IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Lim Chin Pei>
<03 December 2023>

Outline

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

Executive Summary

- Summary of methodologies
  - Collect data using SpaceX REST API and web scrapping from Wikipedia
  - Process data by filtering data, replace missing value and convert data into Dataframe
  - Wrangling data to create training labels for different landing outcomes
  - Explore and analyze data by visualizing it with various chart type
  - Analyze data with SQL
  - Visualize and explore the launch site success rate and its proximity to different location
  - Visualize variables and its landing outcome using an interactive dashboard
  - Predict landing outcome by various model
- Summary of all results
  - Flight number, payload mass and orbit type are correlated to landing outcomes
  - Decision tree model is the best model

Introduction

Background

- SpaceX, a leader in space industry, managed to launch a rocket with a lower cost ($62mil) compared to its competitors ($165mil), as Falcon 9 (by SpaceX) could reuse its first stage.
- As a competitor to SpaceX, our company, Space Y would like to determine the price of each launch by predicting if the first stage will land.
- To achieve the objective, we have conducted a project to train a machine learning model and use data to predict if the first stage will land successfully.

Problem Statement

- What are the attributes correlated with successful landing?
- What is the best model to predict if the first stage will land successfully?

Section 1

Methodology

Methodology

Executive Summary

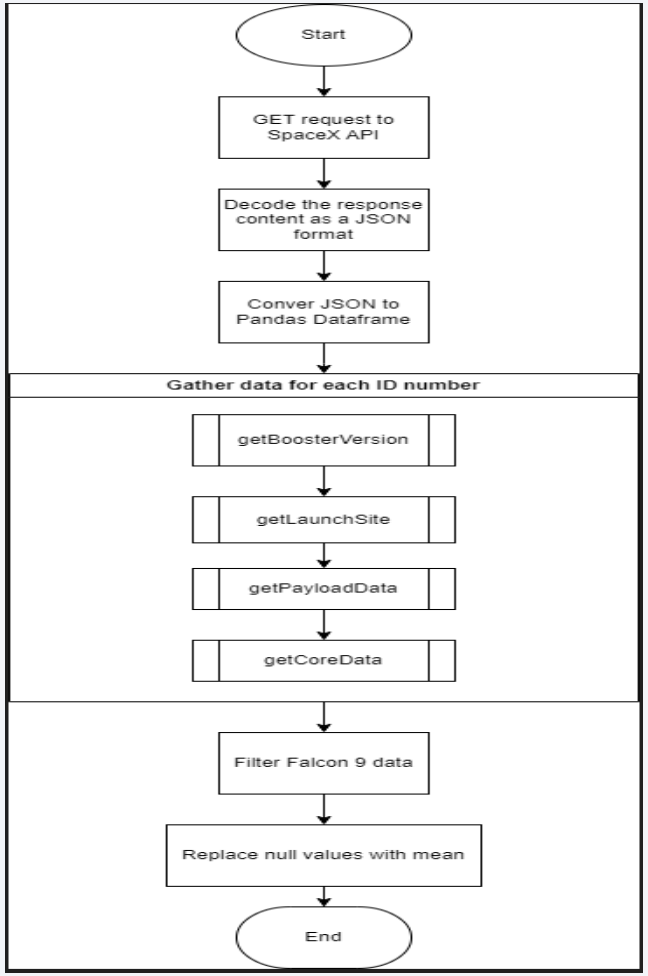
- Data collection methodology:
  - Data was collected from SpaceX REST API and Wikipedia
  - Using API targeting another endpoint to gather specific data for each ID number, replace missing value with mean and to include Falcon 9 data only in our sampling data
- Perform data wrangling
  - Convert outcome into training labels
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection – SpaceX API

- Request rocket launch data from SpaceX API using GET request
- Decode the response content as a JSON
- Convert JSON into Pandas Dataframe using json_normalize function
- For data containing ID only, we will use API targeting another endpoint to gather specific data for each ID number via user defined function (getBoosterVersion, getLaunchSite, getPayloadData, getCoreData)
- Filter Falcon 9 data as our sampling data
- Replace null values with mean

Data Collection – SpaceX API

- GitHub URL: https://github.com/chinpei199/IBMDataScience_Falcon9/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



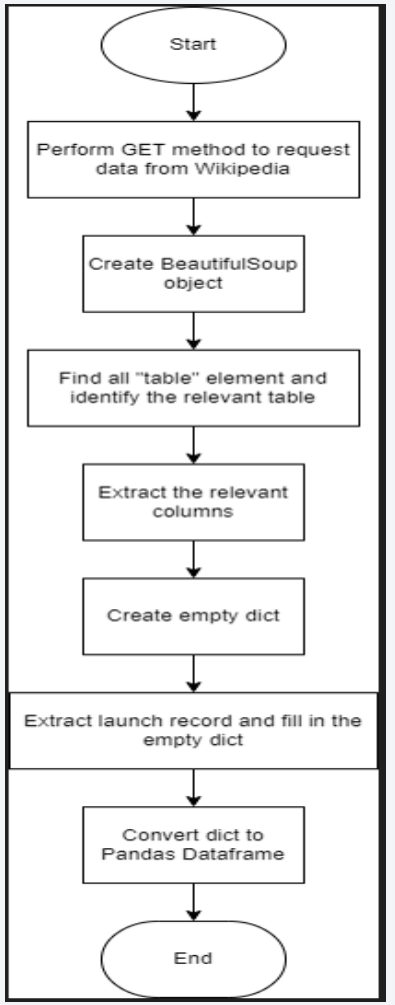
Flowchart of SpaceX API data collection

Data Collection – Web Scrapping

- Perform GET method to request Falcon 9 data from Wikipedia
- Create BeautifulSoup object from the HTML response
- Use find_all function in the BeautifulSoup object with element type “table” and identify the relevant table
- Extract the relevant columns:
  - Flight No, Date and time, Launch site, Payload, Payload mass, Orbit, Customer, Launch outcome
- Create empty dictionary with keys from the column name above
- Extract the launch record from the table rows and fill in dictionary
- Convert the dictionary into a Pandas Dataframe

Data Collection - Scraping

- GitHub URL: https://github.com/chinpei199/IBMDataScience_Falcon9/blob/main/jupyter-labs-webscraping.ipynb

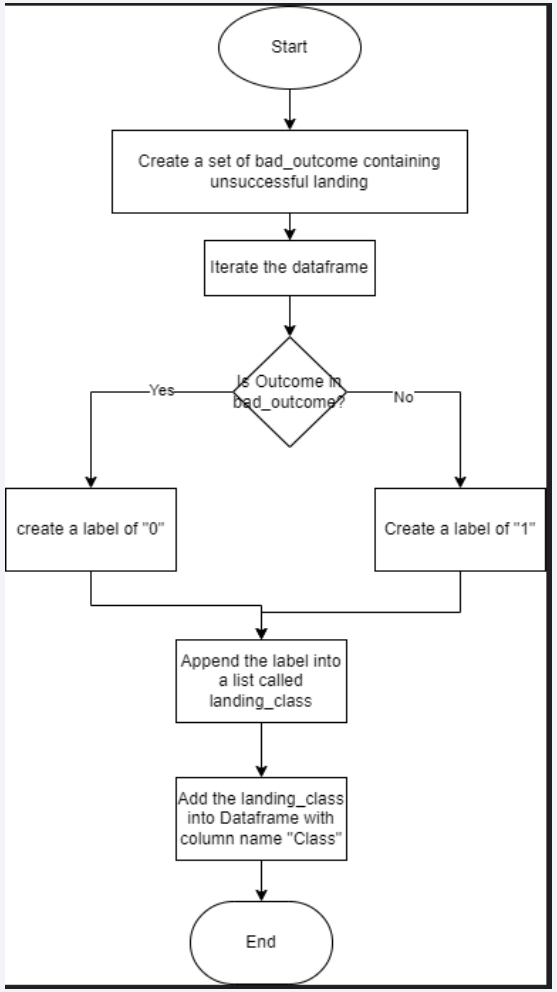


Flowchart of web scrapping

Data Wrangling

Steps:

- Create a set of outcomes where the stage did not land successfully, assigned to variable bad_outcomes
- Iterate the dataframe, create a list where the element is 0 if the corresponding row in Outcome is in the set bad_outcome, otherwise it is 1
- Assign the list to landing_class and add to Dataframe with column name “Class”
- GitHub URL: https://github.com/chinpei199/IBMDataScience_Falcon9/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



