IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Obinna Okoyeigbo
04/09/2022

Outline

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

Executive Summary

Summary of methodologies

- In this project, data was collected from SpaceX API and by web scrapping Wikipedia, after which the data was cleaned and processed.
- Exploratory data analysis was performed using SQL and python visualization packages.
- Interactive Plotly Web App and Folium Maps were also generated to visualize the data.
- Different Machine Learning Models (Logistic Regression, KNN, Decision Tree, SVM) were deployed for predictive analysis of the data

Summary of all results

- The best machine learning model that accurately predicted the outcome was acquired

Introduction

Project background and context

- SpaceX Falcon 9 rocket launches cost 62 million dollars; while other providers cost over 165 million dollars.
- This is because SpaceX can reuse the first stage after a successful landing.
- Therefore if we can determine if the first stage will land, then we can determine the cost of a launch.
- This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Problems you want to find answers to

- Determine what factors influence Falcon 9 launch success
- Will the first stage land successfully
- What condition must be met to increase the probability of success.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - Request launch data from SpaceX's Rest API
    - Web scraping to collect Falcon 9 historical launch records from a Wikipedia
- Perform data wrangling
    - Non relevant data were dropped, missing values were replaced
    - Categorical variables were transformed using One Hot Encoding
- Perform exploratory data analysis (EDA) using visualization and SQL
    - Scatter plots, and bar charts were used to explore relationships, and SQL queries to understand the data
- Perform interactive visual analytics using Folium and Plotly Dash
    - Launch site Maps and interactive dashboard applications were built for interactive analysis
- Perform predictive analysis using classification models
    - Models were built, tuned and evaluated to predict the outcome.

Data Collection – SpaceX API

- Using the Get request, rocket launch data was collected from SpaceX API with the following URL: https://api.spacexdata.com/v4/launches/past
- The response was decoded to a Json file and turned into a Pandas dataframe using .json_normalize()
- Data is cleaned to extract only required information
- A new Pandas data frame is created from the dictionary launch_dict.
- Cleaned data is assigned to new data frame
- The dataframe is filtered to include only Falcon 9 launches,
- Missing values were replaced, and data is exported to a CSV

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

```
response = response.json()
data = pd.json_normalize(response)
```

```
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}
```

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_data[launch_data['BoosterVersion'] != 'Falcon 1']
```

```
data_falcon9.isnull().sum()
```

```
PayloadMass_mean = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, PayloadMass_mean)
```

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

```
launch_data = pd.DataFrame(launch_dict)
```


GitHub URL: https://github.com/obydelion/SpaceX-Falcon9-Landing-Prediction/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

- Request the Falcon9 Launch Wiki page from its URL
- Create a BeautifulSoup object from the HTML response
- Find all tables from the HTML
- Extract all column names from the HTML table header
- Create an empty dictionary with keys from the extracted column names
- Fill up the launch_dict with launch records extracted from table row
- Create dataframe and export it to a CSV

```
data  = requests.get(static_url).text
```

```
soup = BeautifulSoup(data,"html.parser")
```

```
html_tables = soup.find_all('table')
```

```
launch_dict= dict.fromkeys(column_names)
```

```
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']=[]
```

```
df=pd.DataFrame(dict([(k, pd.Series(v))
                      for k, v in launch_dict.items()]))
```

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

GitHub URL : https://github.com/obydelion/SpaceX-Falcon9-Landing-Prediction/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

- Load SpaceX data from previous section
- Calculate the number of launches on each site
- Calculate the number and occurrence of each orbit
- Calculate the number and occurrence of mission outcome per orbit type
- Create a landing outcome label from Outcome column
- Export data to a CSV

```
df=pd.read_csv("https://cf-courses-data.s3.u
```

```
df['LaunchSite'].value_counts()
```

```
df['Orbit'].value_counts()
```

```
landing_outcomes=df['Outcome'].value_counts()
```

```
landing_class = []
for outcome in df['Outcome'].tolist():
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

```
df.to_csv("dataset_part_2.csv", index=False)
```


GitHub URL: https://github.com/obydelion/SpaceX-Falcon9-Landing-Prediction/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

The following charts were plotted for EDA and data visualization:

- **Bar chart** to visualize the relationship between success rate of each orbit type
- **Scatter Plots** to visualize the relationship between variables
  - Flight Number vs PayloadMass
  - Flight Number vs Launch Site
  - FlightNumber vs Orbit type
  - Payload vs Launch Site
  - Payload vs Orbit type
- **Line Charts** to visualize the launch success yearly trend

GitHub URL: https://github.com/obydelion/SpaceX-Falcon9-Landing-Prediction/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

SQL queries were performed to:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was acheived
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass.
- 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.
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order

GitHub URL: https://github.com/obydelion/SpaceX-Falcon9-Landing-Prediction/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb

Build an Interactive Map with Folium

To build an interactive map with folium, the following steps where taken:

- TASK 1: Mark all launch sites on a map
- TASK 2: Mark the success/failed launches for each site on the map
- TASK 3: Calculate the distances between a launch site to its proximities

To achieve the above, the following objects where added to the map:

- folium.map.Marker(): to add a marker on each launch site
- Folium.Circle(): to add a highlighted circle area with a text label on a specific coordinate
- MarkerClusters(): to simplify a map containing many markers having the same coordinate.
- folium.PolyLine(): to show a distance line between two launch sites

Build a Dashboard with Plotly Dash

The dashboard app was built with the following plots and interactions added:

- Launch Site Drop-down menu: to enable us select different launch sites.
- Pie Chart: to visualize the success rate of the different launch sites.
- Callback function: to render success-pie-chart based on selected site dropdown
- Scatterplot: to visualize how the launch outcome correlates with the payload mass for the different booster version.
- Callback function: to render the success-payload-scatter-chart scatter plot
- Range Slider: to enable selection of different Payload mass range

GitHub URL: https://github.com/obydelion/SpaceX-Falcon9-Landing-Prediction/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

Building the Model

- Load the dataframe
- Create a NumPy array from the column Class, and assign it to Y
- Standardize and transform the data in X
- Split the data X and Y into training and testing data
- Create a logistic regression object and a GridSearchCV object
- Fit the object to find the best parameters

Evaluating the Model

- Calculate the accuracy on the test data using the method score
- Plot the confusion matrix

Improving and finding the best Model

- Fit the object to find the best parameters
- Find the best performing model
- Model with best accuracy on both traning and test data is the best model

GitHub URL: https://github.com/obydelion/SpaceX-Falcon9-Landing-Prediction/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

