IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sandra Sazdovska, MSc
29.09.2025

Outline

Executive Summary

Methodology used for building the Machine Learning Model in this Project

- Summary of methodologies
  - Data Collection
  - Data Wrangling
  - EDA with Data Visualization and SQL
  - Building and Interactive Map with Folium
  - Building and Interactive Dashboard with Plotly Dash
  - Predictive analysis (Classification)
- Summary of all results
  - Insights drawn from EDA
  - Launch Sites proximities Analysis
  - Plotly Dash Dashboard
  - Classification results

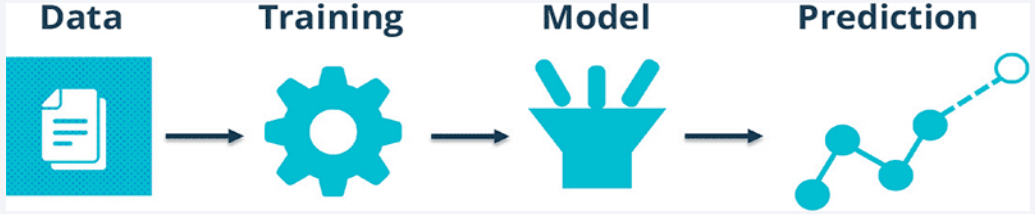
Introduction

Background

- Commercial Space Industry is growing rapidly
- SpaceX leads with reusable first-stage rockets
- Reusability reduces launch costs from **$165M-to-$62M**

My Mission

- Predict Falcon 9 first-stage landing success
- Estimate launch cost using public data
- Train ML models (not rocket science!)
- Build interactive dashboards for Space Y team



Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API and Web Scraping Wiki pages
- Perform data wrangling
  - Filter the data frame to include only Falcon 9 launches, deal with the missing values, perform initial EDA and create the Classification Label
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build predictive models (Logistic regression, Support Vector Machine, Decision Tree and K-nearest Neighbour, tune the parameters for each model, calculate and plot a Confusion Matrix for each, perform Accuracy test on test and training data, and choose the best performing model.

Data Collection

SpaceX REST API

GET request

JSON Data

json.normilize()

Data Frame

Wikipedia Launch Tables

GET request

Parsed HTML tables

BeautifulSoup()

Data Frame

Data cleaning, wrangling
analyzing, processing

...

Data Collection – SpaceX API

Data source SpaceX API URL
https://api.spacexdata.com/v4/launches/past

Fetch the Launch Data and parse it as Pandas Data Frame
GET request method
(fetch the launch data from the API as JSON file)
.json() method
(decode the request response into Python dictionary object)

.json.normalize() method
(convert dictionary object into Panda's Data Frame object)

Use the IDs to access the APIs with helper functions
Dataframe Subset of interest
[flight number, date_utc, rocket, launchpad, payload, cores]

getBoosterVersion(rocket) > list: BoosterVersions
getLaunchSite(LaunchSite) > lists: Longitude, Latitude, LaunchSite
getPayloadSite(Payload) > lists: Payload _mass in kg, Orbit
getCoreSite(cores) > lists: Landing outcome, Number of flights, GridFins used or not, Core reused or not, Legs used or not, Landing pad used or not, Used Block of the core, #times core was used, Serial number of the core

Construct the dataset for analysis

Combine output columns into dictionary

```
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 Panda Data Frame from the dictionary

