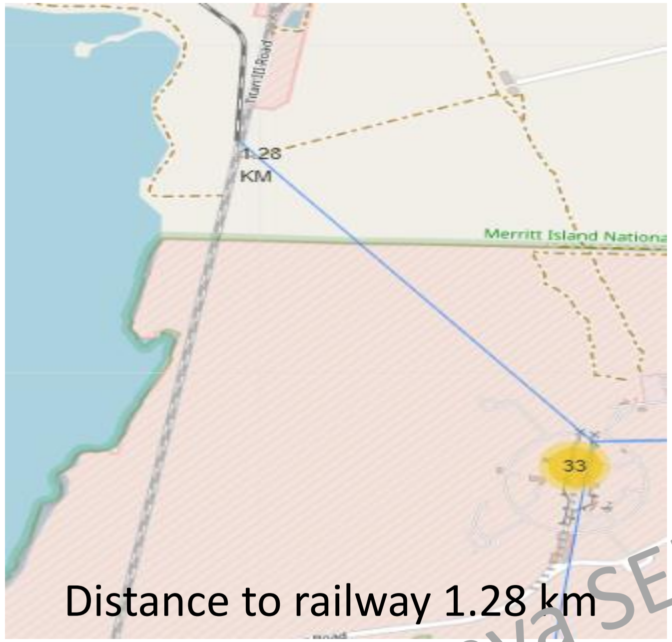
Proximity of Launch Sites to Transportation, Urban Areas, and Coastlines