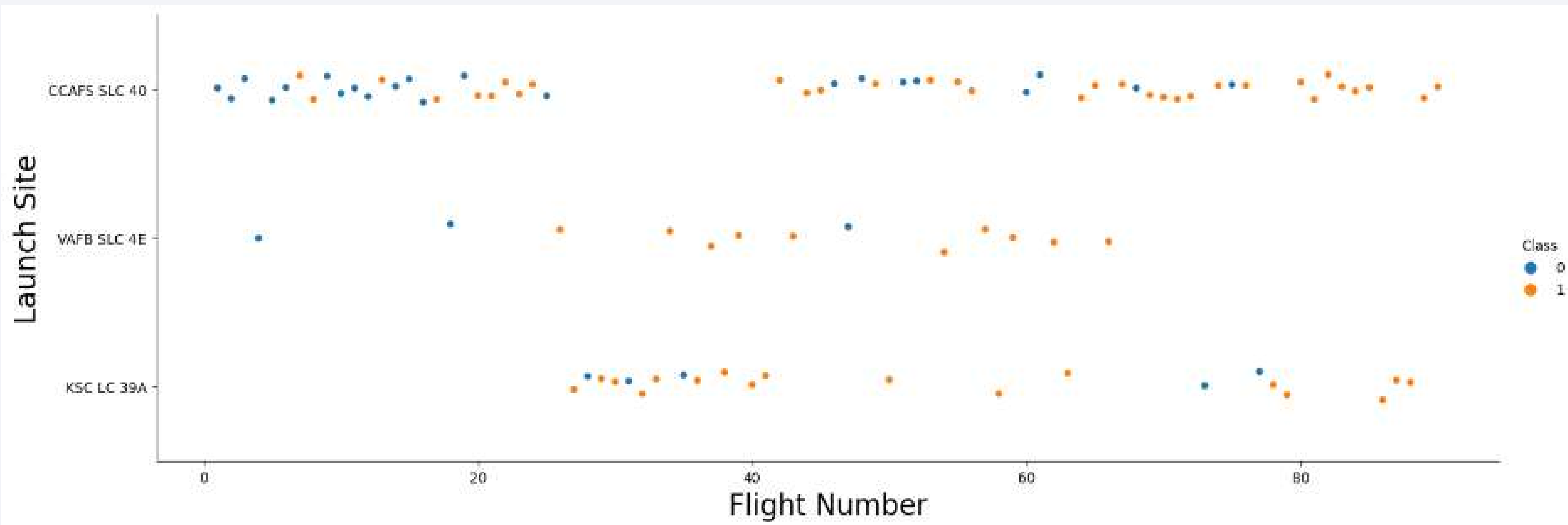
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

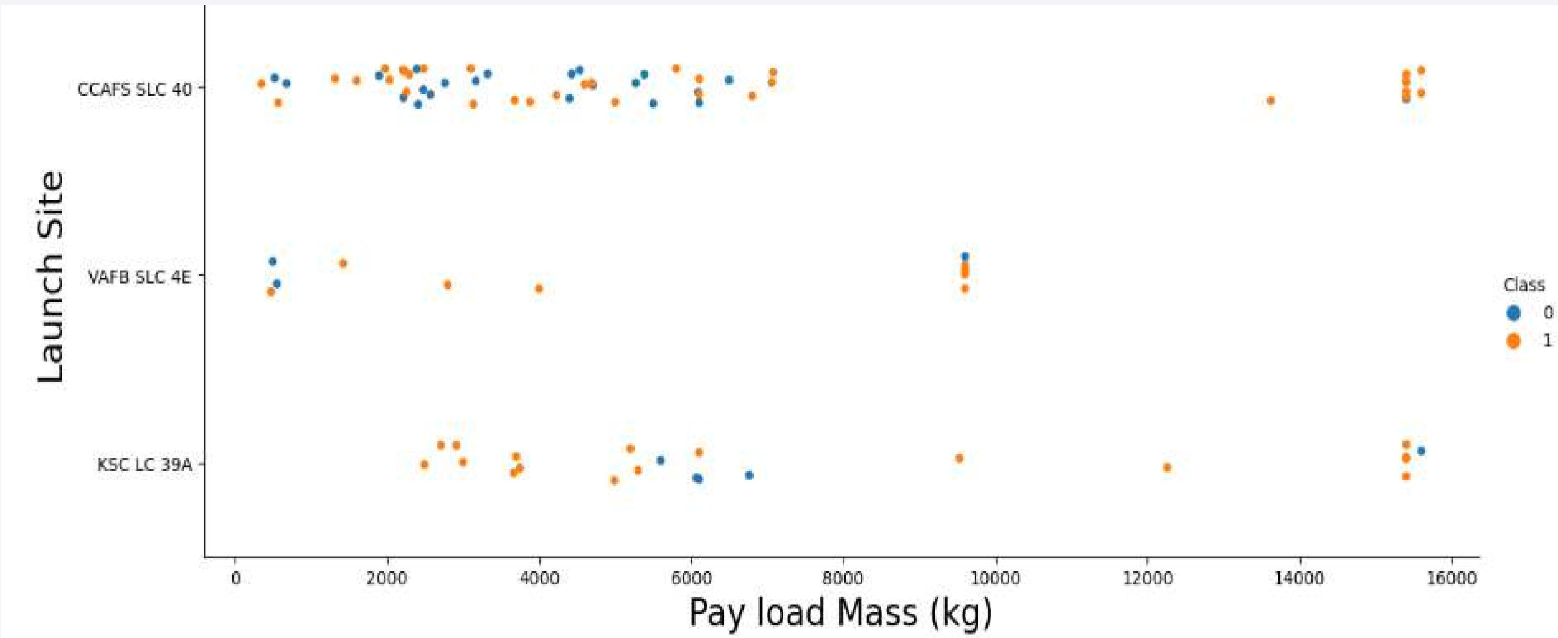
Flight Number vs. Launch Site

- It is observed that as the Flight number increases, the number of successful landing increases.
- CCAFS SLC 40 Launch site had more launches than other sites.

