Winning Space Race with Data Science

<Soh Yan Le Mervin>
<October 7th 2021>

Outline

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

Executive Summary

Summary of methodologies

- Data Collection from “SpaceX REST API” and Web scraping of historical records.
- Data Wrangling for labels to be used in training supervised models
- Exploratory Data Analysis (EDA) and Data Feature Engineering (DFE)
- Interactive Visual Analytics and Dashboard (Folium and Plotly Dash)
- Machine Learning Predictive Analysis

Summary of all results

- Process of Data collection, wrangling, EDA and DFE results
- Output of Interactive Visual Analytics, dashboard and Machine learning with SVM, Classification Trees and Logistic Regression

Introduction

Project background and context

- SPACE X, a commercial space flight company provides relatively inexpensive rocket launches (USD 62 Million) as compared to other providers (USD 165 Million). This is because SPACE X can reuse the first stage development of rockets where most of the work and costs are incurred. During the first stage development, rockets may be reused or sacrificed due to mission parameters. The goal is to train machine learning model and use available information provided to predict if SpaceX will reuse the first stage and the determine the cost of each launch.

Problems you want to find answers

- Factors and conditions which impacts the launch success rates
- Machine learning pipeline to predict launch outcomes

Section 1

Methodology

Methodology (Executive Summary)

Data Collection Methodology

- Representational state transfer (REST) API and Web scraping by parsing HTML data into python dictionary.

Data Wrangling

- Using Pandas dataframe to create labels and manipulate data for training supervised models based on Data collected.

Exploratory Data Analysis

- Querying IBM Db2 SQL data base to summarize information on landing outcomes and rocket parts
- Visualize relationships of data with seaborn and matplotlib

Interactive Visual Analytics

- Analyze launch success factors such as payload, orbit type, launch site, positionings with Folium
- Build dashboard with Plotly on detailed launch records

Predictive Analysis using classification models

- Find the best hyper parameter and use Scikit-Learn for machine learning with Support Vector Machine, Classification Trees, Logistic Regression and K Nearest Neighbor

Data Collection

EXTRACT	TRANSFORM	LOAD
1. Import Libraries and Define Auxiliary Functions	3. Convert, Normalize structures, or parse data into data frames created.	5. Export data from data frames created into files such as CSV for reusability and further steps
2. Request data from APIs using GET Request or Web Pages using HTML GET Method	4. Filter and manipulate data based on requirements	

Data Collection – SpaceX API

Github Link

1. Import Libraries
2. Use GET requests to specific columns in Space X API and define functions to append the data into lists.

```
import requests
# Pandas is a softwar
import pandas as pd
# NumPy is a library
arge collection of hi
import numpy as np
# Datetime is a libra
import datetime
```

```
# Takes the dataset and uses the rocket column to call the API and append the data to the
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

3. Establish Get Requests In Variable “Response”
4. Decode the response content and turn it into a Pandas dataframe.

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

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

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

5. Apply Functions to append data into lists
6. Assign Lists to dictionary

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

```
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}
```

7. Create Pandas dataframe from the dictionary launch_dict
8. Filter Values and manage null values.

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

```
data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']
```

```
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, PL_mean)
```

9. Export data from data frames created into files such as CSV

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

Data Collection – Scraping

Github Link

1. Import Libraries
2. Define Functions - web scraping process on HTML Tables

```
!pip3 install beautifulsoup4
!pip3 install requests
```

```
import sys
import requests
from bs4 import BeautifulSoup
import re
import unicodedata
import pandas as pd
```

```
def landing_status(table_cells):
    """
def date_time(table_cells):
    """
def booster_version(table_cells):
    """
```

3. HTTP GET method to request HTML Page
4. Create a BeautifulSoup object from the HTML response

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

```
soup = BeautifulSoup(data,'html5lib')
```

Find all column names from HTML Table
Create Dictionary based on column names extracted