```
pd.DataFrame.from_dict(launch_dict)
```

Data wrangling

GitHub URL of the completed SpaceX API calls notebook

https://github.com/sazdovska/TestRepo/blob/92c76dd2d3d2033632bded22a0add274d7e60663/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

Data source Wikipedia page
"List of Falcon 9 and Falcon Heavy launches"
https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

Fetch the Launch Data and parse it as BeautifulSoup Object
HTTP GET request method
(fetch the launch data from the Wiki page as HTML)

BeautifulSoup() method
(parse the page HTML into a **BeautifulSoup Object**)

Find all tables and assign the result to a list
BeautifulSoup method
.find_all('tables')

the Launch records = page HTML table List[3]

Get the Column names and use it as keys to create an empty dictionary object
iterate through the **<th>** elements and apply the **extract_column_from_header()**
dict.fromkeys(['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome', 'Version Booster', 'Booster landing', 'Date','Time'])

Create a data frame by parsing the launch HTML tables
Iterate through the <tr> elements > get table row

Is a first table heading a Launch number?

yes

no

Fill up the dictionary with the parsed launch records using helper functions when needed

Create a Panda Data Frame from the dictionary

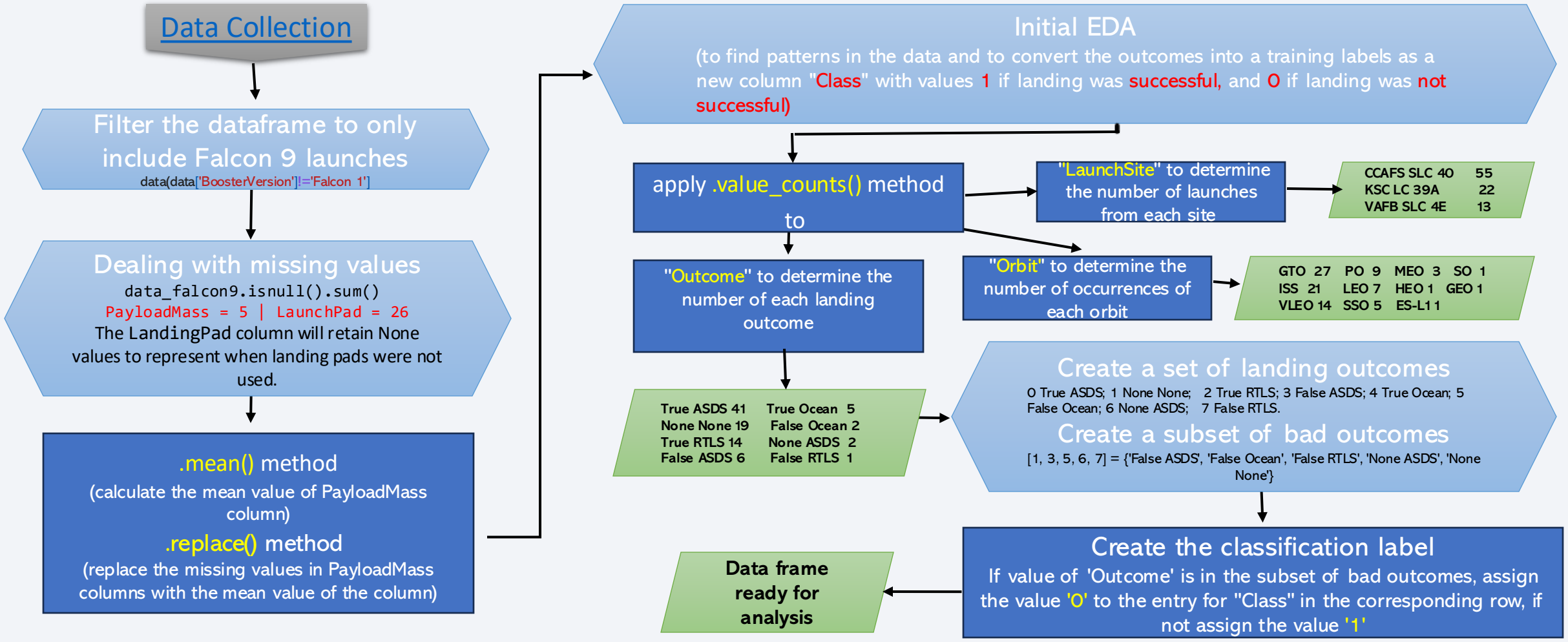
```
pd.DataFrame({ key:pd.Series(value)
for key, value in launch_dict.items() })
```

Data wrangling

GitHub URL of the completed web scraping notebook

https://github.com/sazdovska/TestRepo/blob/3d5727e80ebebe5b22bb36834a19887d9afaa9e5/jupyter-labs-webscraping.ipynb

Data Wrangling



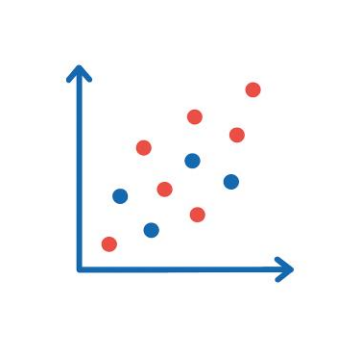
GitHub URL of the completed data wrangling notebook

https://github.com/sazdovska/TestRepo/blob/93937f6dd974d5181eff79a25bed12de237c0806/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

Scatter Charts

… to observe the relationship or correlation between two numerical variables:



Observed variables:

- Flight Number and Launch Site
- Payload Mass and Launch Site
- Flight Number and Orbit Type
- Payload Mass and Orbit Type

Bar Charts

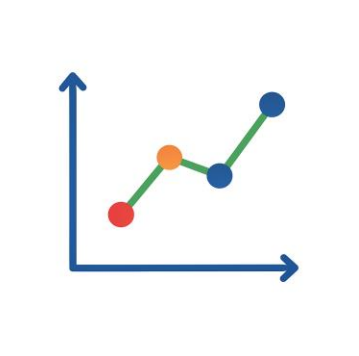
… to observe the relationship or correlation between categorical variables:

Observed variables:

- Success Rate of each Orbit Type

Line Charts

… to observe the change of the variable over time



Observed variables:

- Launch Success Yearly Trend

GitHub URL of your completed EDA with data visualization notebook