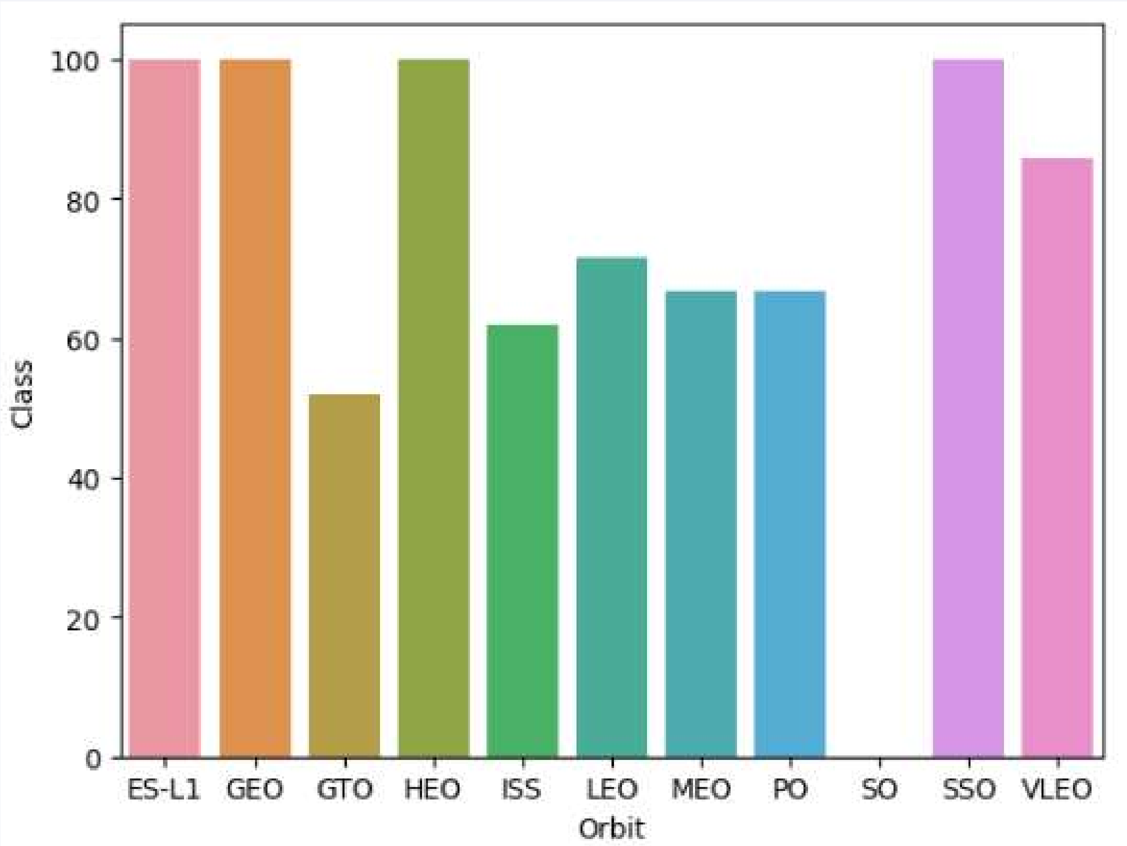


Payload vs. Launch Site

- As payload mass increases, success rate also increases.
- Payload mass above 8000 kg have a higher success rate.
- There isn't a clear correlation between payload mass and success rate.