Flowchart of Data Wrangling

EDA with Data Visualization

Type of chart	Data display	Analysis
Scatter chart	FlightNumber, Payload, Class	To analyze how the Flight Number and Payload variables would affect the launch outcome
Scatter chart	FlightNumber, LaunchSite, Class	Visualize the relationship between Flight Number and Launch Site
Scatter chart	Payload, LaunchSite, Class	Visualize the relationship between Payload and LaunchSite
Bar chart	Success rate, Orbit	Visualize the relationship between success rate of each orbit type
Scatter chart	FlightNumber, Orbit, Class	Visualize the relationship between Flight Number and Orbit type
Scatter chart	Orbit, Payload, Class	Visualize the relationship between Payload and Orbit type
Line chart	Success Rate, Date	Visualize the launch success yearly trend

- GitHub URL: https://github.com/chinpei199/IBMDataScience_Falcon9/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- SQL queries:
  - Display the names of the unique launch sites
  - Display 5 records where launch sites begin with ‘CCA’
  - Display 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 achieved
  - List the names of the booster which have success in drone ship and 4000 < payload mass <6000
  - List the total number of successful and failure mission outcome
  - List the names of the booster_versions which have carried the maximum payload mass
  - List the records of month, failure landing_outcomes, booster version, launch_site for the months in year 2015
  - Rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20
- GitHub URL: https://github.com/chinpei199/IBMDataScience_Falcon9/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Marking on map
  - Create yellow circle highlighting NASA Johnson Space Center and red circle for all launch sites, as well as adding a pop up label showing the sites' name, in order to provide intuitive insights about where are those launch sites.
  - Mark the launch outcome on each sites, we use green for successful launch and red for unsuccessful launch in order to see which sites have high success rates
  - Draw polyline between the launch site to the nearest coastline and city, as well as adding pop up label to show the distance in order to analyze the proximities of launch sites.
- GitHub URL: https://github.com/chinpei199/IBMDataScience_Falcon9/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

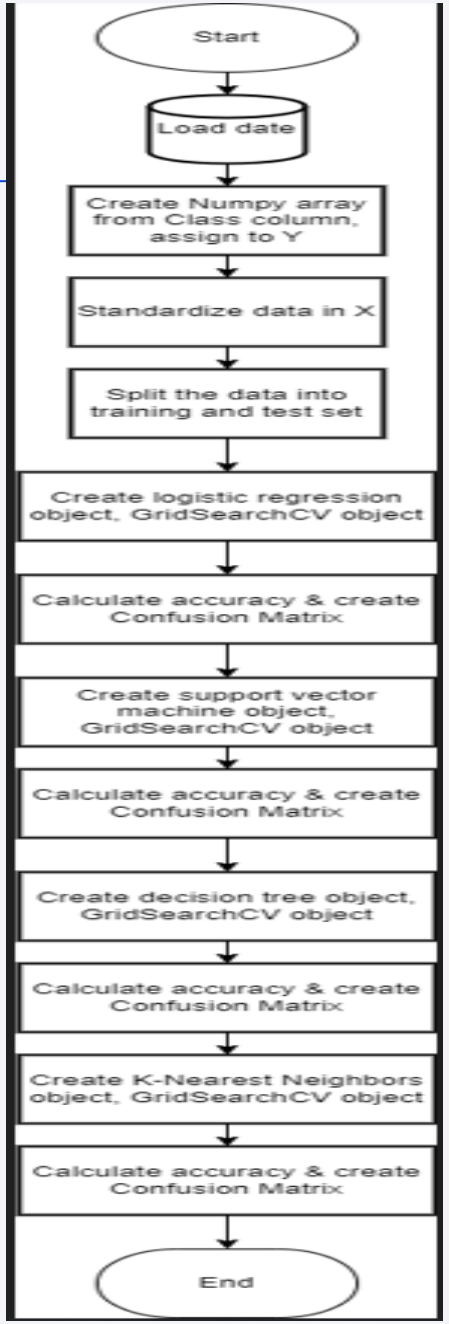
Build a Dashboard with Plotly Dash

- Dropdown list with option to choose all launch sites or a specific launch site
  - Allow user to select all launch site or a specific launch site
- Pie chart showing successful launch
  - Allow user to view the percentage of successful and unsuccessful launch based on the option selected in the dropdown list
- Slider of payload mass range
  - Allow user to select the range of the payload mass
- Scatter chart showing the payload mass and the launch outcome
  - Allow user to visualize and analyze the relationship between the payload mass and the launch outcome for each launch site
- GitHub URL: https://github.com/chinpei199/IBMDataScience_Falcon9/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

Steps:

- Load the Dataframe
- Create Numpy array from the column Class and assign it to variable Y
- Standardize the data in X with StandardScaler and fit_transform the data
- Split the data into training and test data using the function train_test_split
- Create a logistic regression object, a GridSearchCV object with cv=10, fit the object to find the best parameters, then calculate the accuracy using score method as well as create the confusion matrix
- Perform the same steps for support vector machine, decision tree and K-Nearest Neighbors.
- Identify the best model based on accuracy
- GitHub URL: https://github.com/chinpei199/IBMDataScience_Falcon9/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb



Results

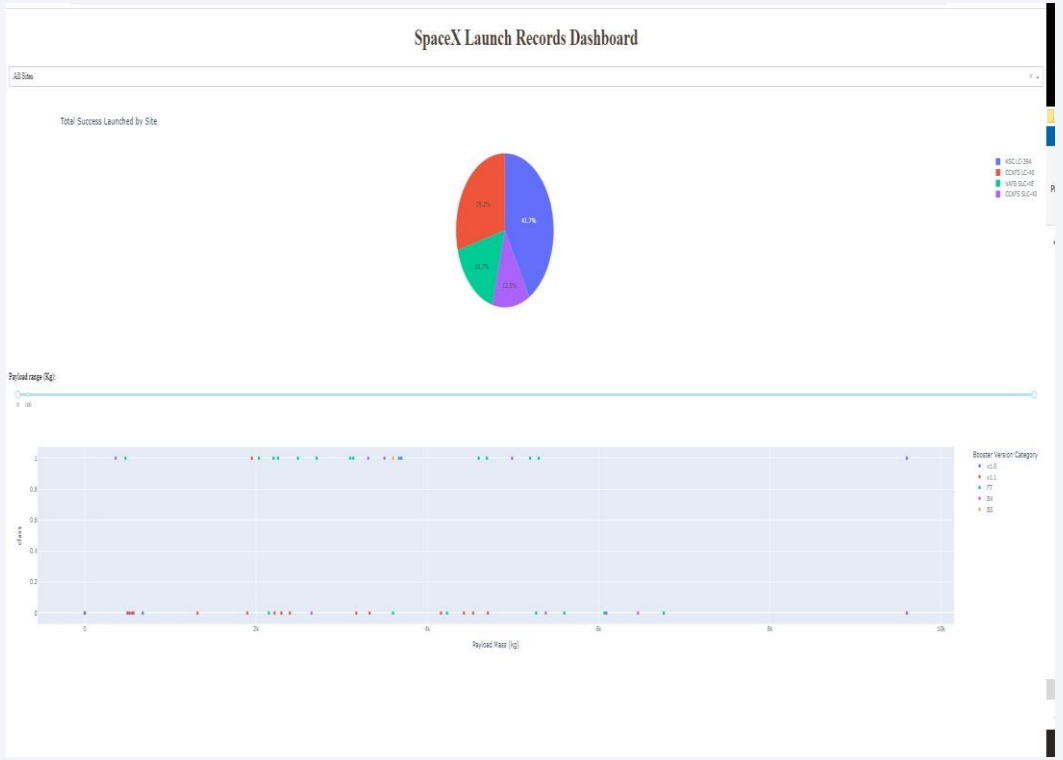
Exploratory data analysis results

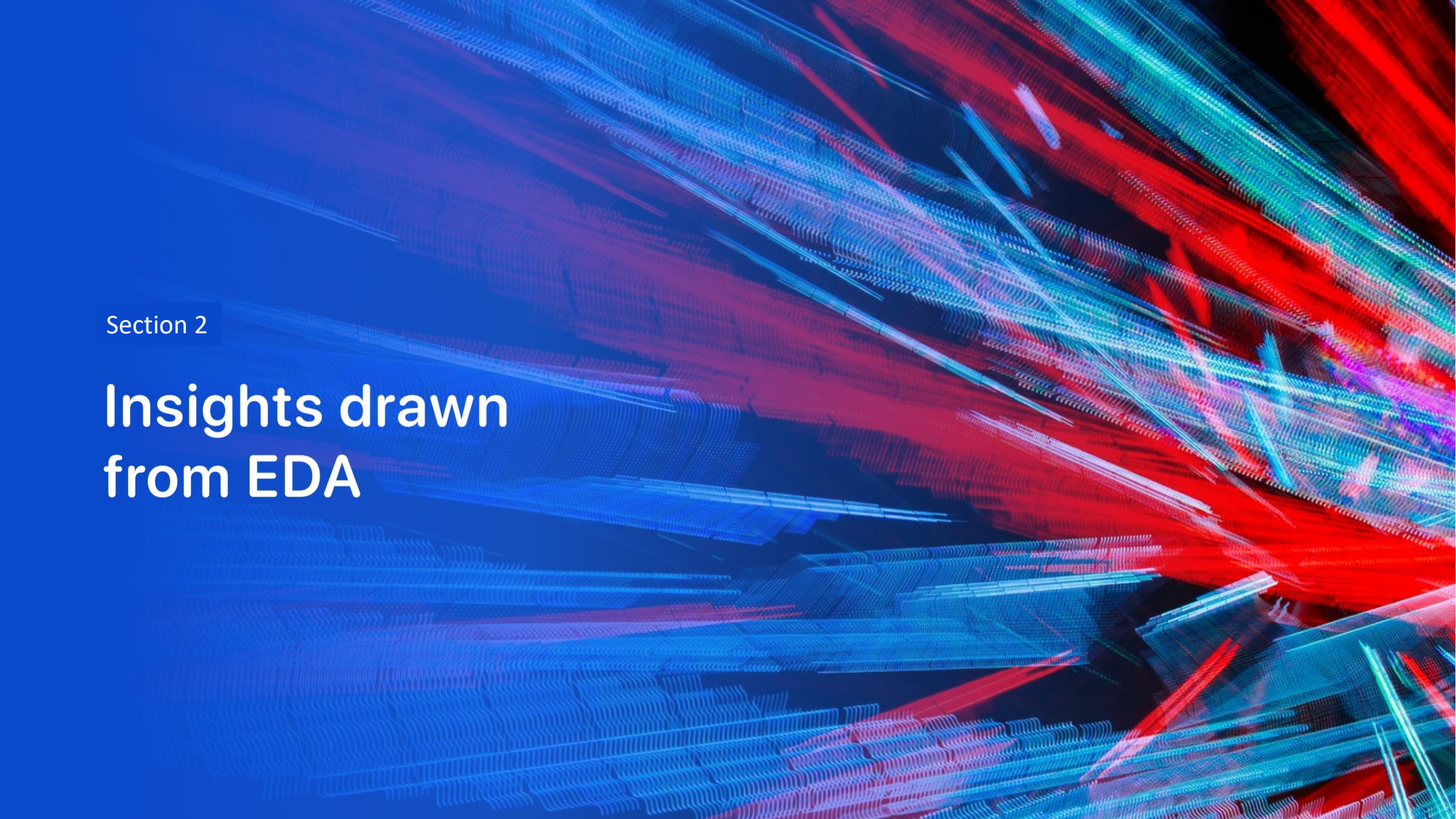
- KSCLC-39A and VAFBSLC-4E has higher success rate of 77%
- ES-L1, GEO, HEO and SSO orbit have high success rate
- The success rate improved over the years

Predictive analysis results

- Decision tree model is the best predictive model with higher accuracy

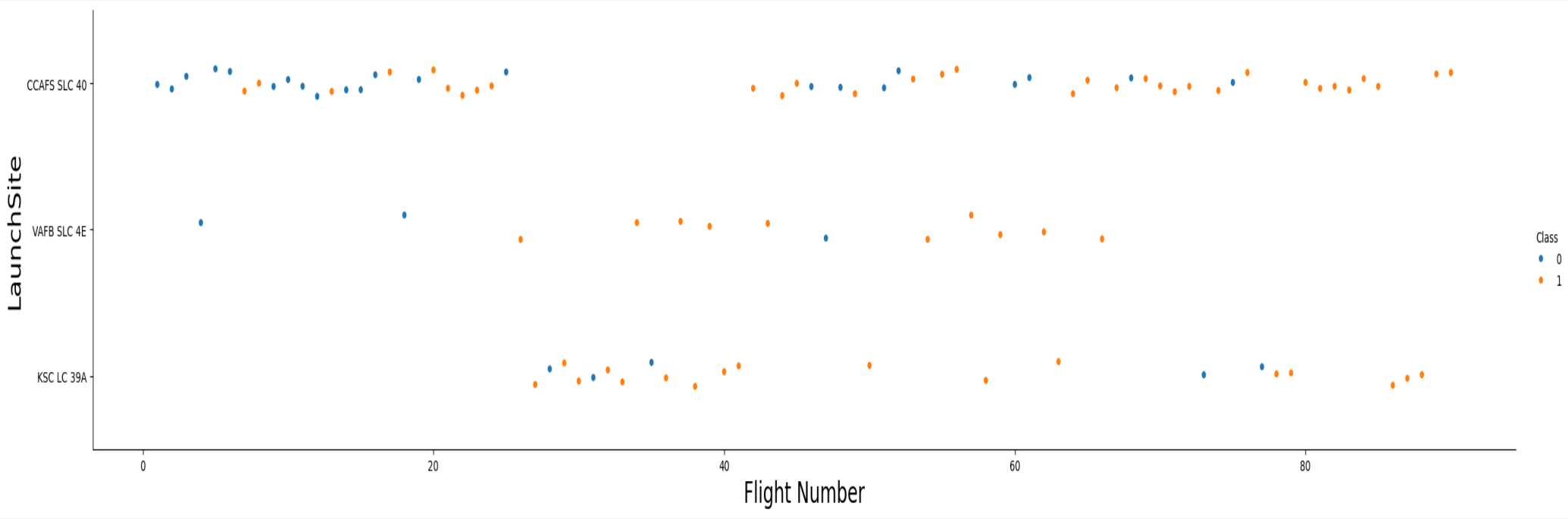
Interactive analytics demo in screenshots




Section 2
Insights drawn from EDA

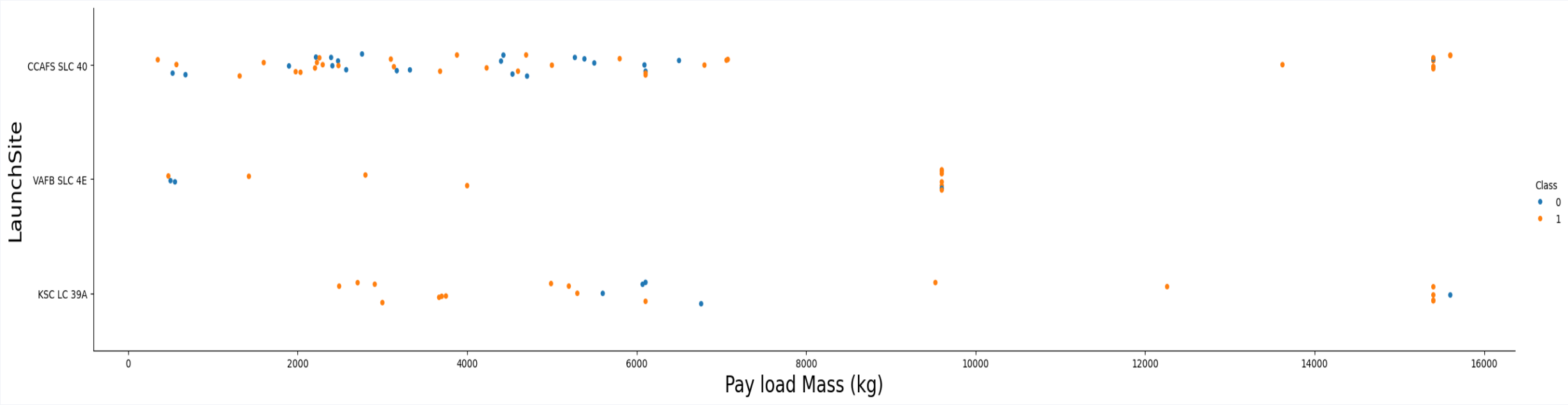
Flight Number vs. Launch Site

- From the scatter plot, we can observed that as the flight number increases, the first stage is more likely to land successfully.