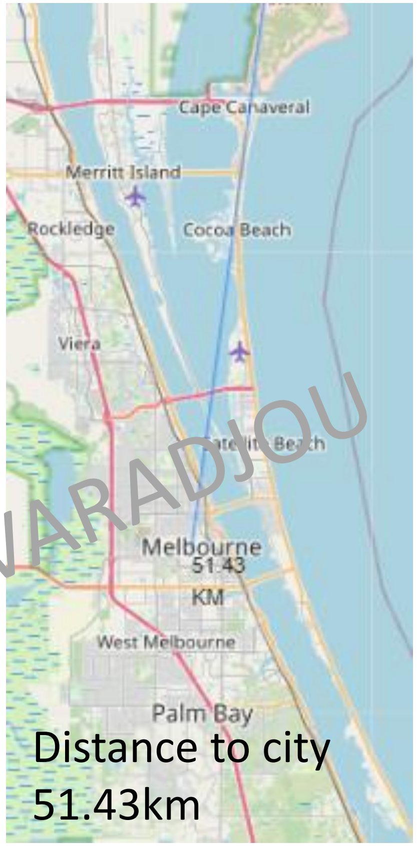
Distance to coastline 0.86 km



Distance to closest highway 0.58 km



Distance to railway 1.28 km



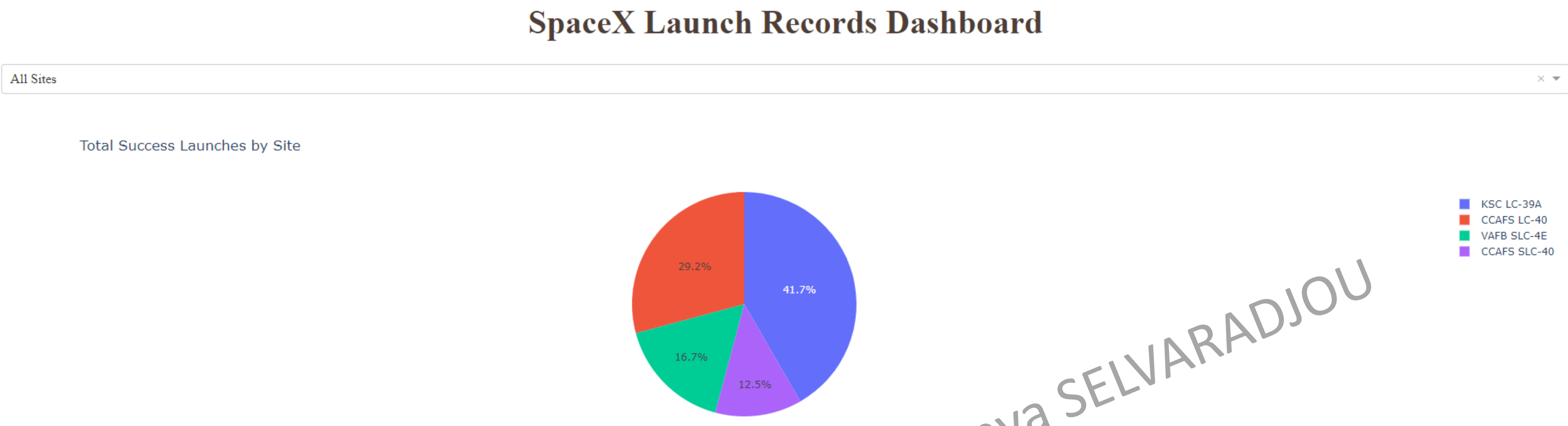
Distance to city 51.43km

- Launch sites are in close proximity to coastline, highways and railways. It is easier to transport equipment, supplies, and personnel to and from the launch site.
- Launch sites are not in close proximity to cities, this is understandable and it is to minimize the danger to population.
- Launch sites are not in close proximity to equator.

Section 4

Build a dashboard with Plotly Dash

Successful launches by site



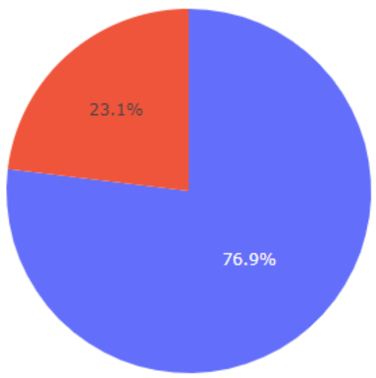
The launch site with the highest success rate is KSC LC-39A, with 41.7% of the total successful launches.

Success vs. Failed count for specific launch site

SpaceX Launch Records Dashboard

KSC LC-39A

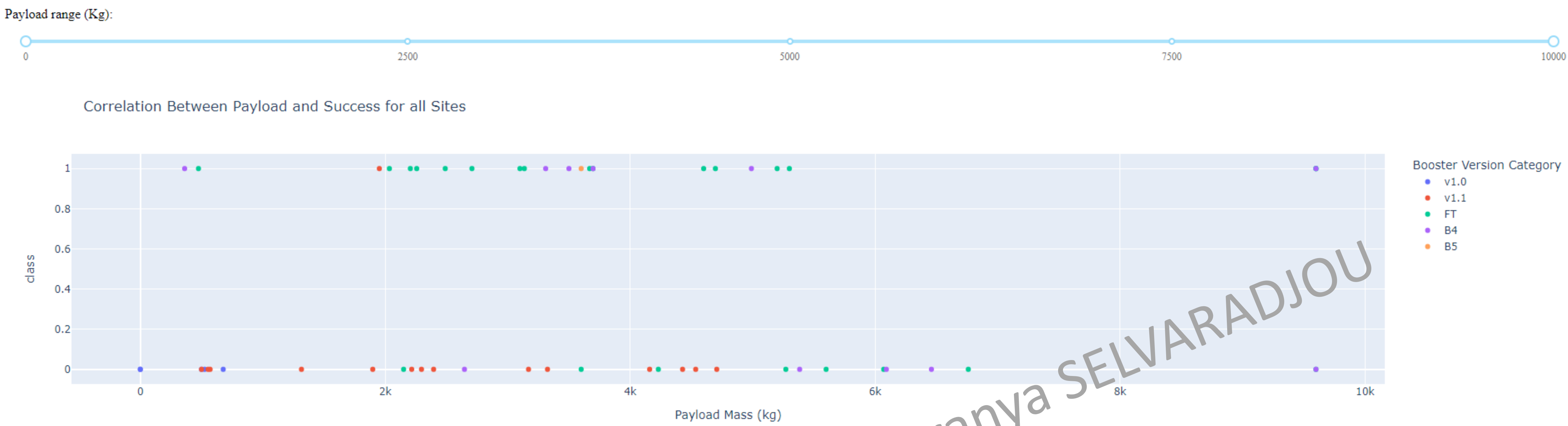
Total Success Launches for site KSC LC-39A



1
0

KSC LC-39A has a success rate of 76.9% and a failure rate of 23.1%.