```
# Assign the result to a list called
html_tables = soup.find_all('table')
```

```
column_names = []
```

```
print(column_names)
```

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each
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']=[]
```

7. Parsing Process to extract data required into dictionary

```
extracted_row = 0
#Extract each table
for table_number,table in enumer
    # get table row
    for rows in table.find_all("
        #check to see if first t
        if rows.th:
            if rows.th.string:
                flight_number=ro
                flag=flight_numb
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save ce
        if flag:
            extracted_row += 1
```

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

Data Wrangling

Github Link

1. Import Libraries
2. Load Data SpaceX Data Set

```
import pandas as pd
import numpy as np
```

```
df=pd.read_csv("https://
v")
```

3. Compute the following requirements
4. Create a landing outcome label (“Class”) from Outcome column

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

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

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

```
for i,outcome in enumerate(Landing_outcomes.keys()):
    print(i,outcome)
```

```
bad_outcomes=set(Landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
landing_class = []

for i in df['Outcome']:
    if i in set(bad_outcomes):
        landing_class.append(0)
    else:
        landing_class.append(1)
```

5. Calculate required values with “Class” Column
6. Export computed data

```
df["Class"].mean()
```

```
df.to_csv("dataset_part\_2.csv", index=False)
```

EDA with Data Visualization

GitHub Link

Scatter Plots:

Used to show relationship between two variables:

- Flight Number vs Launch Site
- Payload vs Launch Site
- Payload vs Orbit Type
- Flight Number vs Orbit Type
- Payload vs Orbit Type

Line Chart

Used to show data across a period of time and the increasing or decreasing trend it sets:

- Space X Launch Success Rate since 2013

Bar Chart

Used to compare data between the different Orbit Types:

- Success Rate of Each Orbit Type

Build an Interactive Map with Folium

Github Link

Summary Syntax	Map Objects Created
folium.Circle site_map.add_child(circle)	Add a highlighted circle area
folium.map.Marker site_map.add_child(marker)	Text label
MarkerCluster() def assign_marker_color(launch_outcome) site_map.add_child(marker_cluster)	Create MarkerCluster object and add marker cluster based on assigned market color to map
site_map.add_child(mouse_position)	MousePosition on the map to get coordinate for a mouse over a point on the map
folium.PolyLine(locations=[launch_loc, coast_loc], color='green').add_to(site_map)	Draw a line between the marker to the launch site

EDA with SQL

Github Link

Action	Syntax
load the SQL extension and establish a connection with the database	%load_ext sql
Connect to SQL IBM DB2 database	%sql ibm_db_sa://fkv43691:s8uOyTqmMLHxptw1@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb?security=SSL
Task 1: Display the names of the unique launch sites in the space mission	%sql SELECT DISTINCT(launch_site) FROM SPACEXDATAIBM
Task 2: Display 5 records where launch sites begin with the string 'CCA'	%sql SELECT * FROM SPACEXDATAIBM WHERE launch_site LIKE 'CCA%' LIMIT 5
Task 3: Display the total payload mass carried by boosters launched by NASA (CRS)	%sql SELECT SUM(payload_mass_kg) FROM SPACEXDATAIBM WHERE customer = 'NASA (CRS)'
Task 4: Display average payload mass carried by booster version F9 v1.1	%sql SELECT AVG(payload_mass_kg) FROM SPACEXDATAIBM WHERE booster_version = 'F9 v1.1'
Task 5: List the date when the first successful landing outcome in ground pad was acheived.	%sql SELECT MIN(date) FROM SPACEXDATAIBM WHERE landing_outcome = 'Success (ground pad)'
Task 6: List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000	%sql SELECT booster_version FROM SPACEXDATAIBM WHERE landing_outcome = 'Success (drone ship)' AND payload_mass_kg_ BETWEEN 4000 and 6000;
Task 7: List the total number of successful and failure mission outcomes	%sql SELECT mission_outcome, COUNT(*) FROM SPACEXDATAIBM GROUP BY mission_outcome
Task 8: List the names of the booster_versions which have carried the maximum payload mass. Use a subquery	%sql SELECT booster_version FROM spacexdataibm WHERE payload_mass_kg = (SELECT MAX(payload_mass_kg) FROM spacexdataibm)
Task 9: List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015	%sql SELECT landing_outcome, booster_version, launch_site FROM spacexdataibm WHERE landing_outcome = 'Failure (drone ship)' AND YEAR(DATE) = 2015
Task 10: 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(landing_outcome) FROM spacexdataibm WHERE date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY landing_outcome ORDER BY COUNT(landing_outcome) DESC