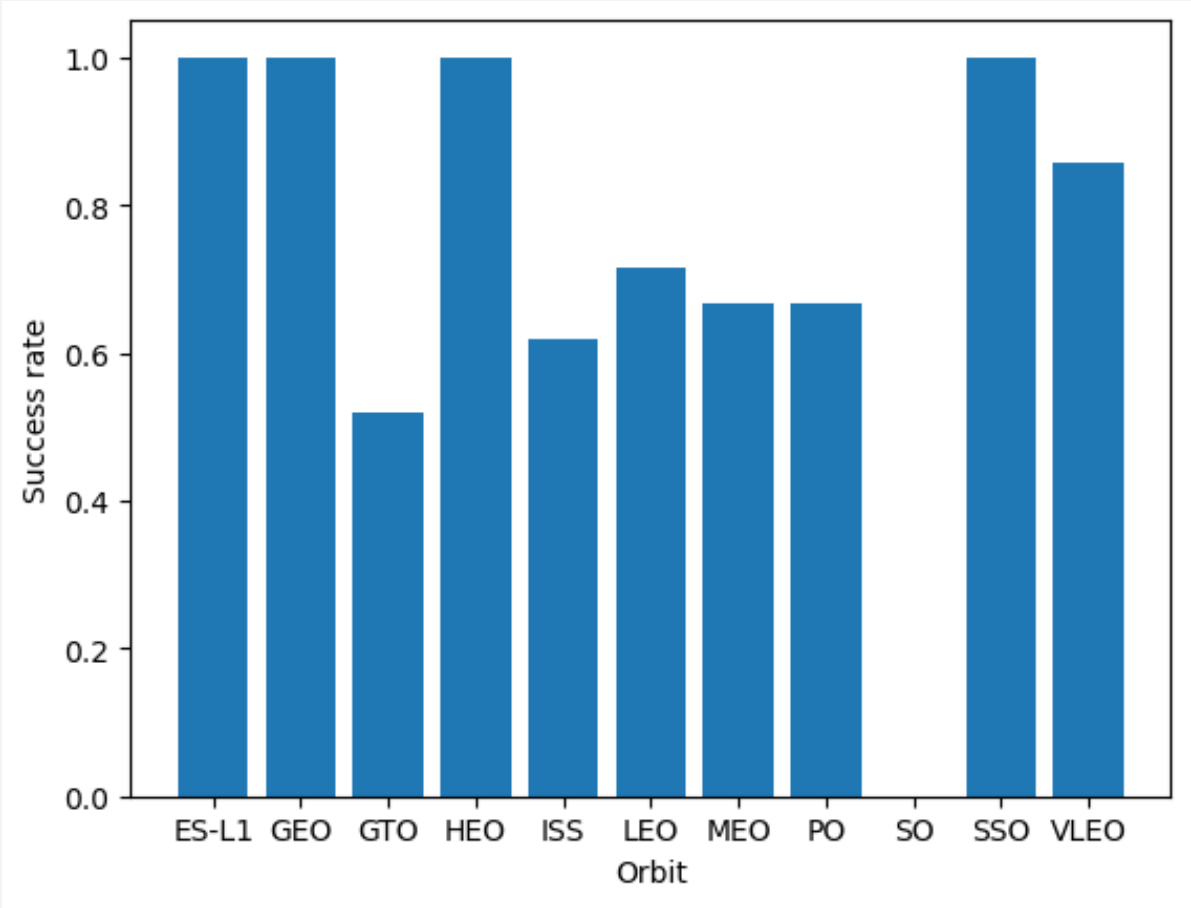
Payload vs. Launch Site

- For VAFB-SLC launch site, there are no rockets launched for heavy-payload mass (greater than 10,000kg)
- The higher the payload mass, the higher the success rate
- KSC LC39A has higher successful rate for payload mass less than 4,000 kg



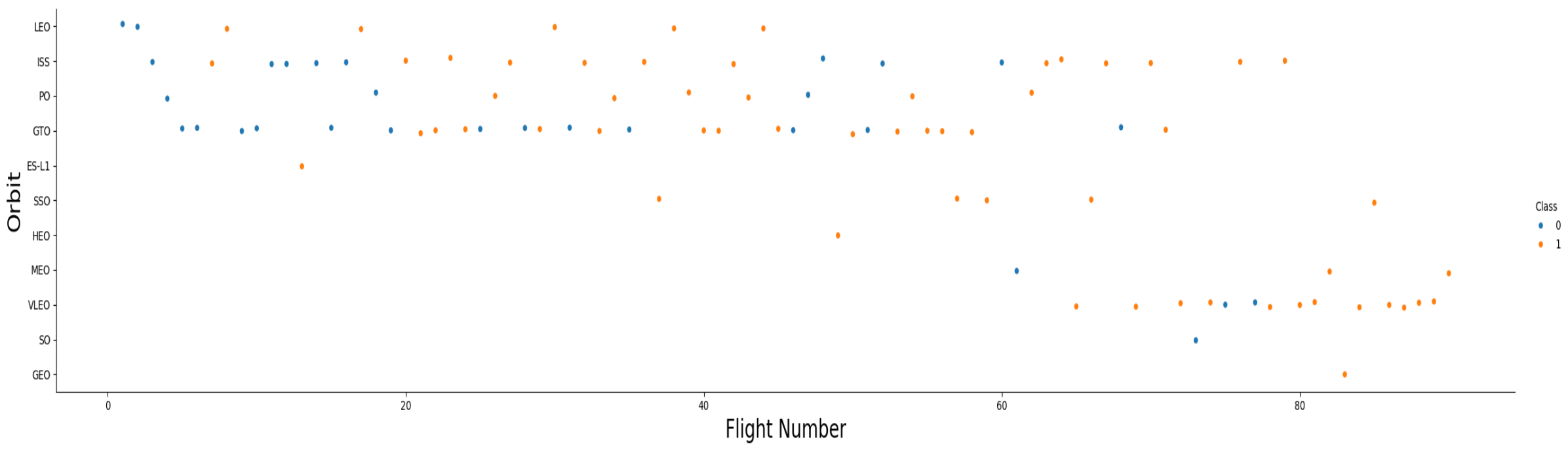
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO and SSO has higher success rate, while SO has very low or no success rate
- GTO, ISS, LEO, MEO,PO,VLEO has success rate ranging from approximately 50% - 80%



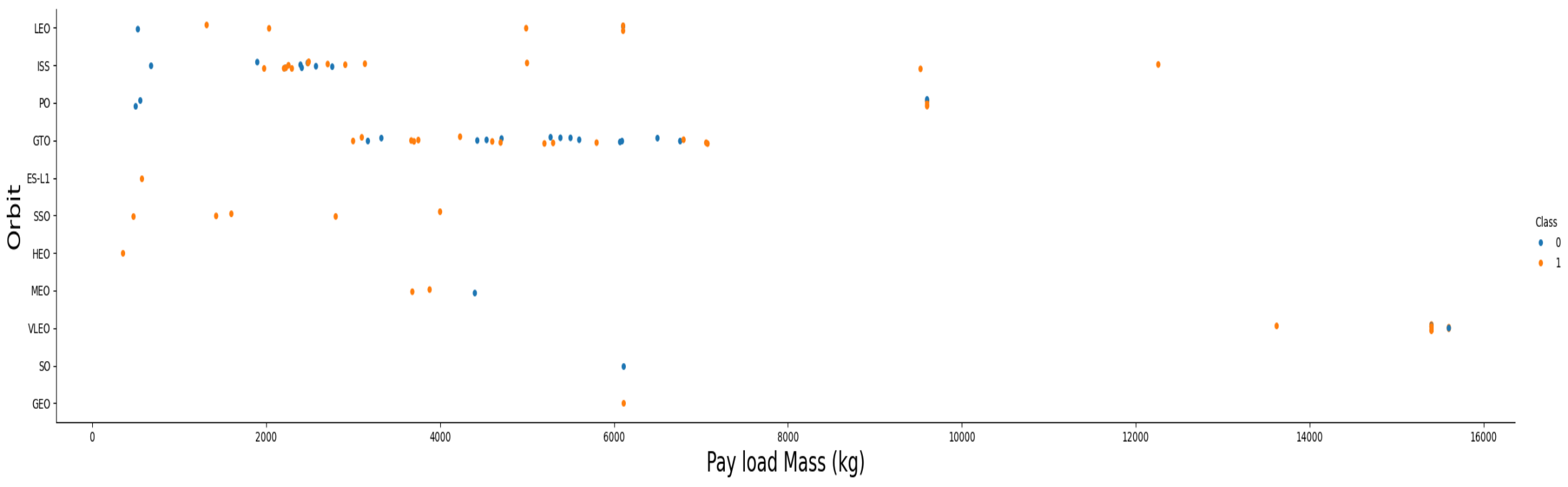
Flight Number vs. Orbit Type

- In the LEO and VLEO orbit, the success rate appears to be related to the number of flights
- There is no relationship between flight number and GTO orbit