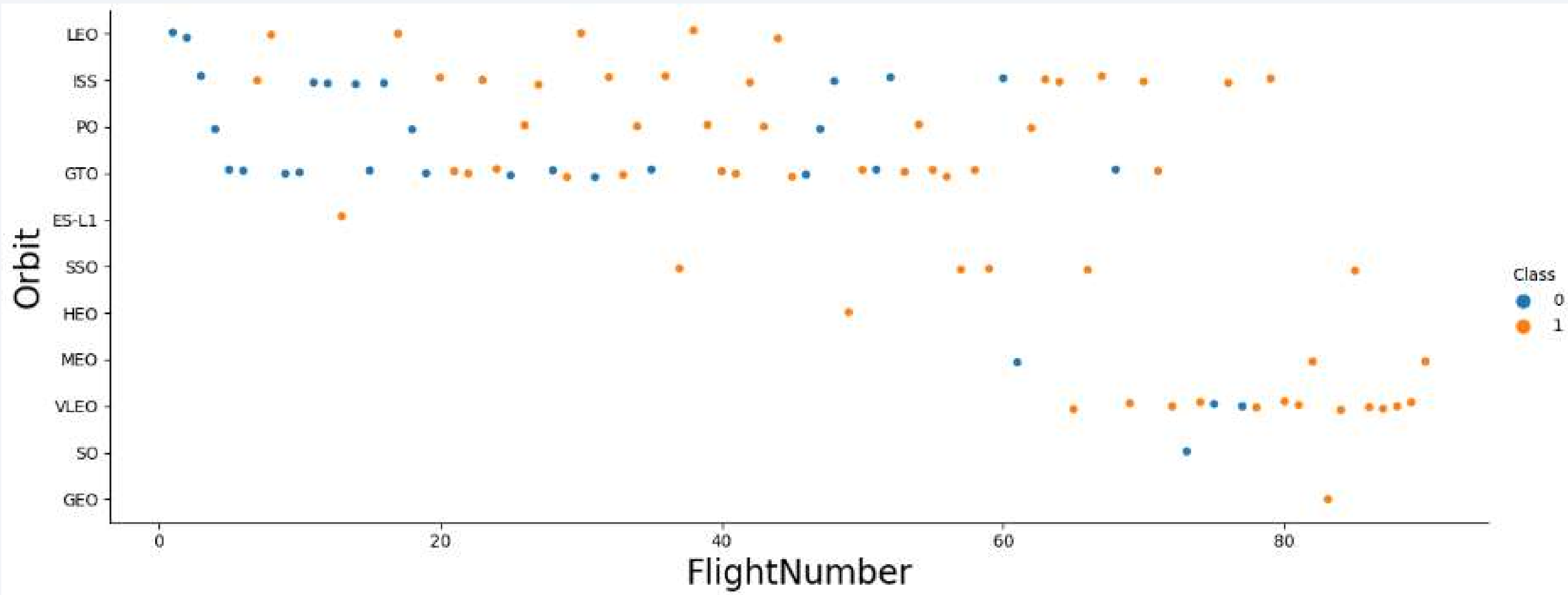


Success Rate vs. Orbit Type



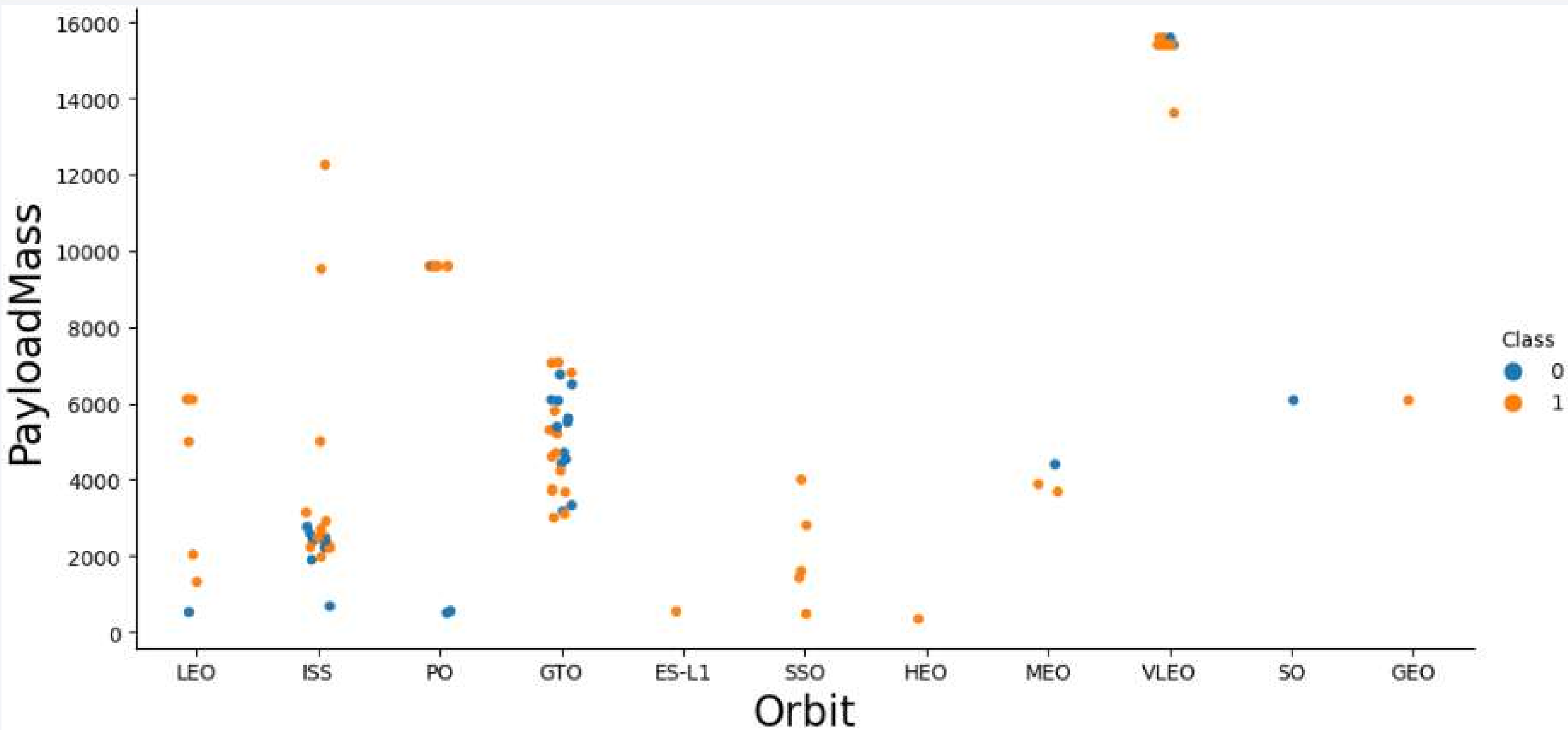
- ES-L1, GEO, HEO and SSO orbits have a 100% success rate
- SO orbit has a 100% failure rate (0% success rate).

Flight Number vs. Orbit Type



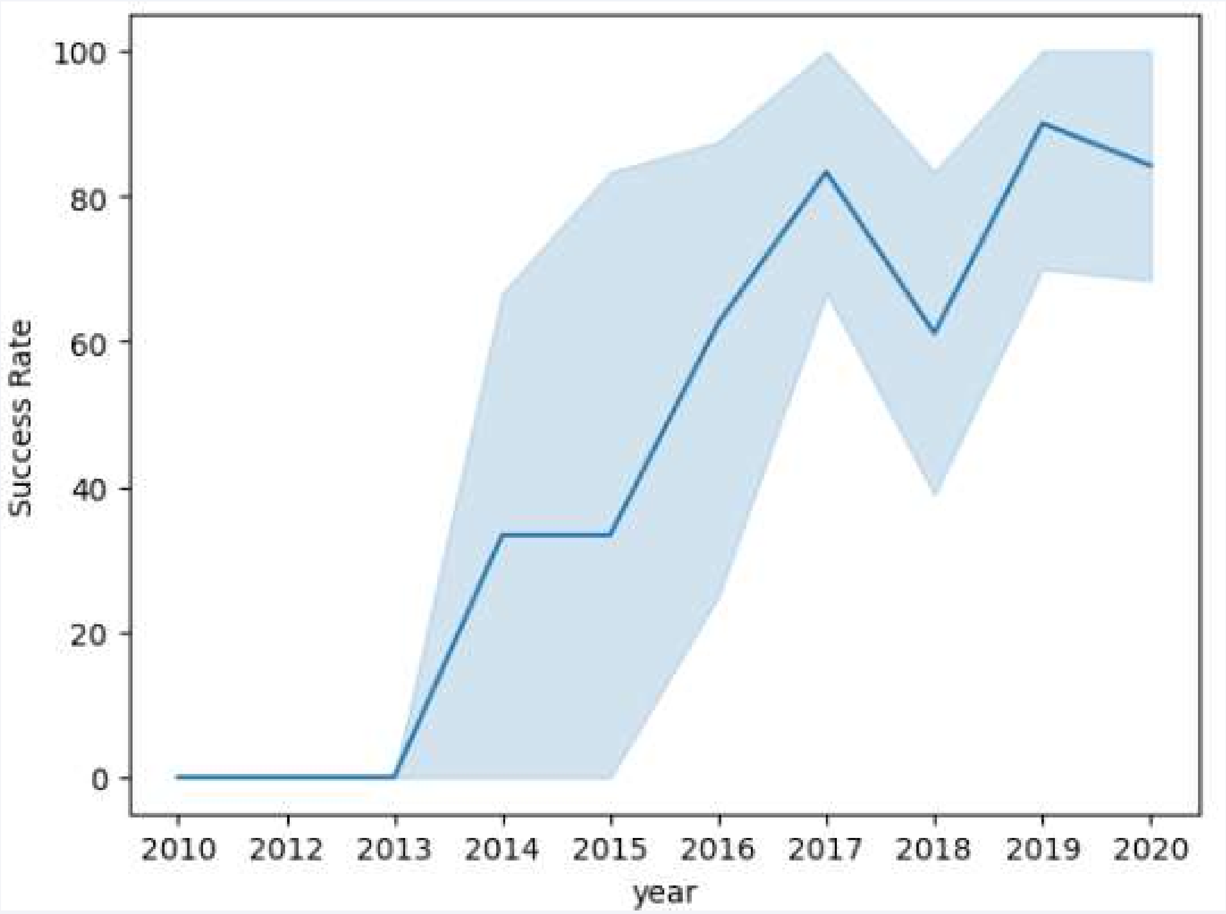
- The 100% success rate of ES-L1, GEO & HEO orbit could be because they had just 1 launch respectively.
- The SSO orbit had 5 launches which were all successful (100%)
- The SO orbit had only 1 launch which was unsuccessful (100%)

Payload vs. Orbit Type



- VLEO orbit is used for very high payload mass above 14000kg
- No strong relationship between the payload mass and orbit

Launch Success Yearly Trend



- It is observed that there was no successful landing between 2010 and 2013
- After 2013, the success rate increased until after 2017 when it dropped till 2018, after which it increased again to 2019

All Launch Site Names

- Using SQL Magic, the database was queried as follows:

```
%sql select distinct launch_site from SPACEXTBL
```

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

- The Distinct Keyword selects distinct (unique) values from the launch_site column.