Build an Interactive Map with Folium

Github Link

Summary Syntax	Map Objects Created
folium.Circle site_map.add_child(circle)	Add a highlighted circle area
folium.map.Marker site_map.add_child(marker)	Text label
MarkerCluster() def assign_marker_color(launch_outcome) site_map.add_child(marker_cluster)	Create MarkerCluster object and add marker cluster based on assigned market color to map
site_map.add_child(mouse_position)	MousePosition on the map to get coordinate for a mouse over a point on the map
folium.PolyLine(locations=[launch_loc, coast_loc], color='green').add_to(site_map)	Draw a line between the marker to the launch site

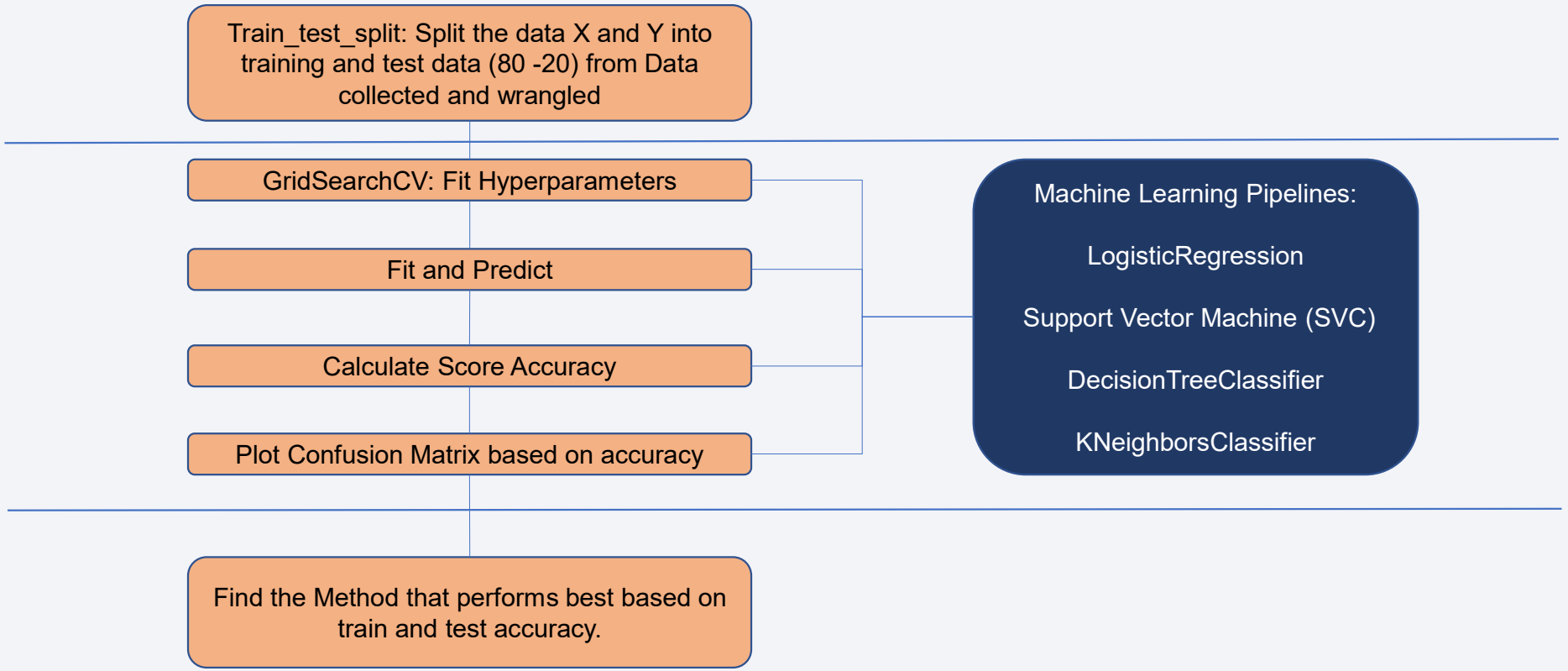
Build a Dashboard with Plotly Dash