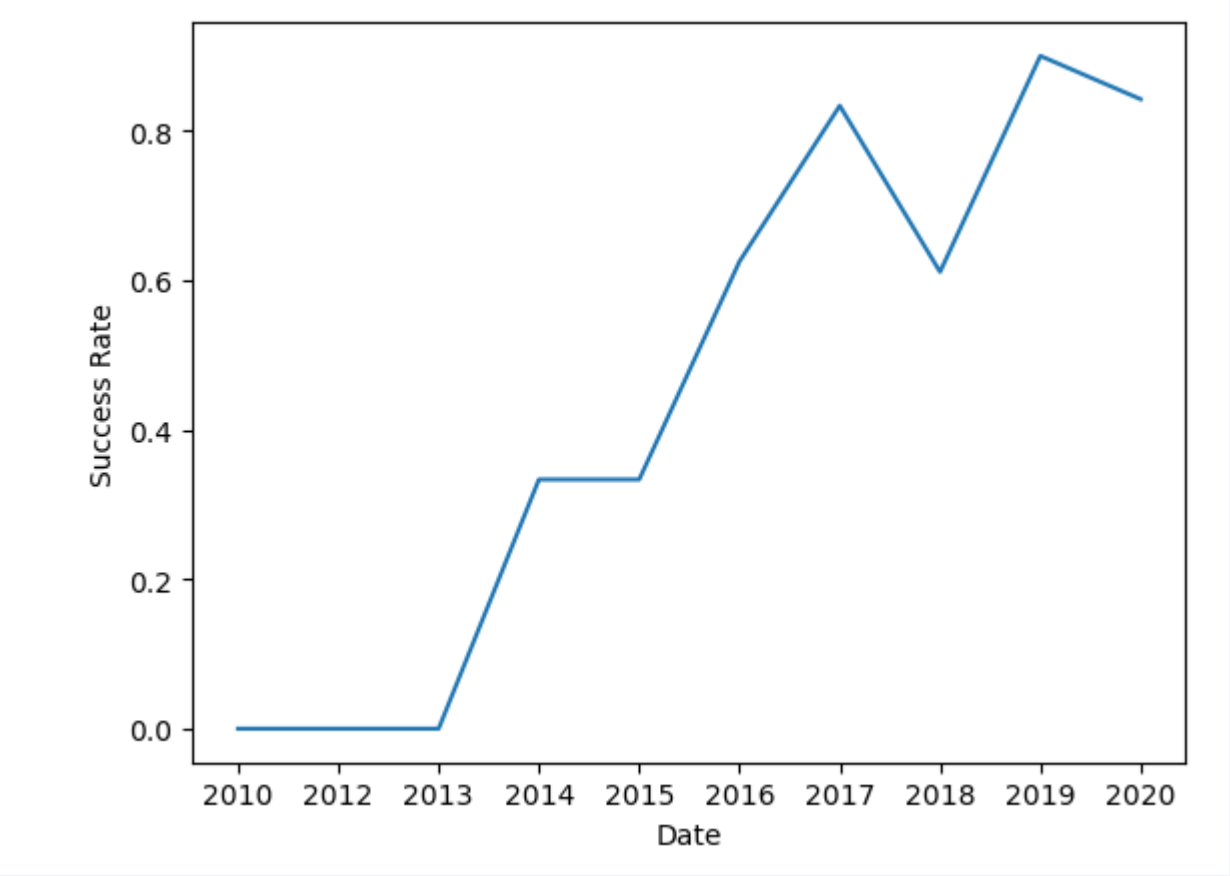
Payload vs. Orbit Type

- With heavy payloads the successful landing rate are more for PO, LEO and ISS.
- SSO and HEO has higher success rate for lighter payload mass (<4000kg)



Launch Success Yearly Trend

- The success rate has improved since 2013



All Launch Site Names

- We use the keyword DISTINCT to show the launch site
- Name of all launch site
  - CCAFS LC-40
  - VAFB SLC-4E
  - KSC LC-39A
  - CCAFS SLC-40

```
In [11]: %%sql

         SELECT DISTINCT Launch_Site FROM SPACEXTBL

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

Out[11]:

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

Launch Site Names Begin with 'CCA'

- We use the keyword LIKE and LIMIT to filter launch site names begin with ‘CCA’ and limit the data to 5 records

Display 5 records where launch sites begin with the string 'CCA'

In [14]:
```
%%sql
SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

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

Out[14]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- We calculate the total payload mass using SUM function and filter the customer using WHERE
- The total payload mass is 45596kg

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

In [17]:
```
%%sql

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

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

Out[17]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- We use the AVG function to calculate the average of payload mass and using WHERE keyword to filter the booster version
- The average payload mass carried by booster version F9v1.1 is 2928.4kg

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

In [23]:
```
%%sql
SELECT AVG (PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version == 'F9 v1.1'
```

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

Out[23]:

AVG (PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- The first successful landing outcome in ground pad is 2015-12-22

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

[13]:
```
%%sql

SELECT MIN(Date) FROM SPACEXTBL WHERE Landing_Outcome ='Success (ground pad)'
```

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

[13]:

MIN(Date)
2015-12-22

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
  - F9 FT B1022
  - F9 FT B1026
  - F9 FTB1021.2
  - F9 FTB1031.2
- We use the WHERE keyword to filter the landing_outcome and operator “<“ and “>” to filter the payload mass

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

[15]:
```
%%sql

SELECT Booster_Version FROM SPACEXTBL WHERE Landing_Outcome ='Success (drone ship)' AND 4000 < PAYLOAD_MASS__KG_ AND PAYLOAD_MASS__KG_<6000
```

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

[15]:

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 total number of success mission outcome is 100 whereas the failure mission outcome is 1
- We use the COUNT and GROUPBY keyword to calculate the number of successful and failure mission outcome

List the total number of successful and failure mission outcomes

[38]:
```
%%sql

SELECT Mission_Outcome, COUNT(*) FROM SPACEXTBL GROUP BY Mission_Outcome
```

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

[38]:

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

Boosters Carried Maximum Payload

- We use the MAX function in the subquery to find out the maximum payload mass, then filter the booster version that matched the maximum payload mass using WHERE keyword

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

[41]:
```
%%sql

SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ == (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

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

[41]:

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

- We filter the year and landing outcome using WHERE keyword and SELECT the relevant columns
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

[43]:
```
%%sql

SELECT substr(Date,6,2), Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTBL WHERE substr(Date,0,5) = '2015' AND Landing_Outcome =='Failure (drone ship)'
```

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

[43]:

substr(Date,6,2)	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

- We filter the Date using WHERE and perform calculation using COUNT function, then arrange the data in descending order using ORDER BY
- 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

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.

[21]:
```
%%sql

SELECT Landing_Outcome, COUNT(*) AS Count FROM SPACEXTBL WHERE Date >= '2010-06-04' AND Date <= '2017-03-20' GROUP BY Landing_Outcome ORDER BY Count DESC
```

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

[21]:

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

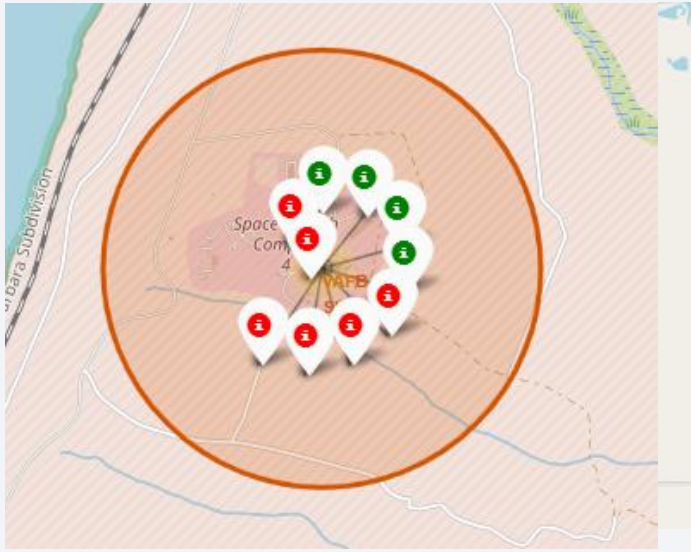
Launch Sites Proximities Analysis

Launch Sites

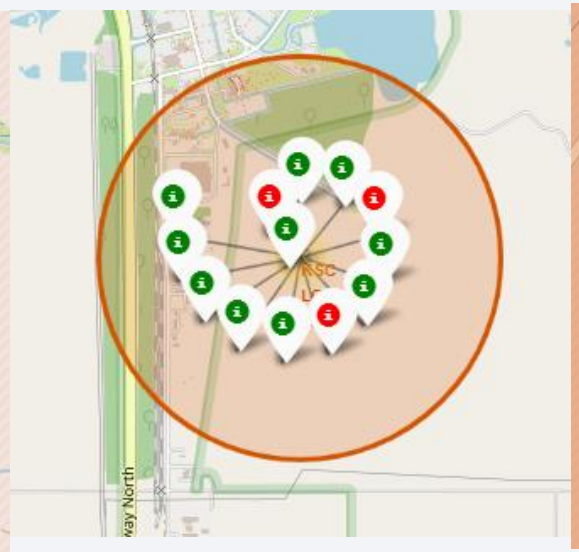
- All launch sides are not close to the equator but very close to the coast as the proximity to the coast provides a safe area for rocket stages to fall back into the ocean during launches.

Launch Outcomes

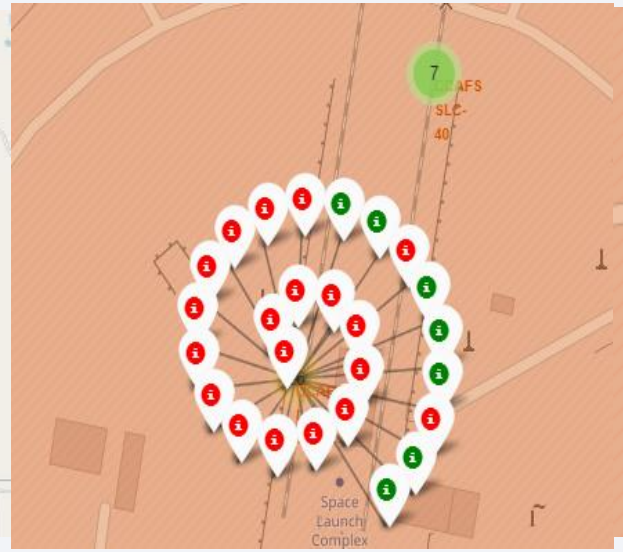
- Green marker shows successful launch and red marker show unsuccessful launch