https://github.com/sazdovska/TestRepo/blob/d9ee367ecf70b95a3f7045256e348a09a55abf54/edadataviz.ipynb

EDA with SQL

A series of SQL queries were performed on the dataset in order 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 achieved.
- 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 all the booster versions that 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 landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

GitHub URL of your completed EDA with SQL notebook

https://github.com/sazdovska/TestRepo/blob/e9c290c0b17c0bf9eddfa16183848cc5a4926eeb/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

Launch success rates depend on factors such as payload mass, orbit type, and especially the geographic location of launch sites, which influence rocket trajectories.
Interactive maps, built with Python's Folium library, help explore existing sites and reveal patterns for selecting optimal locations.

Folium map objects created:

Interactive Web Map (folium Map object, with an initial center location to be NASA Johnson Space Center at Houston, Texas)
folium.Map()

Mark and Visualize all Launch Sites (as highlighted circles)
folium.Circle()
folium.Marker()

Group the launches to a Launch Site
MarkerCluster()
folium.Icon()

Mark the success/failed launches for each site (Green for success, Red for failure)
folium.Marker()
folium.Icon()

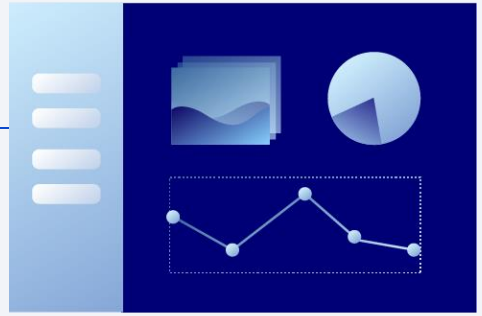
Add Mouse Position (to get the coordinate (Lat, Long) for a mouse over on the map)
MousePosition()

Calculate the distances between a launch site to its proximities (coastline, railway, highway, city etc.)
Folium.Polyline()

GitHub URL of the completed interactive map with Folium map

https://github.com/sazdovska/TestRepo/blob/cf3d03089810e7fed860a8b60eb54eeb932d0533/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash



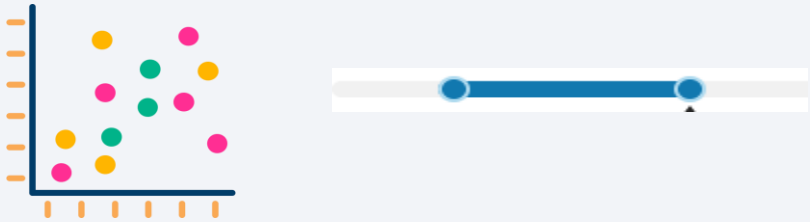
The interactive visual analytics on SpaceX launch data in real-time helps to get a better insight into the data and answer questions about the success rates of each site or the booster version, as well as to explore the effect of payload mass on the success rate, etc.

Success Pie Chart with user input component as a **Dropdown List**

To gain better insight and visually represent:

- the launch site with the highest success rate
- the success rate of each launch site

Success Payload Scatter Chart with user input component as a **Double Range Slider**



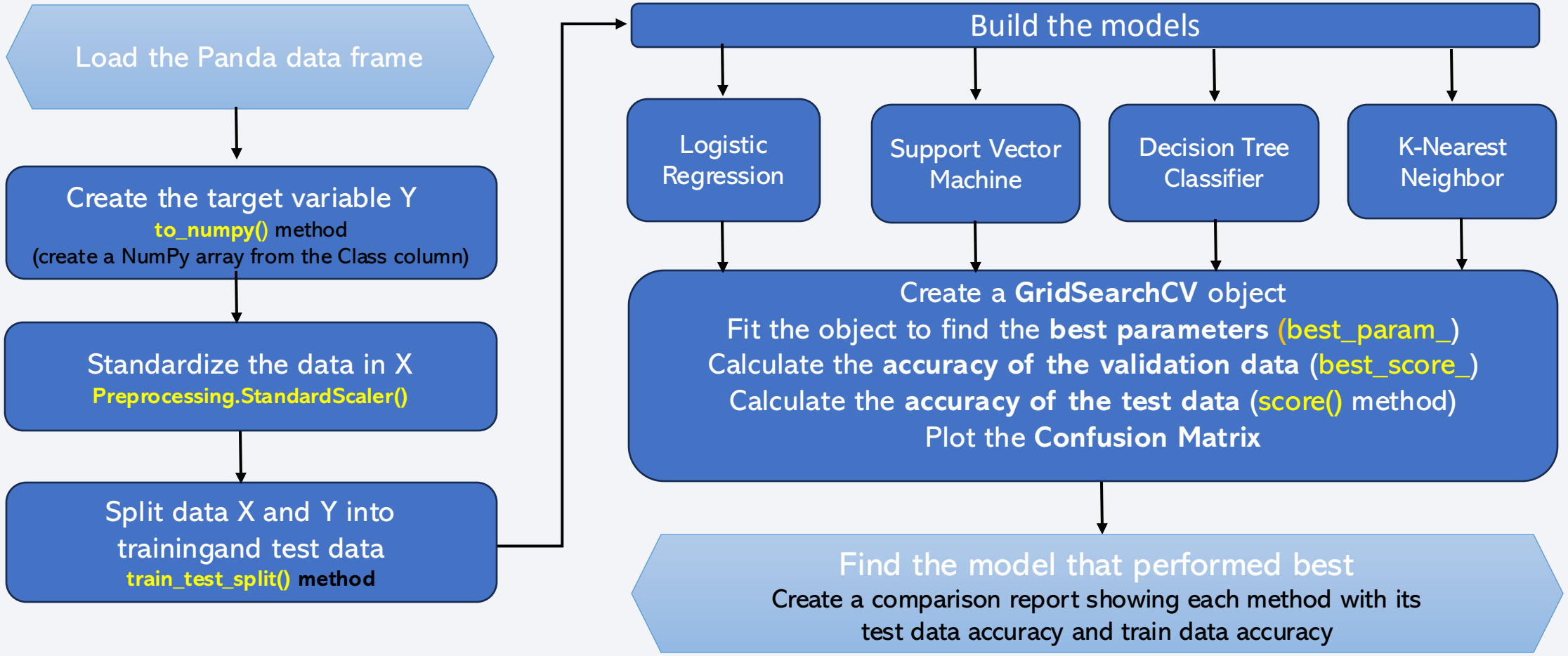
To gain better insight and visually represent:

- the effect of the Payload mass on the success rate of each Launch Site

GitHub URL of your completed Plotly Dash lab

https://github.com/sazdovska/TestRepo/blob/153fa9644585815090b0c96a2c70edfe5322aff4/spacex-dash-app.py

Predictive Analysis (Classification)

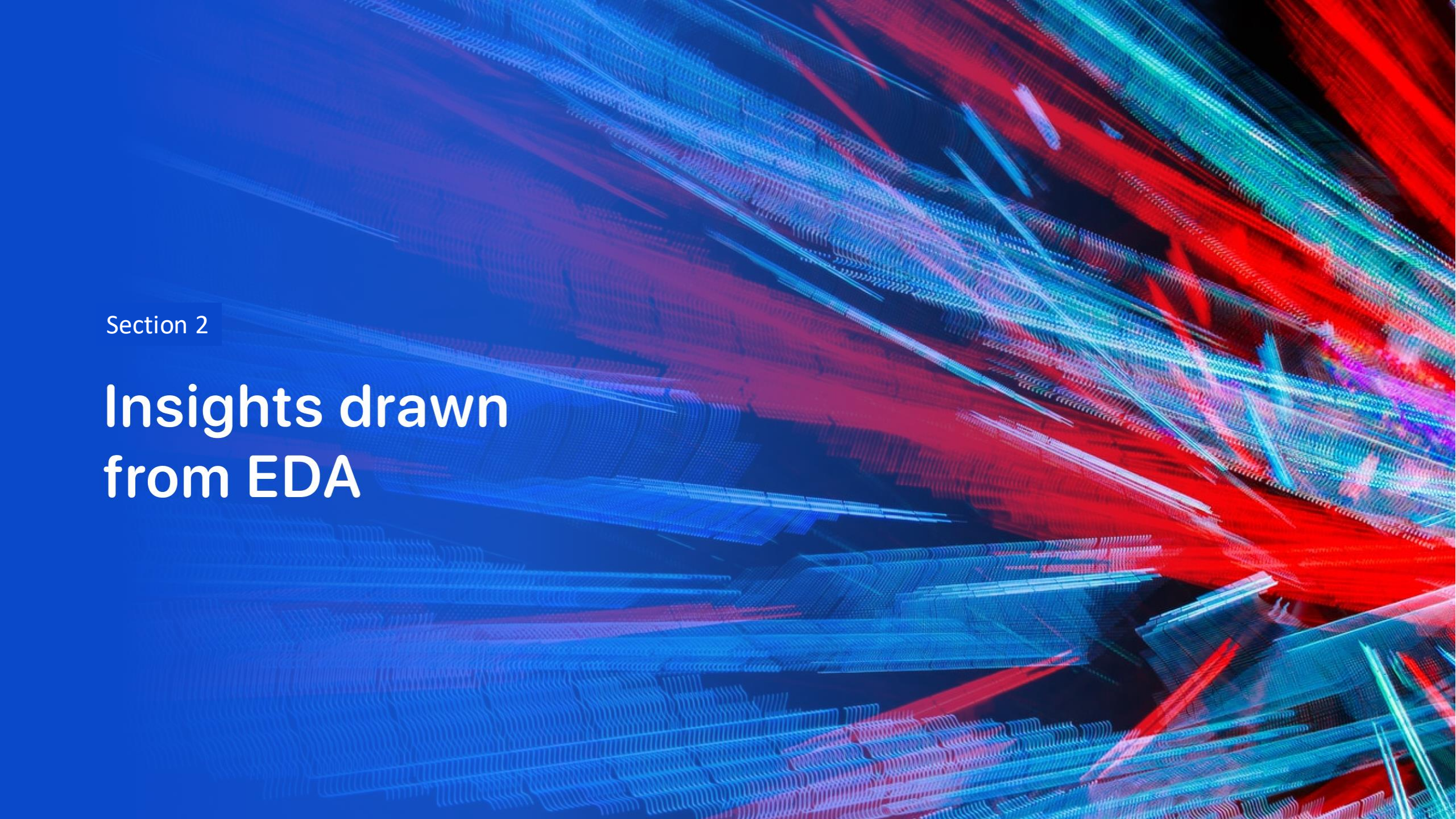


GitHub URL of the completed predictive analysis lab

https://github.com/sazdovska/TestRepo/blob/b3ac701c205e05bfdbcde9a26032596529ec0e1c/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

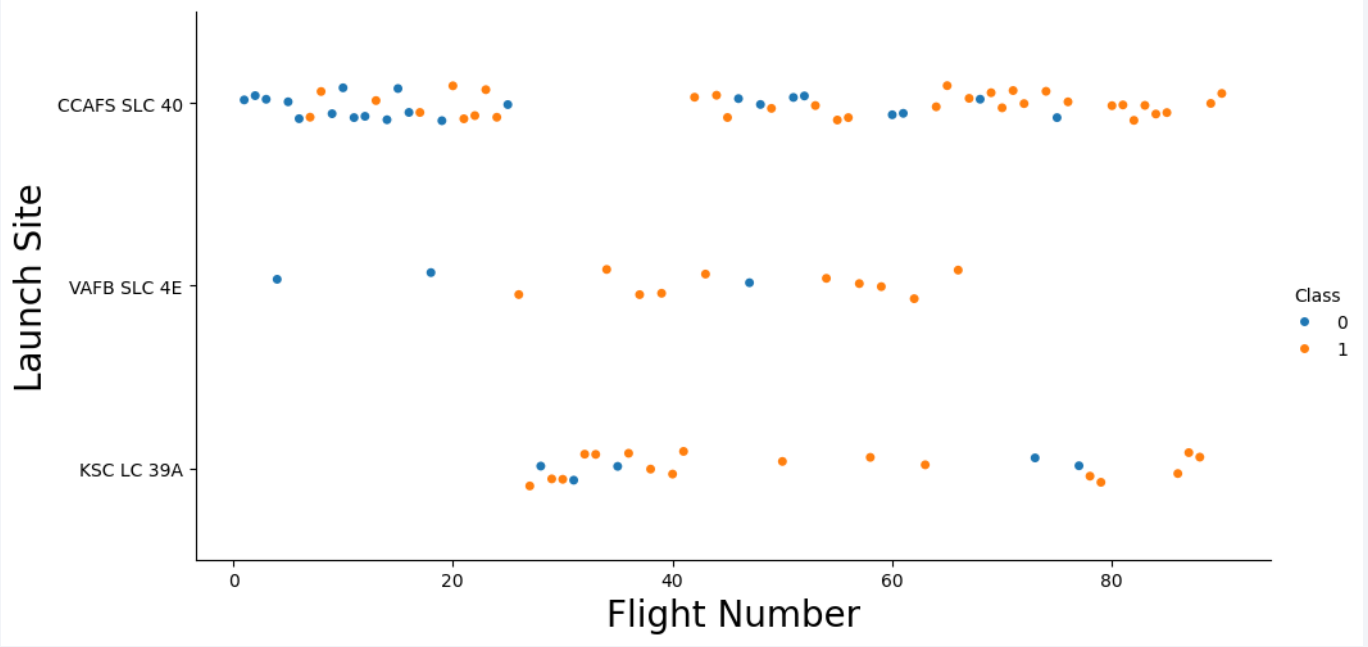
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

The main observation is that there is a positive correlation between the Flight Number and the Lauch Site:

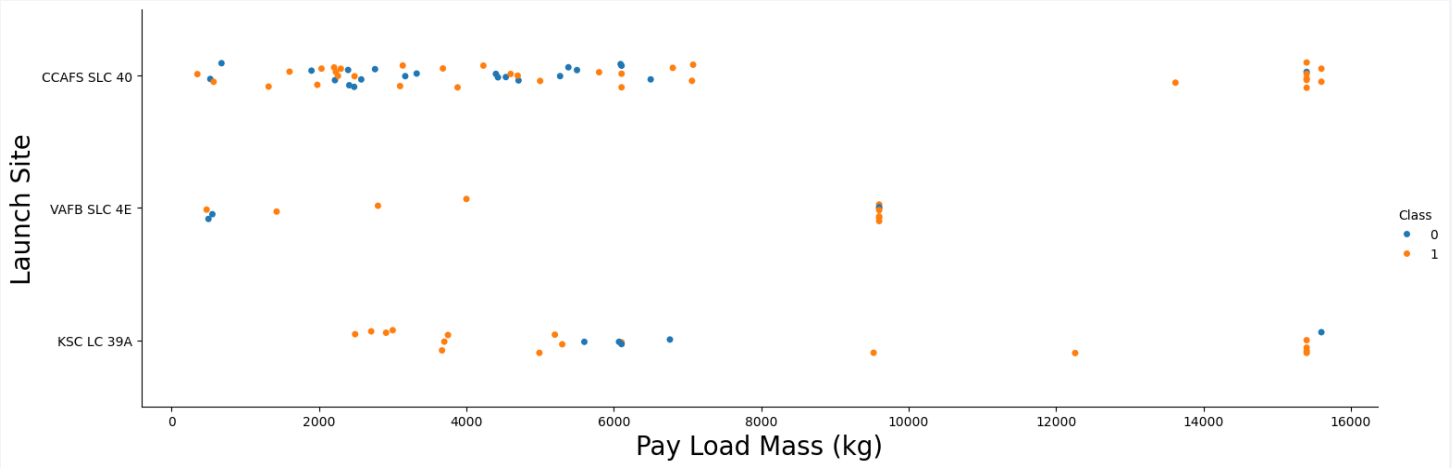
- As the number of flights increases the success rate at each launch site also increases



Payload vs. Launch Site

Main observations are:

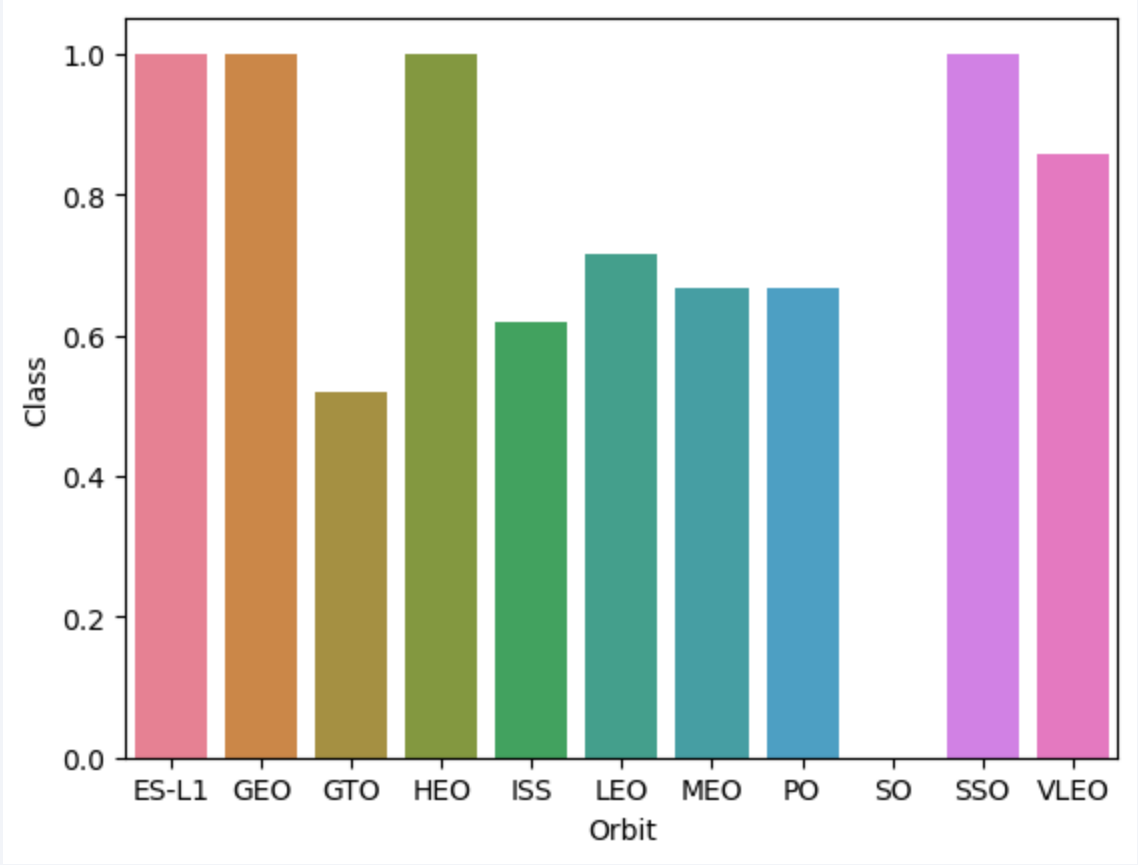
- All Lauch Sites have similar success rate with Pay Load Mass up to 10000kg
- CCAFS SLC 40 and KSC LC 39A have greater success rate for very heavy load mass, whereas
- VAFB SLC 4E has no launches with Pay Load mass greater than 10000kg



Success Rate vs. Orbit Type

Main observations from the Bar Chart representing the correlation of the Success Rate with each Orbit is:

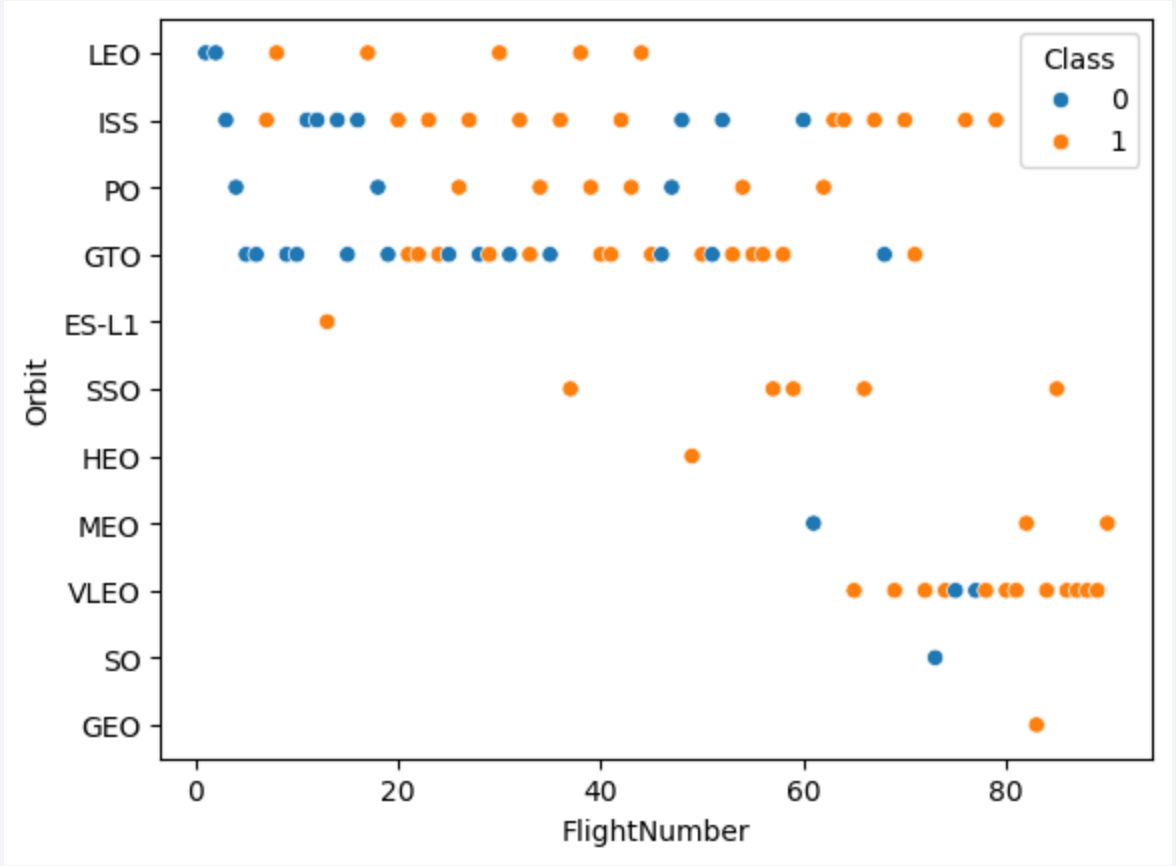
- The Orbits ES-L1, GEO, HEO and SSO have the highest Success Rate of 100%, whereas
- The Orbit SO have the lowest Success rate of 0%.



Flight Number vs. Orbit Type

Main observations are:

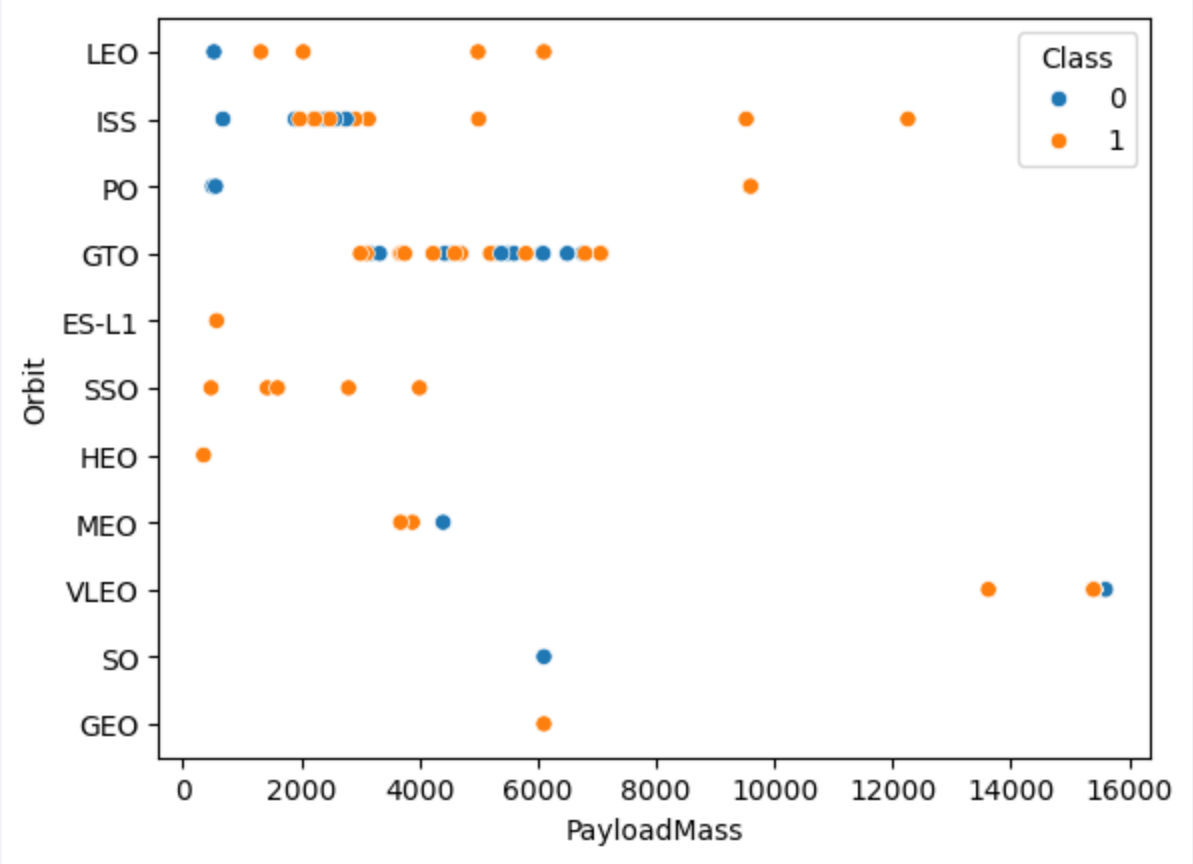
- There is a clearly **evident positive correlation** between the orbit and the number of flights in the case of the orbit **LEO**; whereas
- There is **no evident correlation** between the orbit and the number of flights in the case of the orbit **GTO**



Payload vs. Orbit Type

Main observations are:

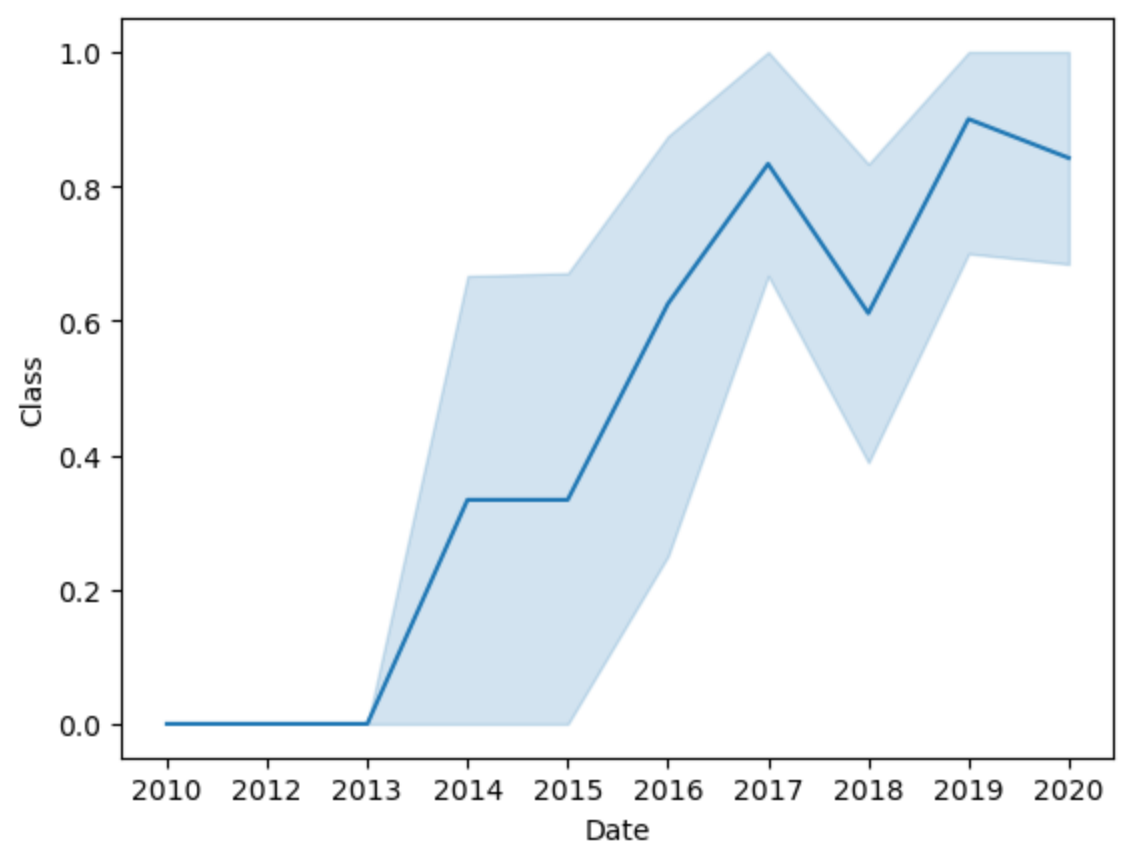
- There is a clear distinction of higher success rate with heavier payload mass for orbits ISS, PO and LEO; whereas
- For orbit GTO is almost impossible to distinguish between successful and unsuccessful landing, as both outcomes are present independently of the payload mass



Launch Success Yearly Trend

Main observation is that from 2013 to 2020 there is a trend of increasing success rate:

- Up to 2013 there were no successful landings as the Success rate is 0;
- From 2013 up to 2020 there is an evident increase in the success rate, with over 50% successful landings from year 2016 up to 2020



All Launch Site Names

SQL Query