Github Link

Plots/graphs and interactions created	Rationale
Dropdown list	Enable Launch Site selection for ALL sites (searchable=True, value='ALL')
Pie Chart	To show total successful launches count for all sites
Slider	select payload range (Min 0 to Max 10000, Step 1000)
Scatter plot	To show the correlation between payload and launch success
Call back Functions	For Pie Chart: Site-dropdown as input and Success-pie-chart as output For Scatter Plot: Site-dropdown and payload-slider as inputs and success-payload-scatter-chart as output
Define functions for Pie Chart and Scatter Plot	Interactivity based on user input

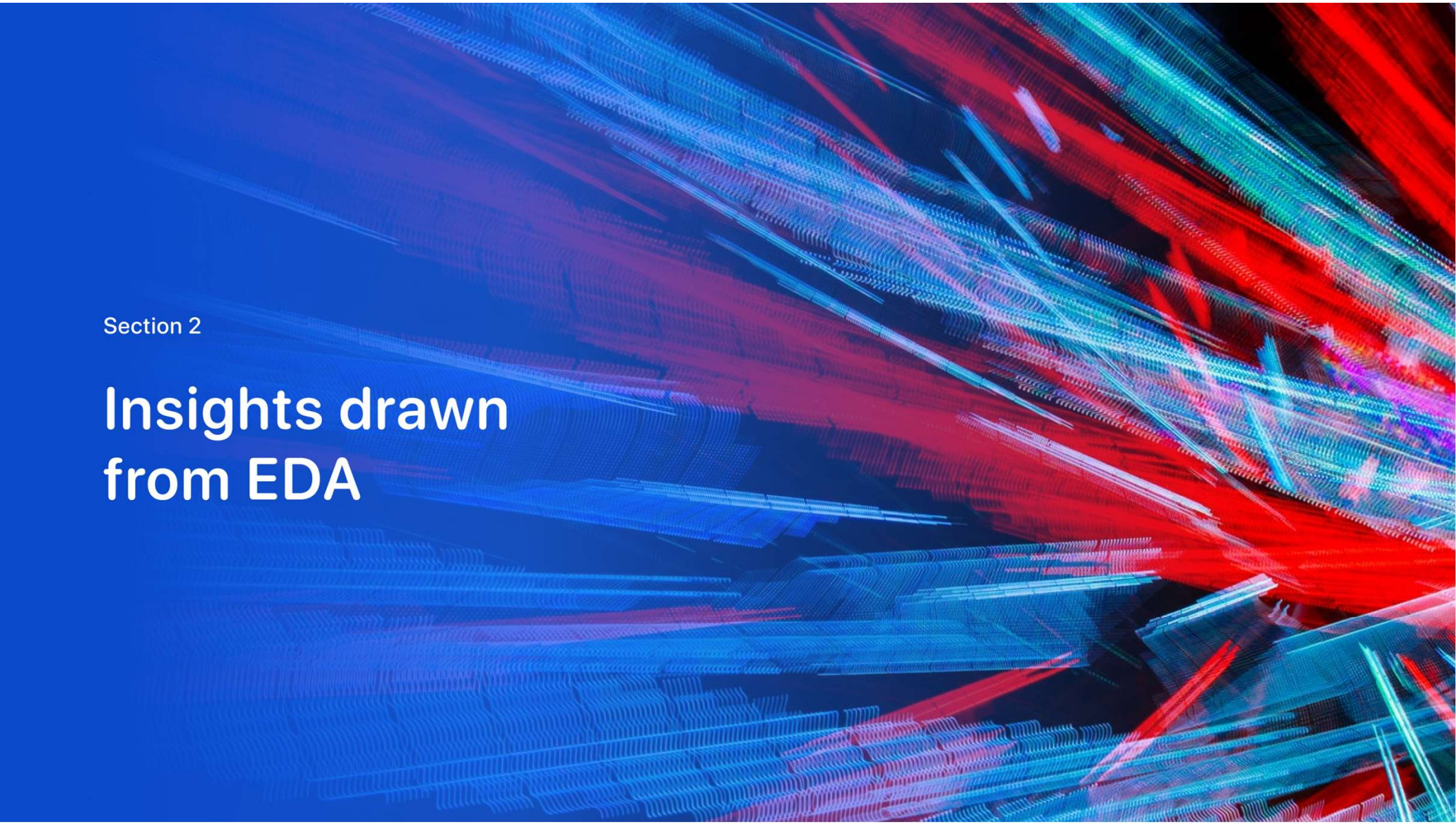
Predictive Analysis (Classification)