Relationship between payload and success



The success rate for launches carrying large payloads is lower compared to those with smaller payloads.

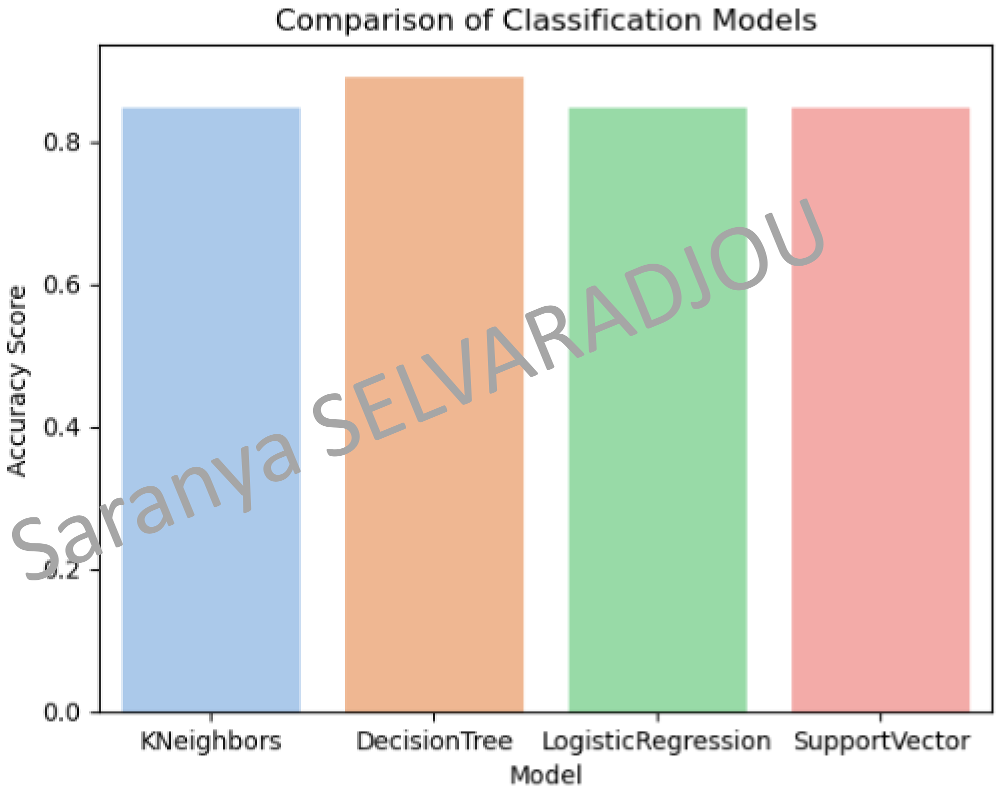
Section 5

Predictive Analysis(classification)