```
%sql SELECT DISTINCT(Launch_Site) FROM SPACEXTABLE;
```

Output

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

SQL **SELECT DISTINCT** statement is used to retrieve the unique values from the Launch_Site column by eliminating duplicate records ensuring that only distinct non-repeated values are returned.

Launch Site Names Begin with 'CCA'

SQL Query

```
%sql SELECT * FROM SPACEXTABLE
     WHERE Launch_Site LIKE 'CCA%'
     LIMIT 5;
```

Output

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

SQL **LIKE** statement is used with wildcard 'CCA%' to filter the SPACEXTABLE and retrieve only the records where the Launch Site name begins with the substring CCA.
LIMIT 5 statement returns the first five records of the filtered table

Total Payload Mass

SQL Query

```
%sql SELECT SUM(PAYLOAD_MASS__KG_)
     FROM SPACEXTABLE
     WHERE CUSTOMER = "NASA (CRS)";
```

Output

SUM(PAYLOAD_MASS__KG_)
45596

SQL **SELECT SUM** statement is used to calculate and display the total payload mass carried by boosters launched by NASA (CRS).

Average Payload Mass by F9 v1.1

SQL Query

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

Output

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

SQL **SELECT AVG** statement is used to calculate and display average payload mass carried by booster version F9 v1.1

First Successful Ground Landing Date

SQL Query

```
%sql SELECT MIN(DATE) FROM SPACEXTABLE
    WHERE Landing_Outcome
    LIKE "%ground pad%";

                    Or

%sql SELECT MIN(DATE) FROM SPACEXTABLE
    WHERE Landing_Outcome = 'Success (ground pad)';
```

Output

MIN(DATE)
2015-12-22

SQL **SELECT MIN(DATE)** statement is used to retrieve the earliest date when a successful landing outcome on ground pad was achieved.
WHERE statement is used to filter the records where the outcome was a successful landing on a ground pad.

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query

```
%sql SELECT Booster_version FROM SPACEXTABLE
     WHERE (PAYLOAD_MASS__KG_
            BETWEEN 4000 AND 7000)
     AND
     Landing_Outcome="Success (drone ship)";
```

Output

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

SQL query is used to retrieve the boosters from the SPACEXTABLE that landed successfully on a drone ship and have payload mass between 4000 and 7000 kg.
WHERE statements with **Boolean operator AND** are used to filter the data that fulfills the given criteria.

Total Number of Successful and Failure Mission Outcomes

SQL Query

```
%sql SELECT(SELECT COUNT(MISSION_OUTCOME)
     FROM SPACEXTABLE
          WHERE MISSION_OUTCOME
          LIKE '%Success%')AS Success,
     (SELECT COUNT(MISSION_OUTCOME)
     FROM SPACEXTABLE
          WHERE MISSION_OUTCOME
          LIKE '%FAILURE%') AS Failure;
```

Output

Success	Failure
100	1

The first SQL subquery with **SELECT COUNT** statement is used to calculate and retrieve the total number of successful landings.
The second SQL subquery with **SELECT COUNT** statement is used to calculate and retrieve the total number of failed landings.

Boosters Carried Maximum Payload

SQL Query

```
%sql SELECT DISTINCT Booster_version
     FROM SPACEXTABLE
     WHERE PAYLOAD_MASS__KG_ =
           (SELECT MAX(PAYLOAD_MASS__KG_)
            FROM SPACEXTABLE);
```

Output

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

SQL **SELECT MAX** subquery statement is used to filter the data and retrieve only the records with maximal payload mass.
The main SQL **SELECT DISTINCT** statement is used to retrieve the unique values from the Booster_version column of the filtered data, by eliminating duplicate records ensuring that only distinct non-repeated values are returned.

2015 Launch Records

SQL Query

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE,
      substr(Date, 6, 2) as 'month',
      substr(Date, 0,5) as 'year'
     FROM SPACEXTABLE
     WHERE
      LANDING_OUTCOME = "Failure (drone ship)"
      AND substr(Date,0,5)='2015';
```

Output

Booster_Version	Launch_Site	month	year
F9 v1.1 B1012	CCAFS LC-40	01	2015
F9 v1.1 B1015	CCAFS LC-40	04	2015

SQL **SELECT** statement is used to retrieve the Boosters names and Launch sites **WHERE** the landing was unsuccessful and took place in year 2015.

Substr function is used to process the date and returns the month and the year as follows:

substr(Date, 6,2) returns the month

substr(Date, 0, 5) returns the year

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

SQL Query

```
%sql SELECT LANDING_OUTCOME,
       COUNT(LANDING_OUTCOME) AS
      'LANDING_OUTCOME_COUNT'
     FROM SPACEXTABLE
     WHERE
       DATE BETWEEN '2010-06-04' AND '2017-03-20'
       GROUP BY LANDING_OUTCOME
       ORDER BY LANDING_OUTCOME_COUNT DESC;
```

Output

Landing_Outcome	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

SQL **SELECT** statement is used to calculate and retrieve the count of distinct landings outcomes between 2010-06-04 and 2017-03-20.
WHERE statement is used to filter out the records between the given dates.
GROUP BY statement is used to group the filtered records by landing outcome.
ORDER BY statement is used to order the grouped records in descending order.