Launch Site Names Begin with 'CCA'

- The code below displays 5 records where launch sites begin with `CCA`.

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

- Limit 5 is used to limit the entries to just 5, while like ‘CCA%’ fetches entries begining 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

- The total payload carried by boosters from NASA is calculated with the following code:

```
%sql select sum(payload_mass__kg_) as sum from SPACEXTBL where customer like 'NASA (CRS)'
```

sum
45596

- The SUM keyword calculates the total of the column

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 is calculated with the following code:

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as AVERAGE FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1'
```

AVERAGE
2928.4

- AVG calculates the average value
- Where filters the output to the given condition

First Successful Ground Landing Date

- The dates of the first successful landing outcome on ground pad was selected with the following code:

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

2015-12-22

- The MIN(DATE) selects the earliest (minimum/ smallest/ least) date.

Successful Drone Ship Landing with Payload between 4000 and 6000

- The names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 were listed with the following code:

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

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

- The where keyword is used to filter the result to satisfy all the conditions in the AND clause.

Total Number of Successful and Failure Mission Outcomes

- The total number of successful and failure mission outcomes was calculated with the following code:

```
%sql select mission_outcome, count(*) as count from SPACEXTBL
group by mission_outcome order by mission_outcome
```

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

- The group by clause ensures the success and failure are counted separately

Boosters Carried Maximum Payload

- The names of the booster which have carried the maximum payload mass were listed using the following code:

```
%%sql SELECT BOOSTER_VERSION FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

- The Select statement in the bracket, selects the max payload mass, and the value is used in the where condition.

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

- The failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015, were listed using the following code:

```
%%sql select substr(Date, 4, 2) as month, "LANDING _OUTCOME", booster_version, launch_site
from SPACEXTBL where substr(Date, 7, 4) ='2015' and "LANDING _OUTCOME" like 'Failure (drone ship)'
```

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

- The where keyword is used to filter the result for Failure(drone ship) and year 2015

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

```
%%sql select "LANDING _OUTCOME", count(*) as count from SPACEXTBL
where date >= '04-06-2010' and date <= '20-03-2017'
group by "LANDING _OUTCOME" order by count desc
```

Landing _Outcome	count
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

- Group by clause ensures the different landing outcomes are counted separately

Section 3

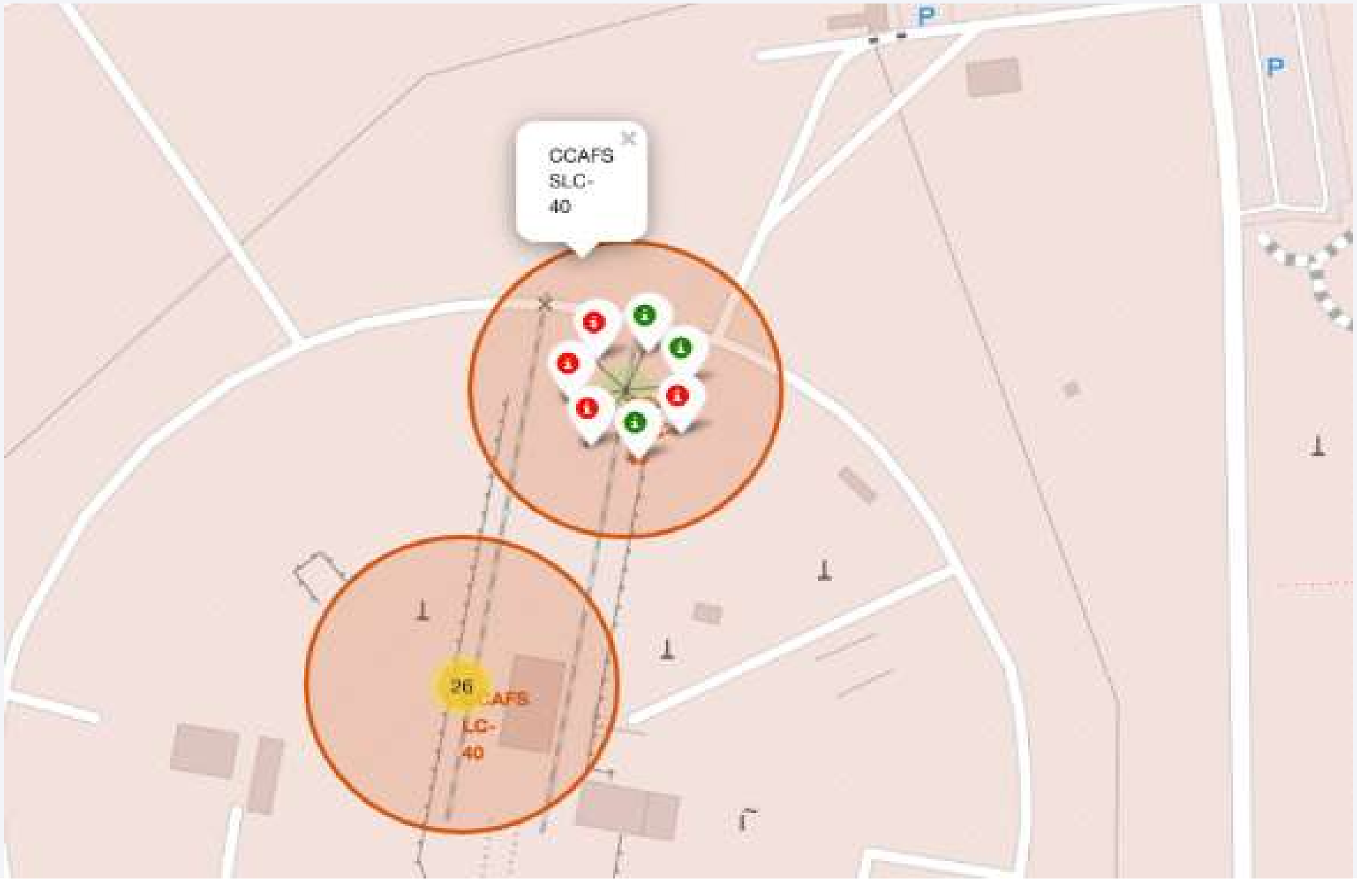
Launch Sites Proximities Analysis

Launch Sites Location Markers

- Folium map showing all launch sites' location markers on a global map

Color-labeled Launch Outcomes

- The Green shows a successful launch while the red shows a failed launch.