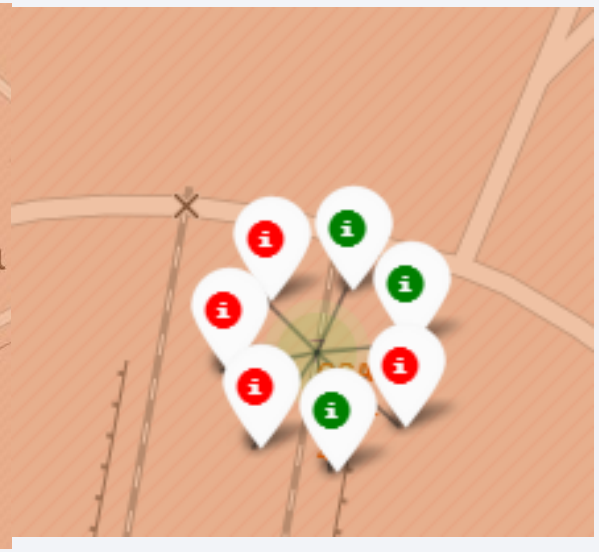
VAFB SLC-4E



KSC LC-39A



CCAFS LC-40

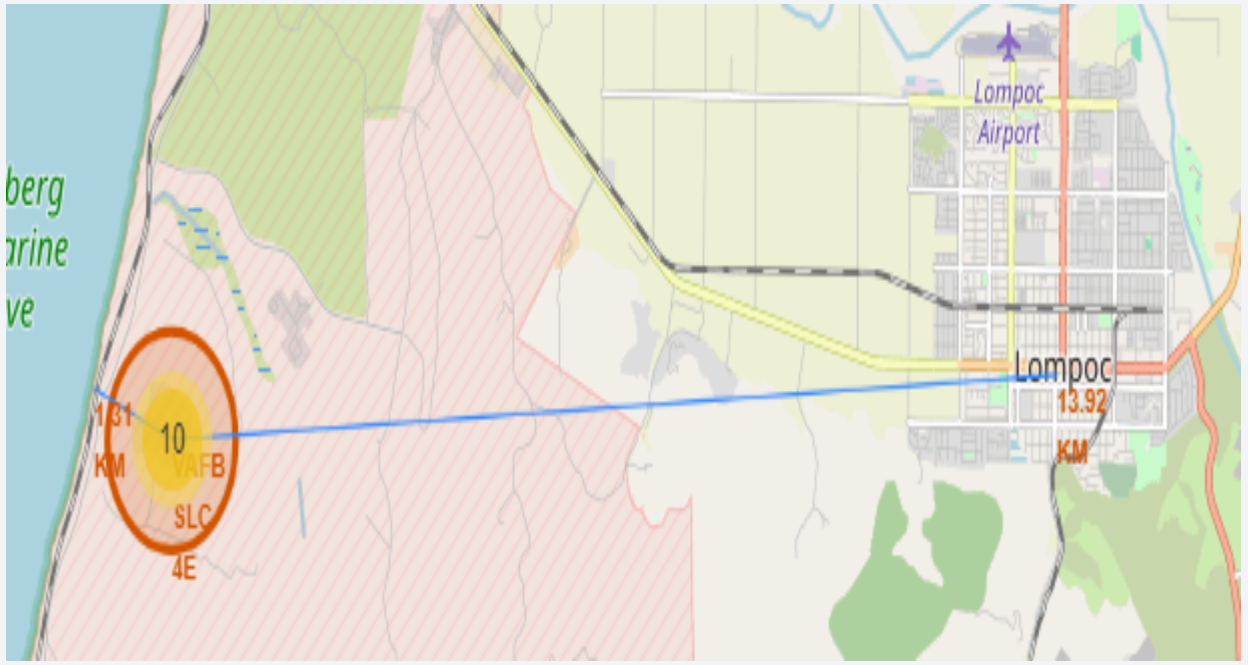


CCAFS SLC-40

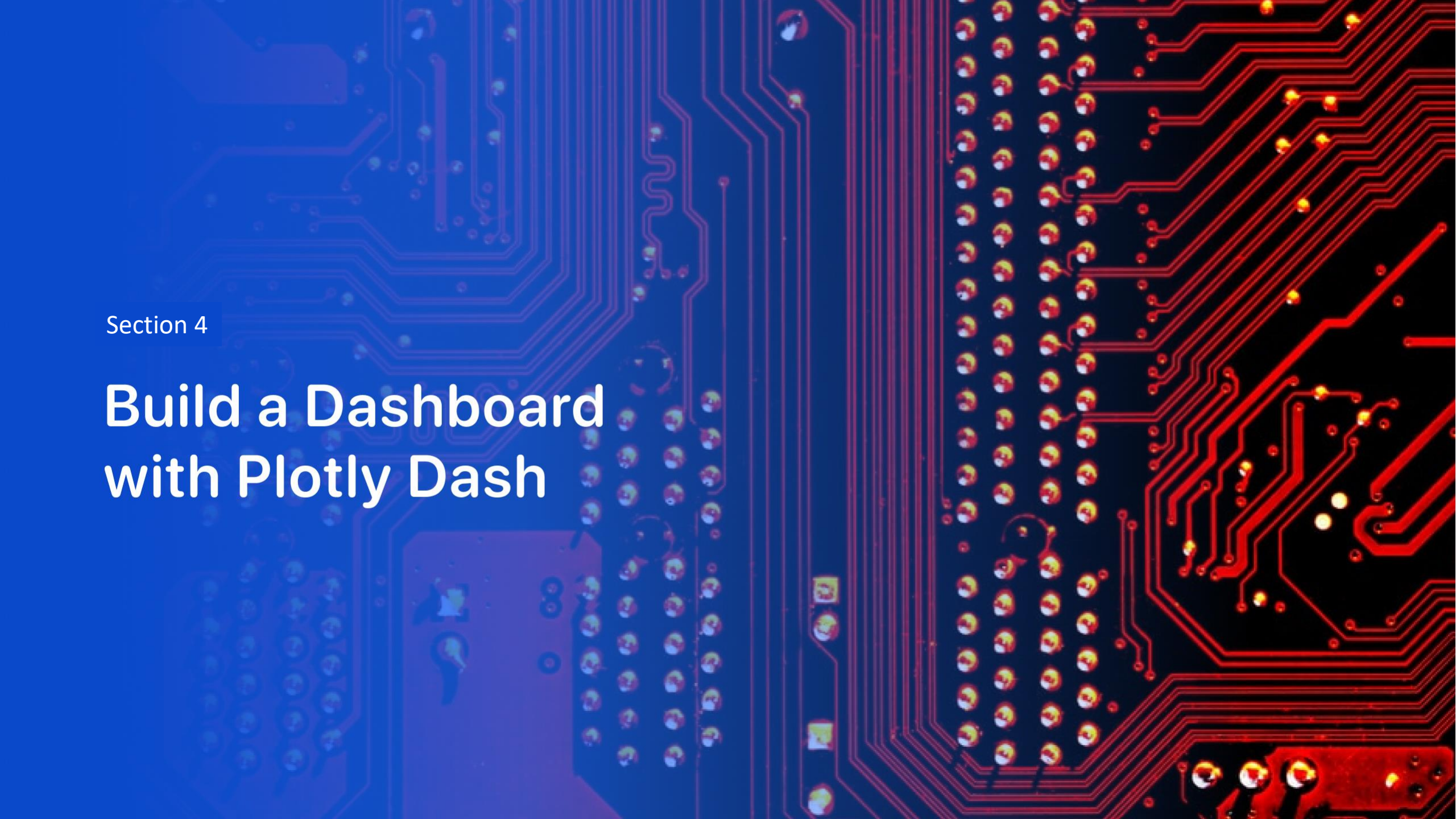
Launch Site Distance to Proximities

- The launch site is close to the coastline but very far from the city most likely due to safety concern in the event of a launch failure, the rocket can be directed over the ocean, reducing the risk to populated areas.

Distance to coastline

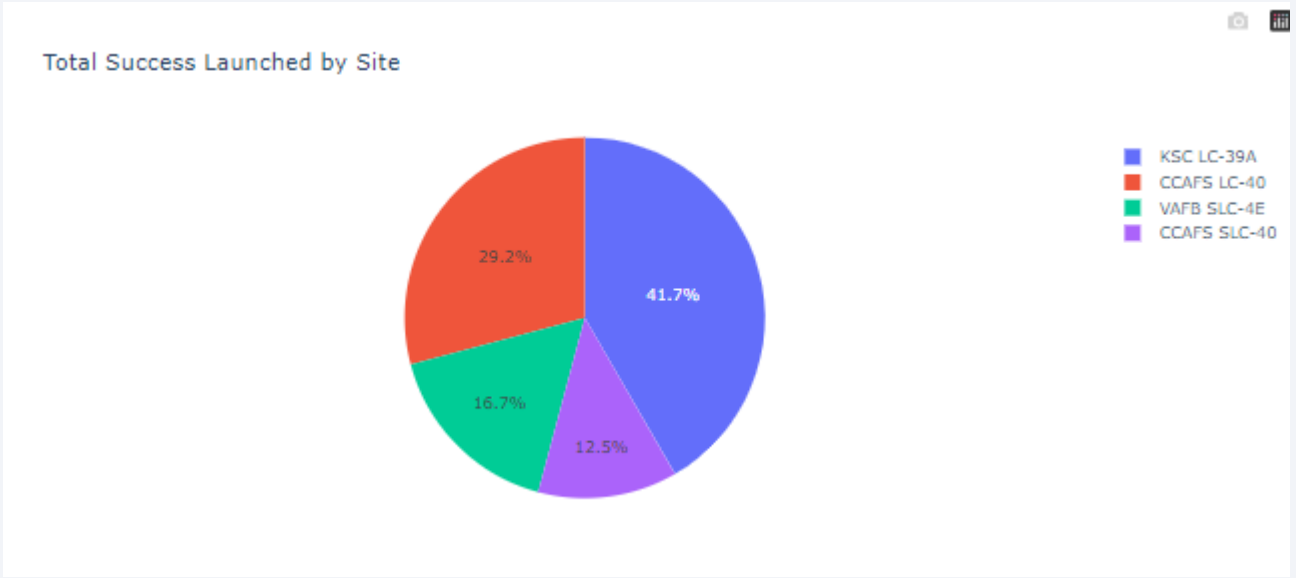


Distance to city


Section 4
Build a Dashboard with Plotly Dash

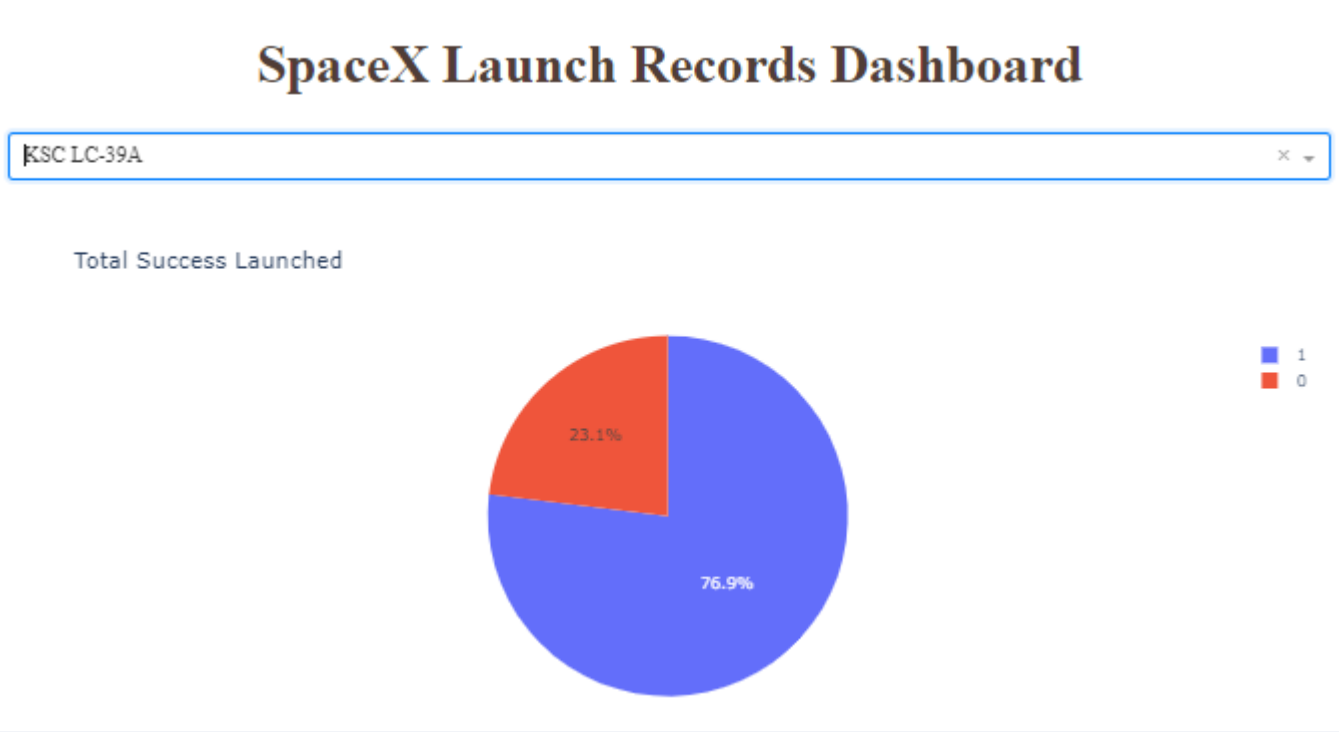
Success Launch Count

- KSC LC-39A has the most successful launch (41.7%) among the 4 launch sites.