Section 3

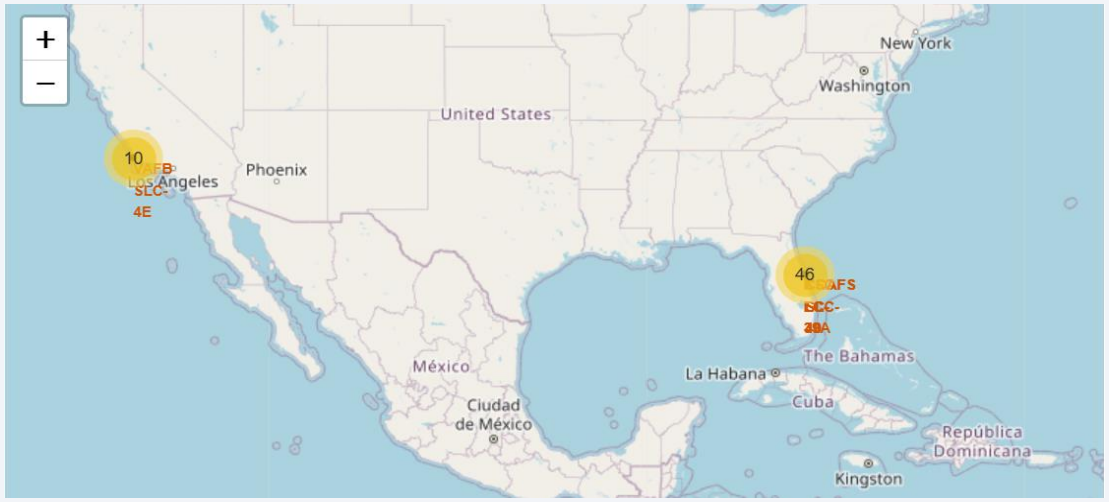
Launch Sites Proximities Analysis

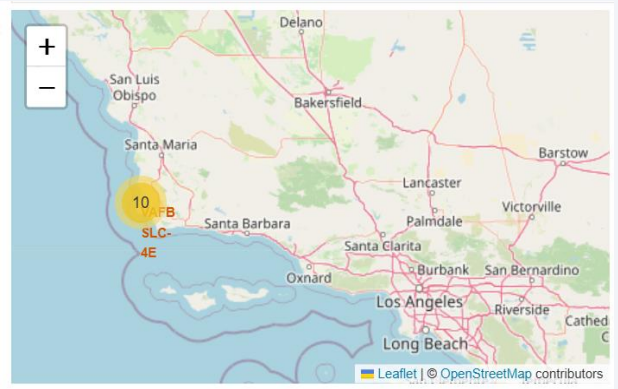
Web Map of the Launch Sites Locations

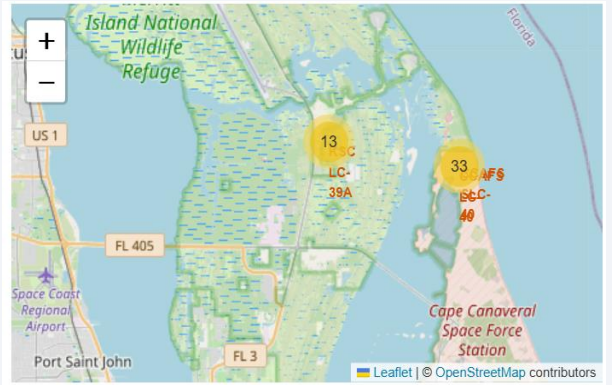
Interactive Web Map showing the location of the Launch sites

All launch sites are located near the USA coastline:

- West Coast (California):
  - VABF SLC-4E
- East Coast (Florida):
  - KSC LC-39A
  - CCAF SLC-40
  - CCAFS SLC-40





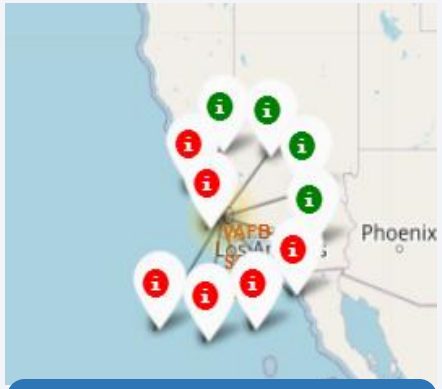


Color labeled Launch Outcomes for each Launch Site

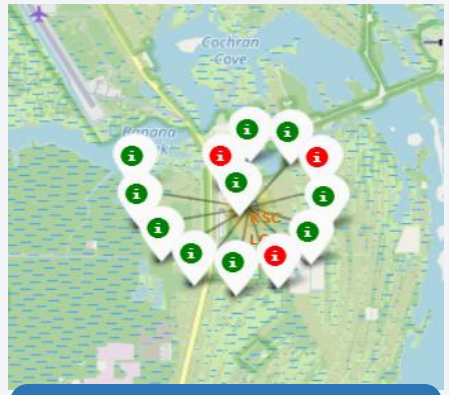
Launches are grouped into clusters by Launch Site. The launch outcomes are represent as follows:

GREEN markers = SUCCESS
RED markers = FAILURE

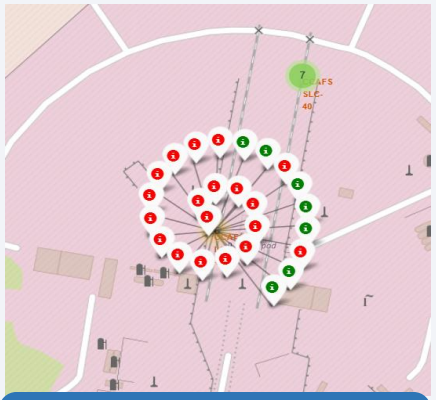
Observation:
KSC SLC-39A Launch Site has the **Greatest Success Rate.**