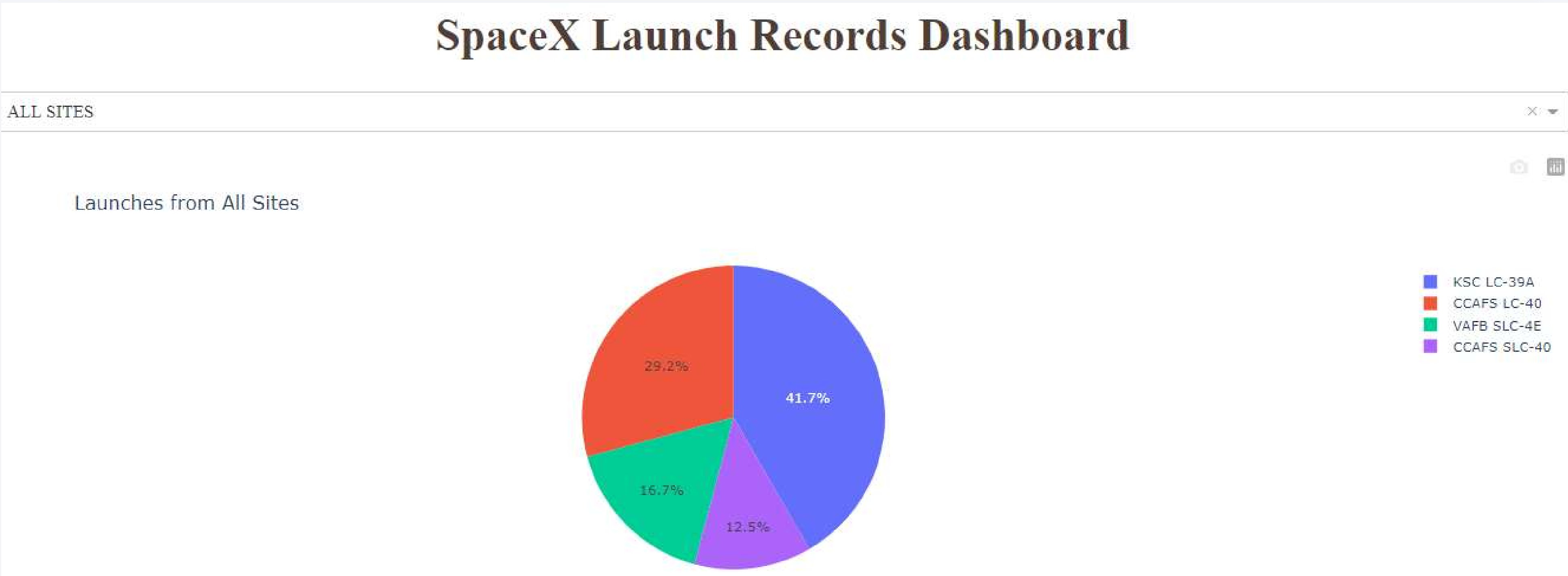
Launch Sites with Distance Markers to Proximities

- The launch site shown below is about 0.90 km to the nearest coastline

Section 4

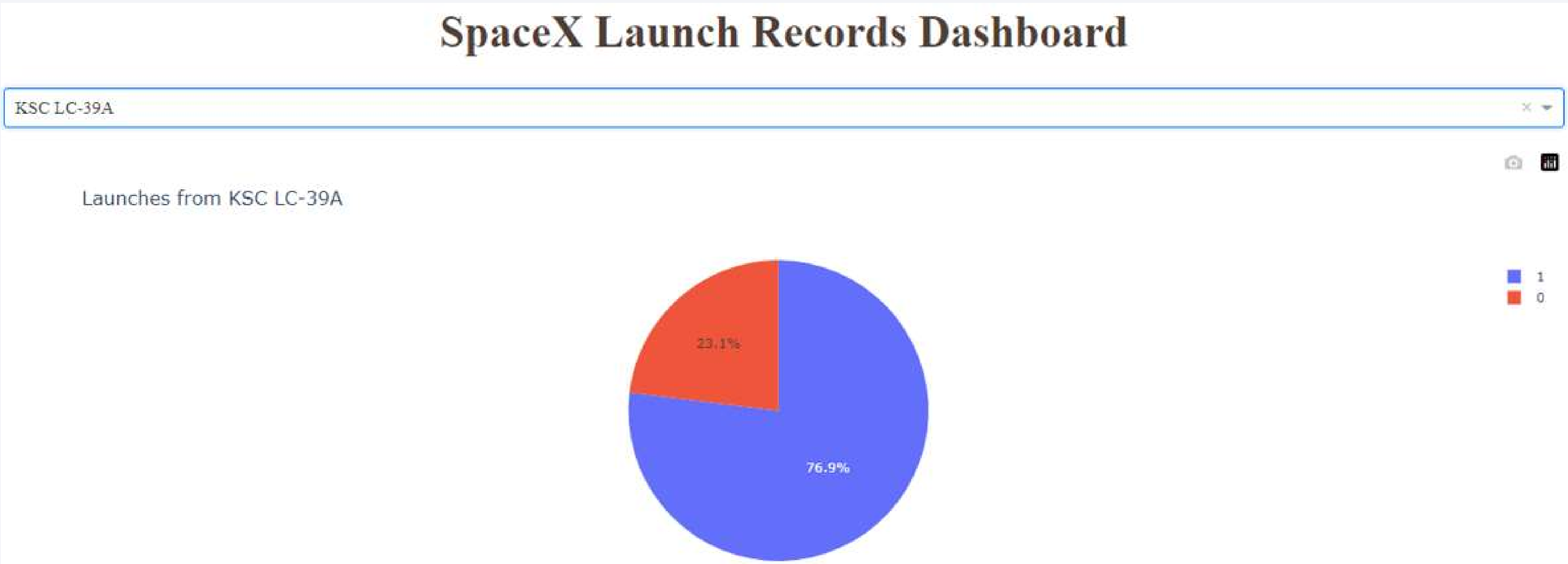
Build a Dashboard with Plotly Dash

Launch Success Count for all Sites



- It is observed that KSC LC-39A had the most successful launches with 41.7%, while CCAFS SLC-40 had the least with 12.5%