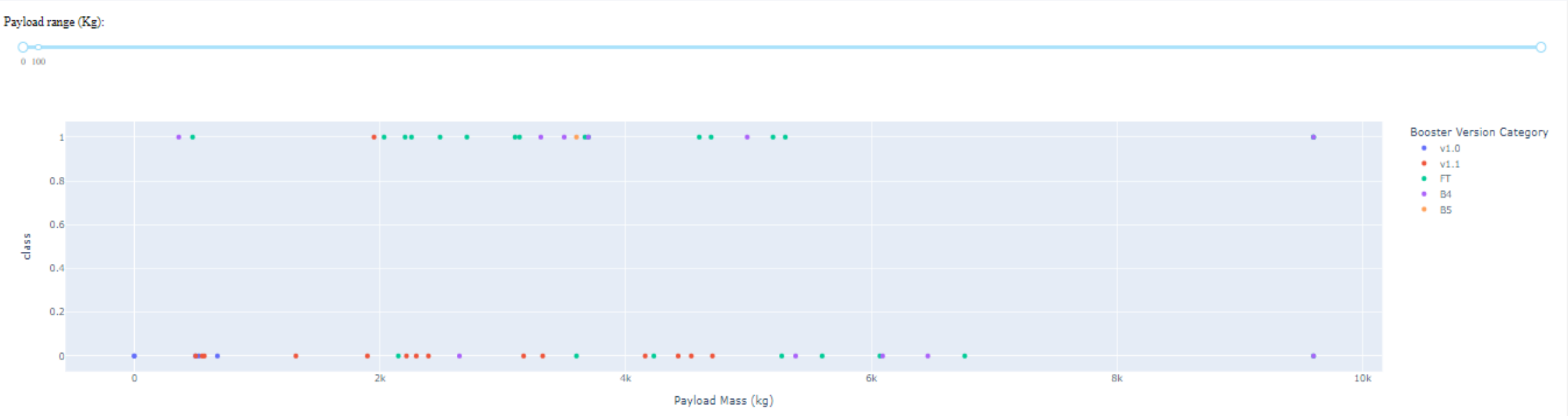
Launch Site with highest success ratio

- KSC LC-39A has the highest launch success ratio of 76.9% among the 4 launch site.



Payload Mass Vs Launch Outcome

- Payload mass between 2,000 kg – 4,000 kg has the higher success launch
- Payload mass between 6,000kg – 8,000 kg has the highest failure rate
- 1 indicate a success launch whereas 0 indicate an unsuccessful launch



Section 5

Predictive Analysis (Classification)

Classification Accuracy

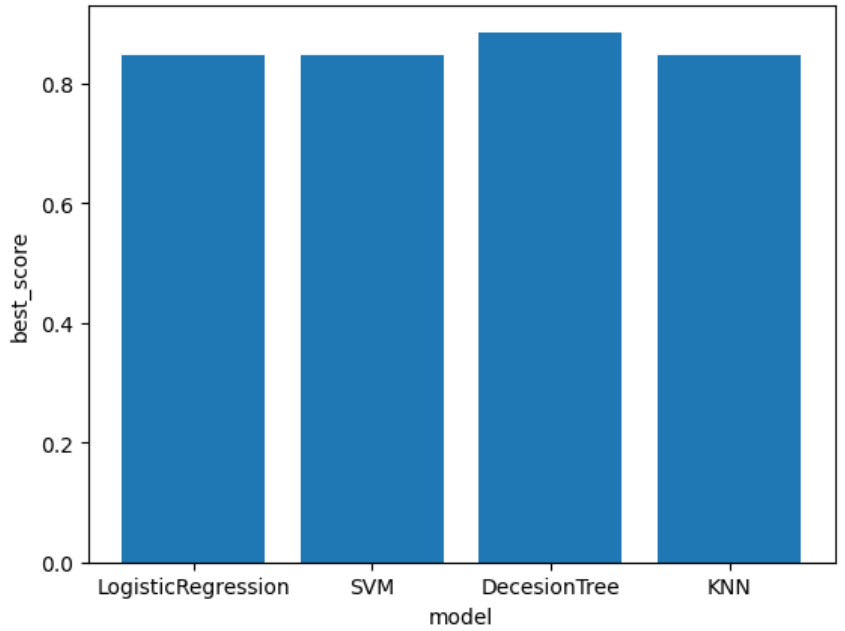
- Decision tree model has the highest accuracy among the 4 models.

```
[38]: # Find which model has the highest classification accuracy

models = {'model':['LogisticRegression', 'SVM','DecesionTree','KNN'],
          'best_score':[logreg_cv.best_score_ , svm_cv.best_score_ , tree_cv.best_score_, knn_cv.best_score_]}
model_df = pd.DataFrame(models)

model_df
```

[38]:

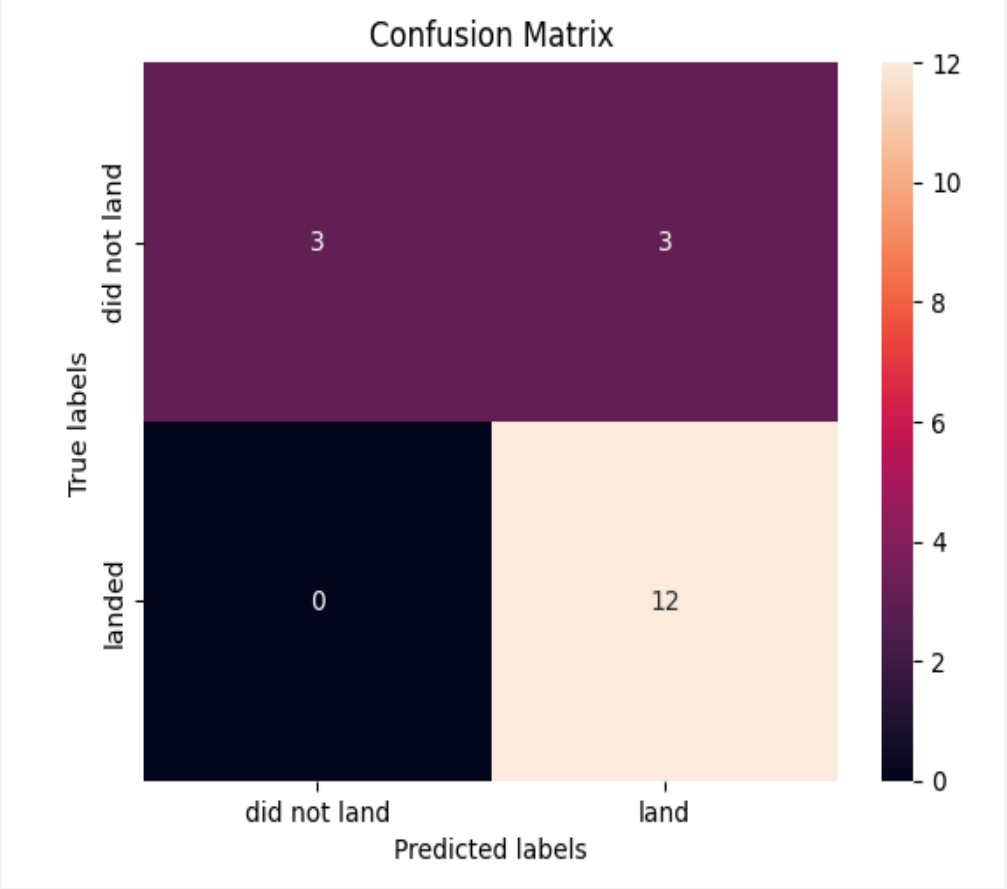
	model	best_score
0	LogisticRegression	0.846429
1	SVM	0.848214
2	DecesionTree	0.885714
3	KNN	0.848214

```
]: # Visualize the built model accuracy for all built classification models, in a bar chart

plt.bar(model_df['model'], model_df['best_score'])
plt.xlabel('model')
plt.ylabel('best_score')
plt.show()
```



Confusion Matrix

- Outcome of the confusion matrix:
  - 12 True positive
  - 3 True negative
  - 3 False positive
  - 0 False negative
- The model obtain 3 False positive (type1 error) which is not good. It indicated that the model predict a successful land but in fact it did not land successfully.
- The graph show the confusion matrix for Decision tree model:



Conclusions

- Flight number, payload mass and orbit type are correlated to the landing outcome.
- The flight number increases, the first stage is more likely to land successfully.
- The higher the payload mass, the higher the success rate.
- ES-L1, GEO, HEO and SSO orbit have higher success rate.
- KSC LC-39A has the most successful launch (41.7%).
- Decision tree model is the best model for the project.

Appendix

- SpaceX REST API URL: https://api.spacexdata.com/v4/launches/past
- Wikipedia URL : https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922
- GitHub URL: https://github.com/chinpei199/IBMDataScience_Falcon9.git

Thank you!