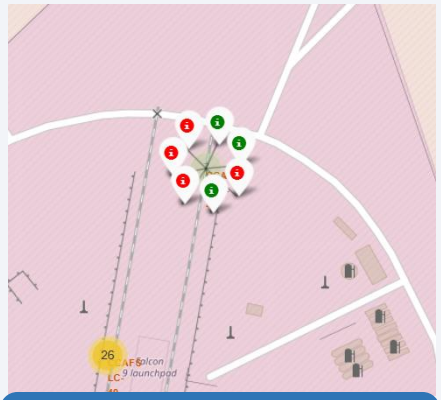
VAFB SLC-4E



KSC SLC-39A



CCAF SLC-40

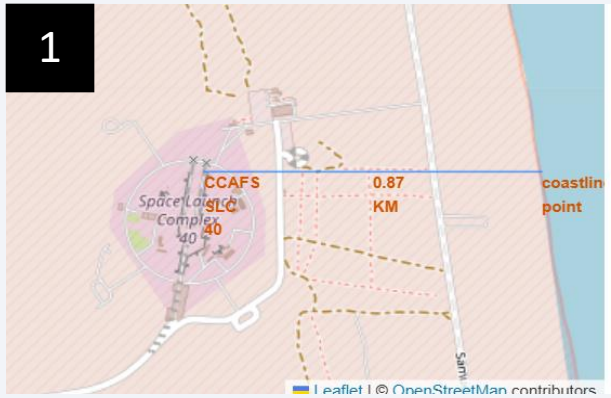


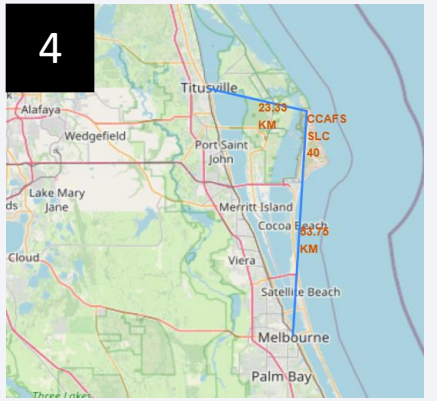
CCAFS SLC-40

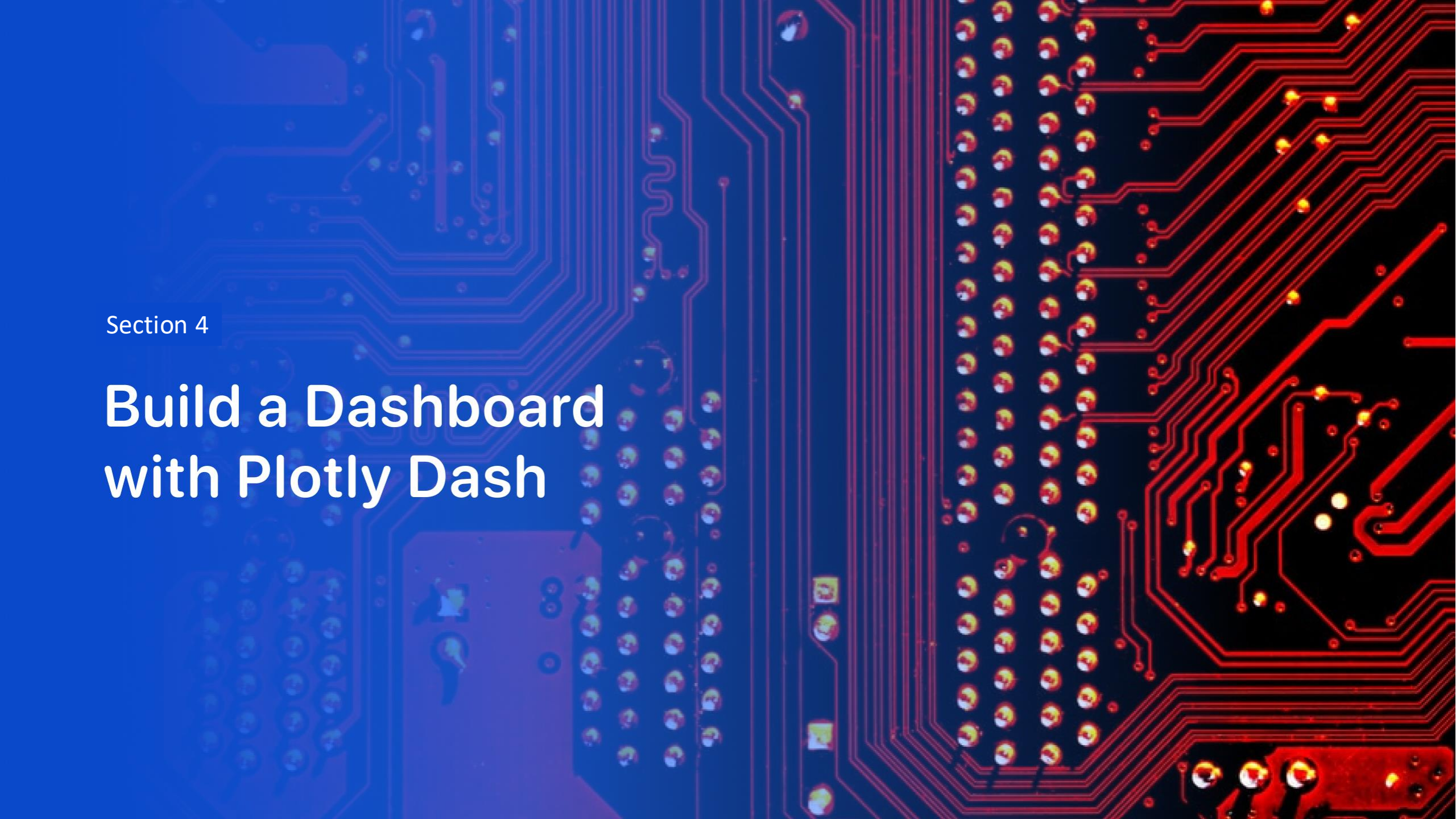
CCAFS SCL-40 Proximity Analysis

Observations about the CCAFS SCL-40 Launch Site proximity to:

1. Coast Line – CCAFS SLC-40 is only 870m (0.87km) from the coastline;
2. Highways – CCAFS SLC-40 is only 590m (0,59km) from the Samuel S Philips Parkway;
3. Railroads – The nearest active railroad, Florida East Coast Railway, is approximately 22km from CCSAF SLC-40;
4. Cities – The two bigger cities in CCAFS SLC-40 proximity, Titusville and Melbourne, are approximately 23.33km and 53.75km away.



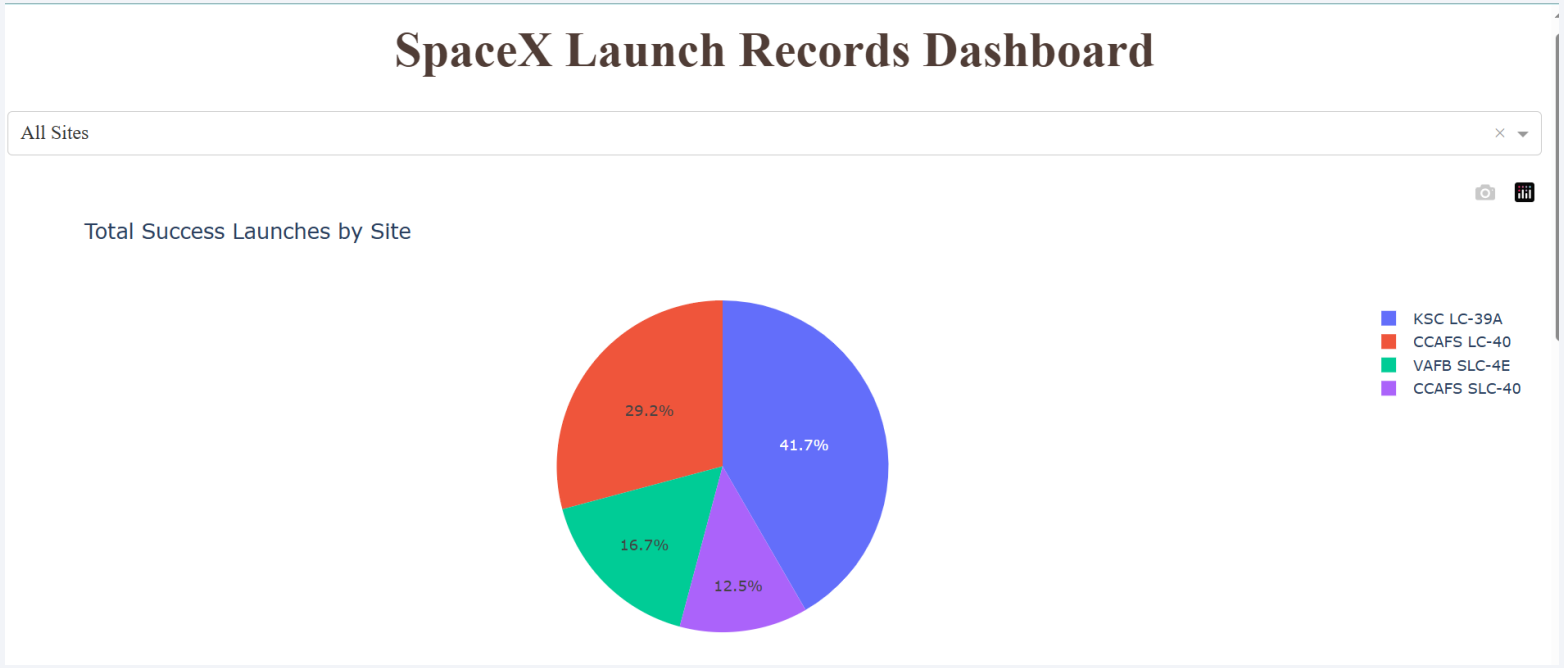



Section 4
Build a Dashboard with Plotly Dash

Pie Chart of Total Success Rates by Launch Site

Observation:

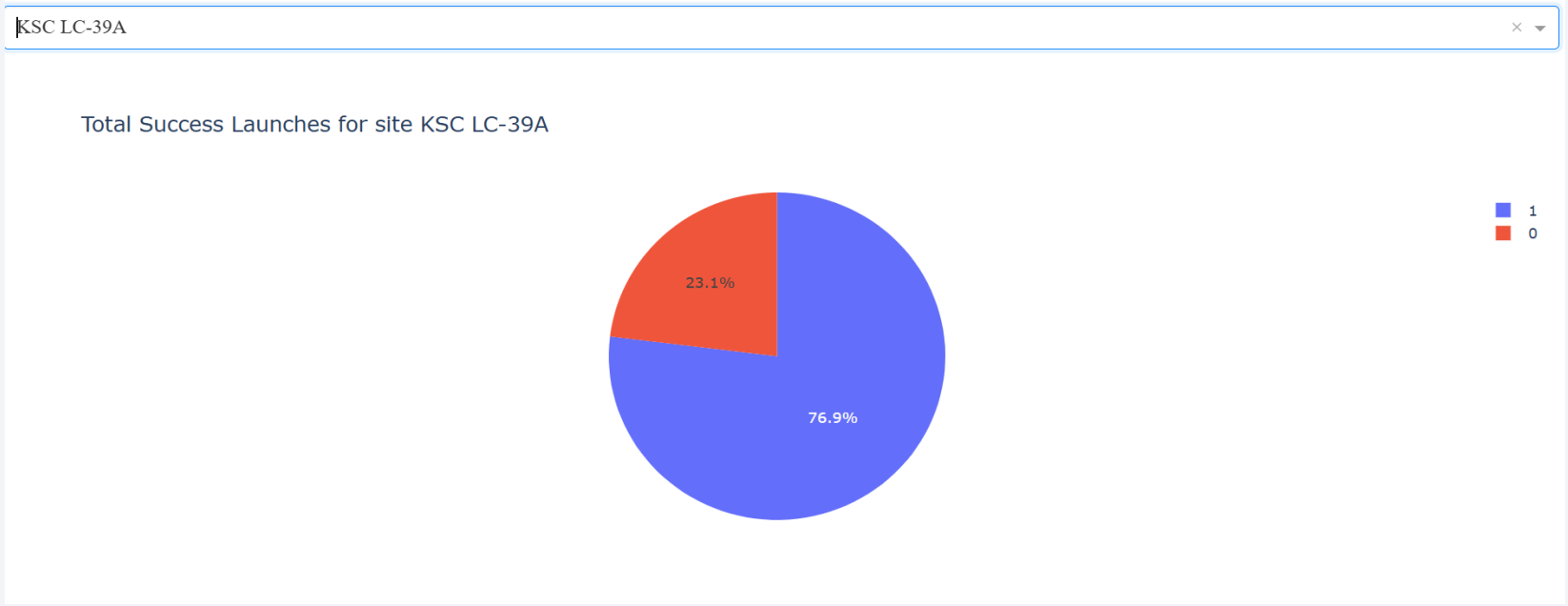
KSC LC-39A with 41.7% successful launches is the launch site with the best success rate



Pie Chart of success/failure rate of KSC LC-39A

Observation:

KSC LC-39A, as the launch site with highest success rate, has successful launching outcome in 76.9% of the times, and failure rate of only 23.1%.