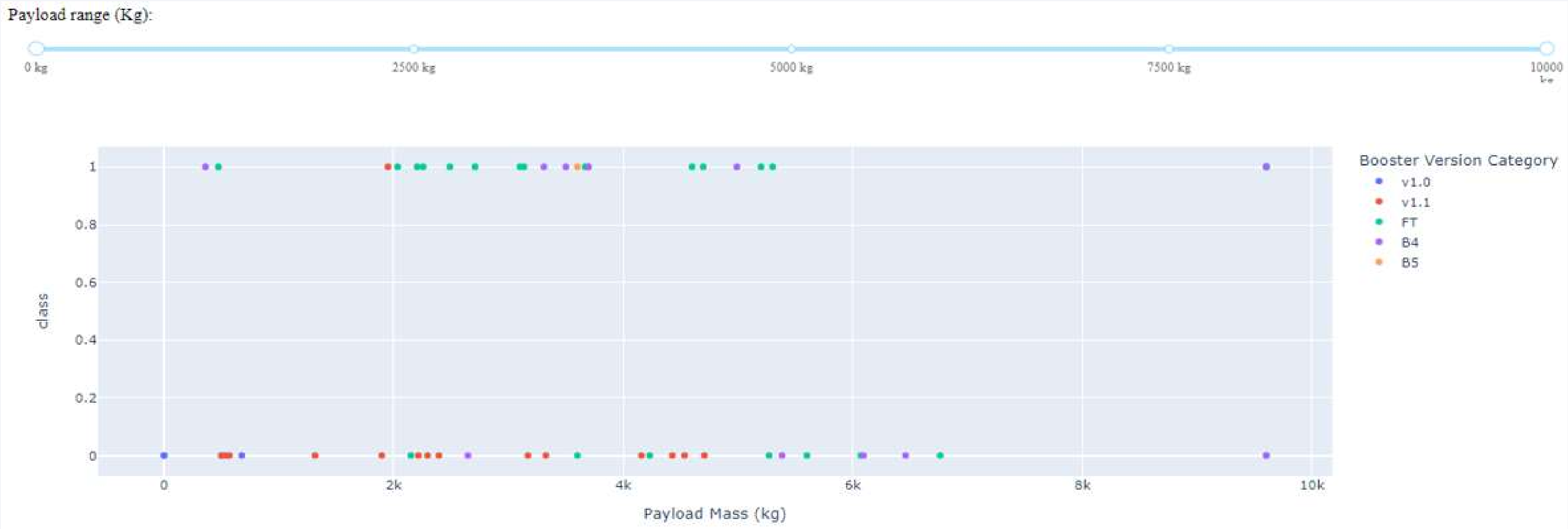
Site with Highest Launch Success Ratio



- KSC LC-39A had the highest launch success ratio, with 76.9% success, and 23.1% failure rate.

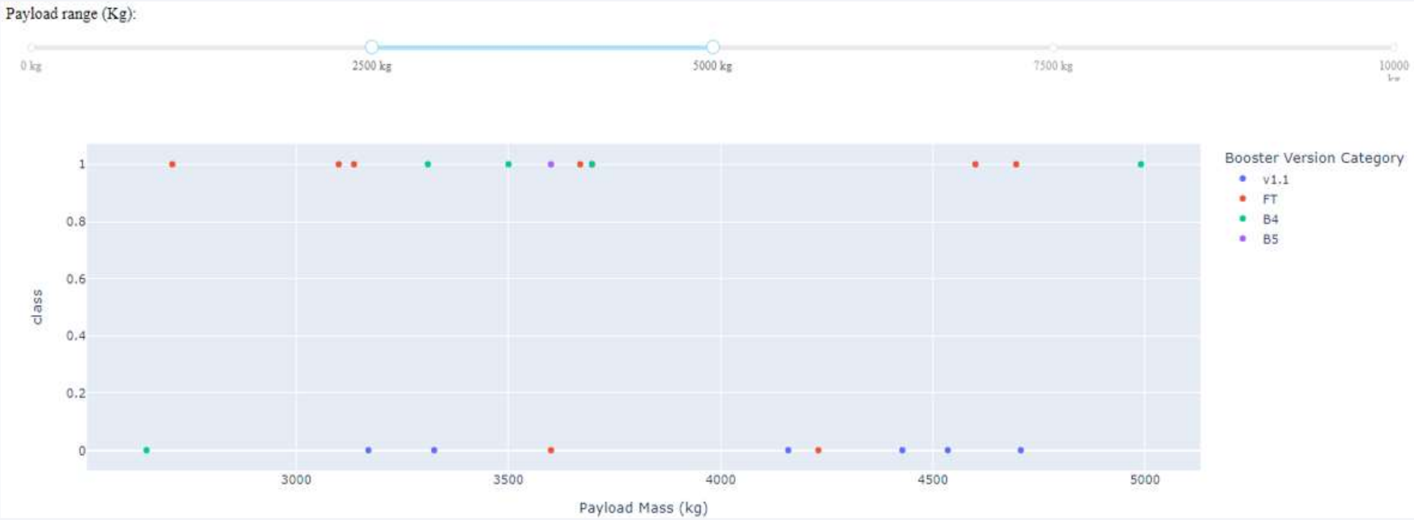
Payload vs. Launch Outcome Scatter Plot

Full Payload Range

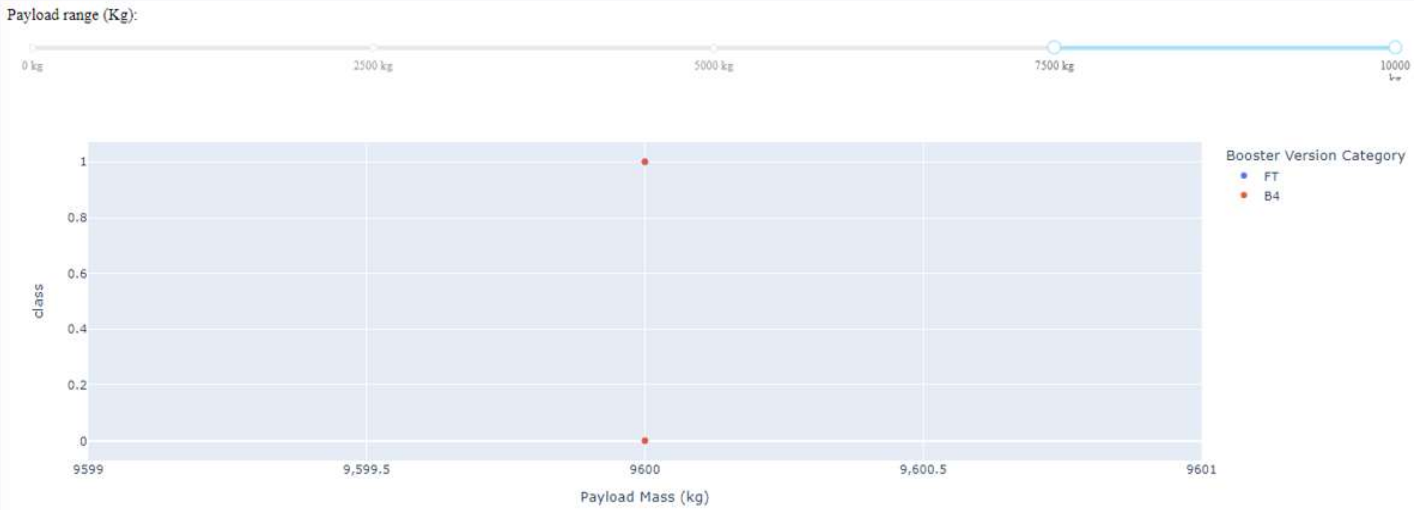


From the full Payload Range, it is observed that most launches occurred between 2500 kg and 5000 kg

Payload vs. Launch Outcome Scatter Plot



2500kg to 5000kg Payload had the most launches.