Classification Accuracy

```
Best model is DecisionTree with a score of 0.8910714285714285
Best params is : {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Model	Accuracy Score
KNN	0.848214
Decision Tree	0.891071
Logistic Regression	0.846429
SVM	0.848214



Confusion Matrix

We can conclude that the Decision Tree model is the best model because it has high TP and TN rates and low FP and FN rates: Out of 18 predictions, 17 are true predictions and 1 is an incorrect prediction.

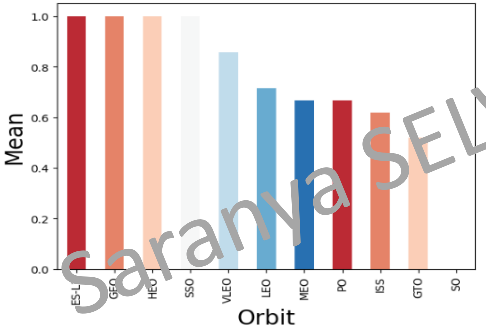
Predicted Values

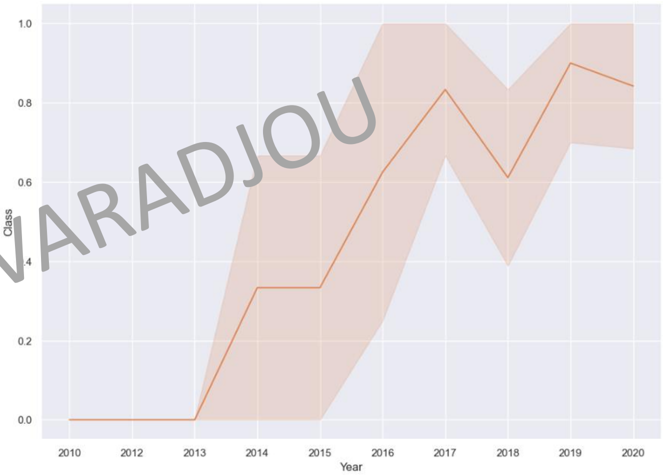
Actual Values	**Negative**	**Positive**
Negative	**TN**	**FP**
Positive	**FN**	**TP**

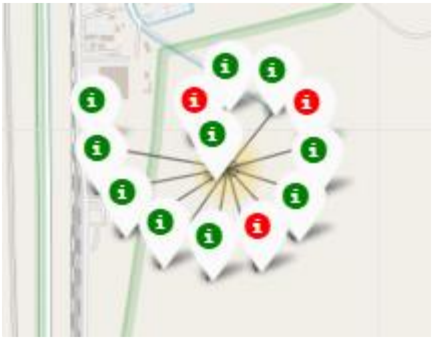
Decision Tree

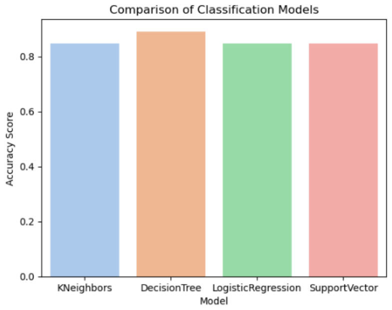
Conclusions

- The success rate increases with time.
- Orbits ES-L1, GEO, HEO and SSO have the highest success rate.
- The launch site with the highest success rate is KSC LC-39A, with 41.7% of the total successful launches.
- The best model is decision tree.







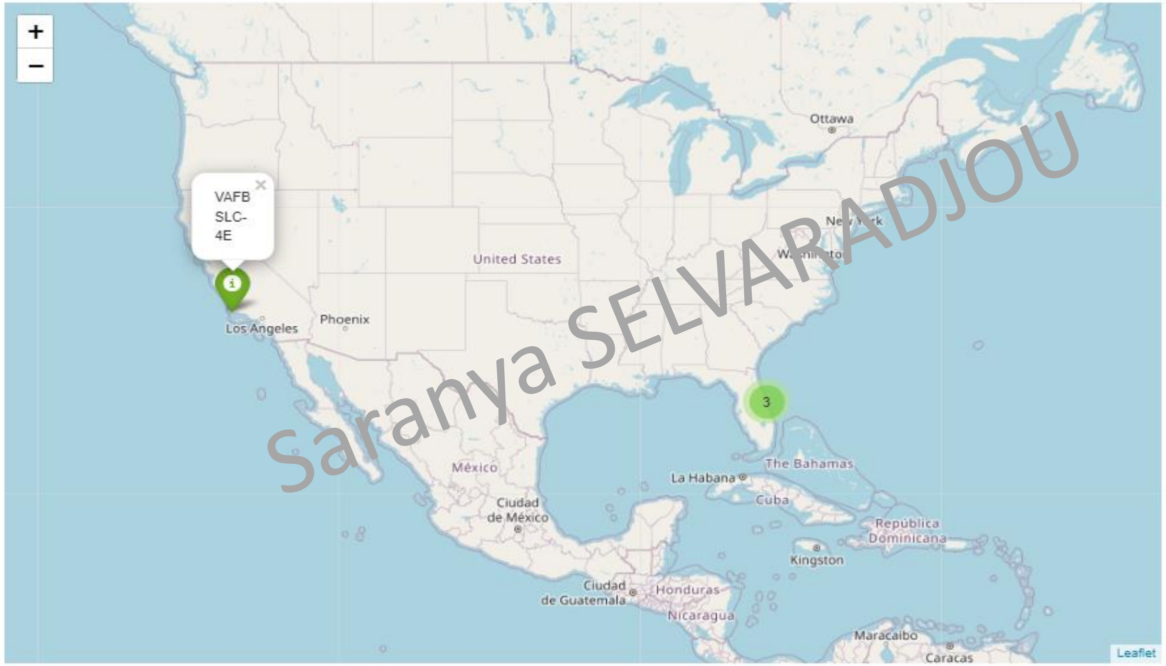