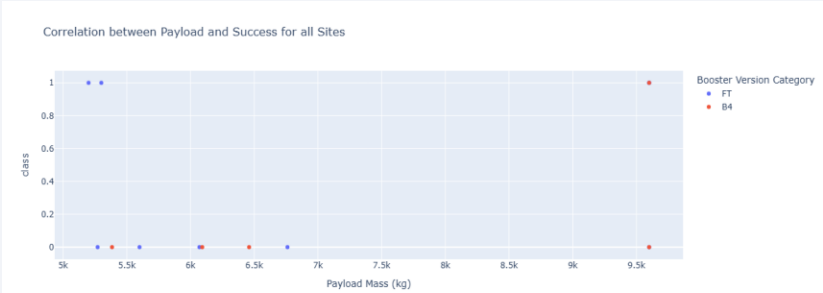


Scatter Plot of Payload vs. Launch Outcome for all sites with different payload mass

Observation:

- Lower payload mass has higher success launch rate;
- Payload mass from 2000-4000kg has highest success rate;
- FT booster version has highest success rate with heaver payload mass
- FT booster version has overall highest success rate with both, lower and heavier payload mass.

Payload mass from 0-4000kg



Payload mass from 5000-10000kg

Section 5

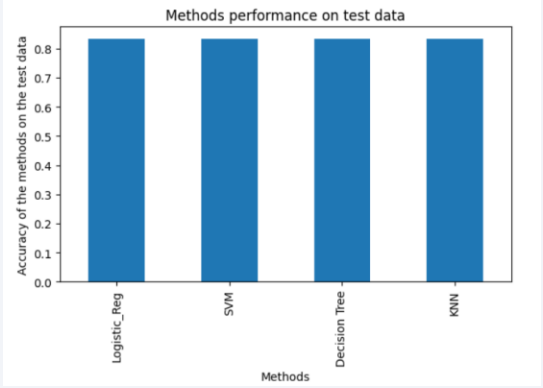
Predictive Analysis (Classification)

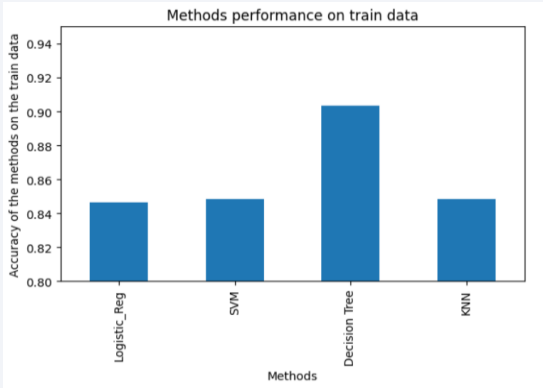
Classification Accuracy

Observations:

- Accuracy test shows that all models performed well
- According Accuracy test on train data the Decision Tree model performed slightly better than other models
- Decision Tree would be the model of choice, unless its higher accuracy is not due to some data leakage

	Test Data Accuracy	Train Data Accuracy
Logistic_Reg	0.833333	0.846429
SVM	0.833333	0.848214
Decision Tree	0.833333	0.903571
KNN	0.833333	0.848214

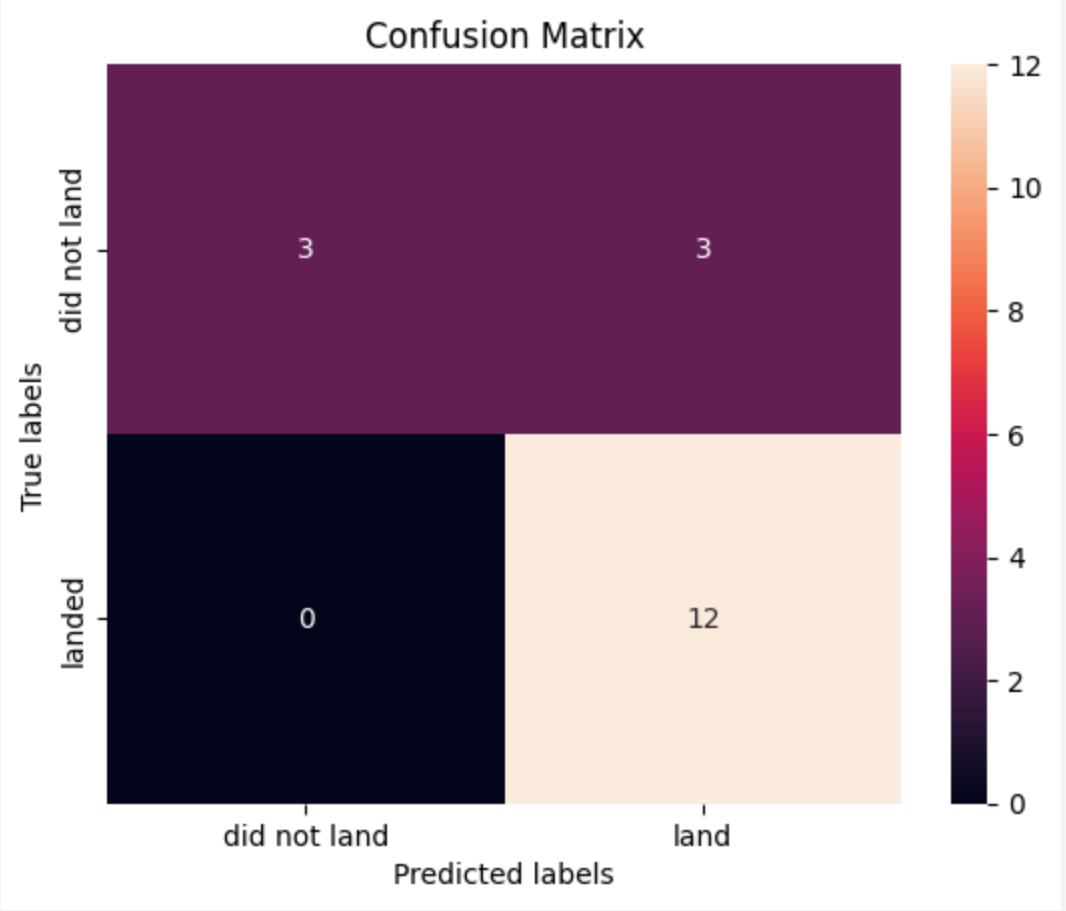




Confusion Matrix

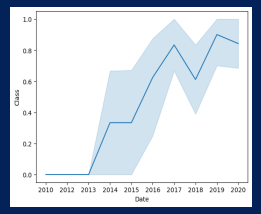
Observations:

- Accuracy test shows that all models performed well
- According the Accuracy test on train data the Decision Tree model performed slightly better than other models
- Decision Tree would be the model of choice, unless its higher accuracy is not due to some data leakage



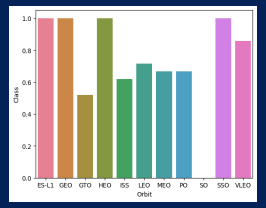
Conclusions

Steady Improvement



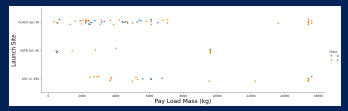
SpaceX landing success rate transitioned dramatically from 0% before 2013 to steadily increasing success by 2020, surpassing 50% after 2016.

Orbit type matters



Certain orbits (ES-L1, GEO,HEO, SSO) reached 100% success, while others like SO had none.

Payload mass effect

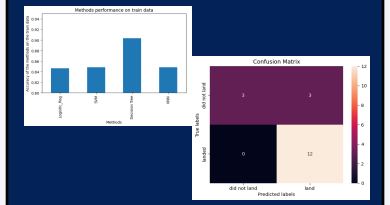


Lighter payloads (2,000–4,000 kg) show the best results; heavy payloads succeed most often at KSCLC-39A and CCAFSSLC-40.The FT booster version has the overall highest success rate..

Top Launch Site

KSC LC-39A stands out with highest success 41.7% successful launches among all sites, and successful launching outcome in 76.9% of the times.

ML Model Performance



All ML models performed well, with Decision Tree showing slightly better accuracy—though it should be validated for potential data leakage.

Thank you!