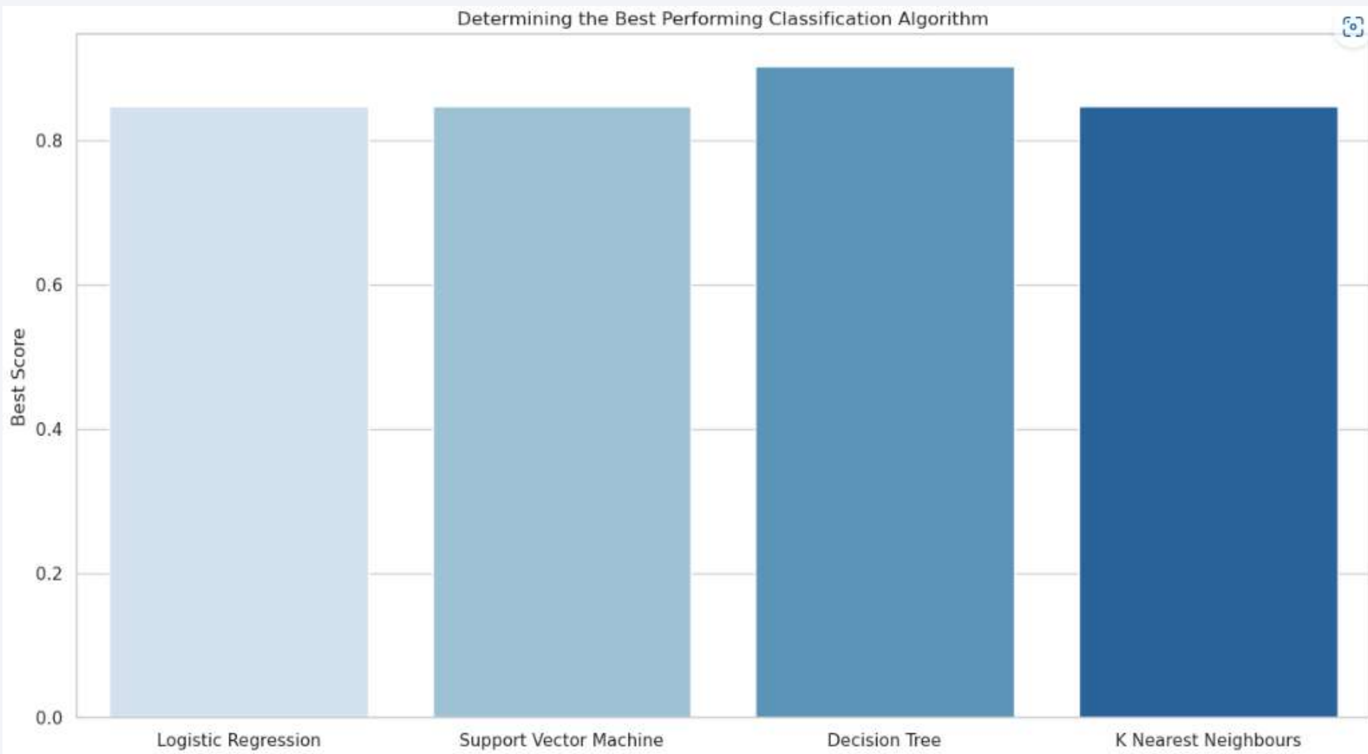


7500kg to 10000kg Payload had the least launches (only 2).

Section 5

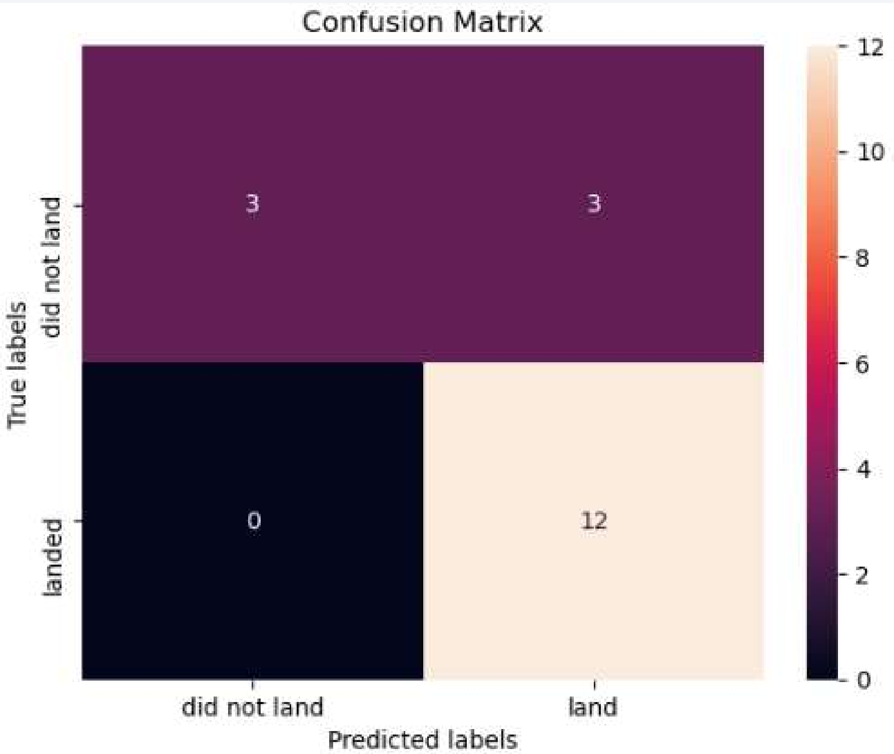
Predictive Analysis (Classification)

Classification Accuracy



- The model with the highest classification accuracy is the Decision Tree, with an accuracy of 0.9028

Confusion Matrix



Decision Tree Confusion Matrix

- The model predicted 12 successful landing when the true label was successful landing (True Positive).
- The model predicted 3 unsuccessful landing when the true label was unsuccessful landing (True Negative).
- The model predicted 3 successful landing when the true label was unsuccessful landing (False Positive).
- The model predicted 0 unsuccessful landing when the true label was successful landing (False Negative).

Conclusions

- KSC LC-39A had the most successful launches, with a 76.9% success and 23.1% failure rate.
- Booster version FT had the highest success rate, and most successful launches occurred between 2500kg and 5500kg payload.
- The decision tree model was the best model to predict the probability of successful landing, with an accuracy of 90.28%.
- The success rate increases as the year increases.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project


Thank you!