Appendix

Data Collection slide template :https://www.youtube.com/watch?v=InyqaOFRwKw&ab_channel=PowerPointSchool
Picture credits: NASA image gallery and pixabay

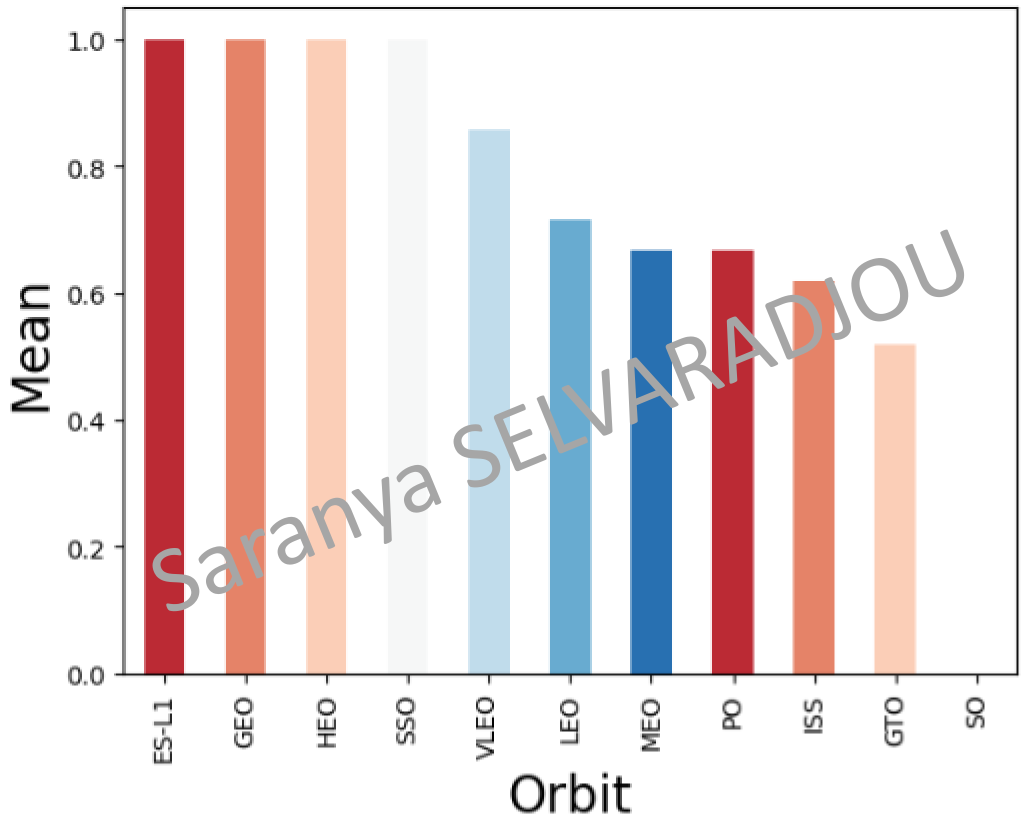
Folium Map

```python
import folium
from folium.plugins import MarkerCluster

site_map = folium.Map(location=nasa_coordinate, zoom_start=5)

# Create a marker cluster for the launch sites
marker_cluster = MarkerCluster().add_to(site_map)

# For each launch site, create a marker and add it to the marker cluster
for index, row in launch_sites_df.iterrows():
    coordinate = [row['Lat'], row['Long']]
    folium.Marker(coordinate, icon=folium.Icon(color='green'), popup=row['Launch Site']).add_to(marker_cluster)

site_map
```



Ordering the bars in the bar chart in ascending order

```
colors = sns.color_palette("RdBu", n_colors=7)
df.groupby(['Orbit']).mean()['Class'].sort_values(ascending=False).plot(kind='bar', color = colors)
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("Mean",fontsize=20)
plt.show()
```




THANK YOU