Github Link



Results

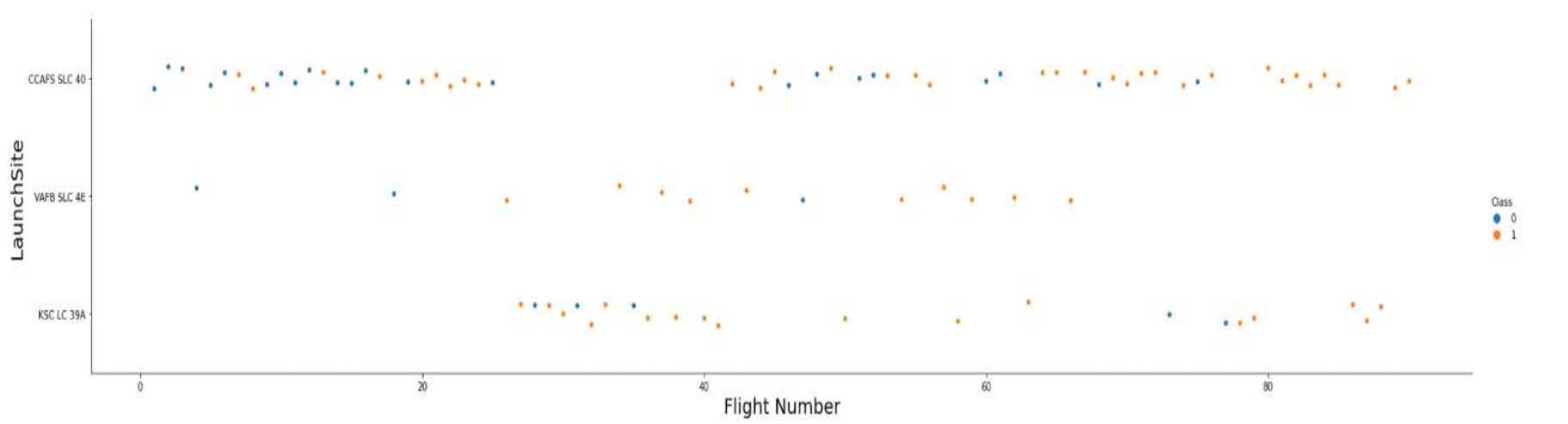
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

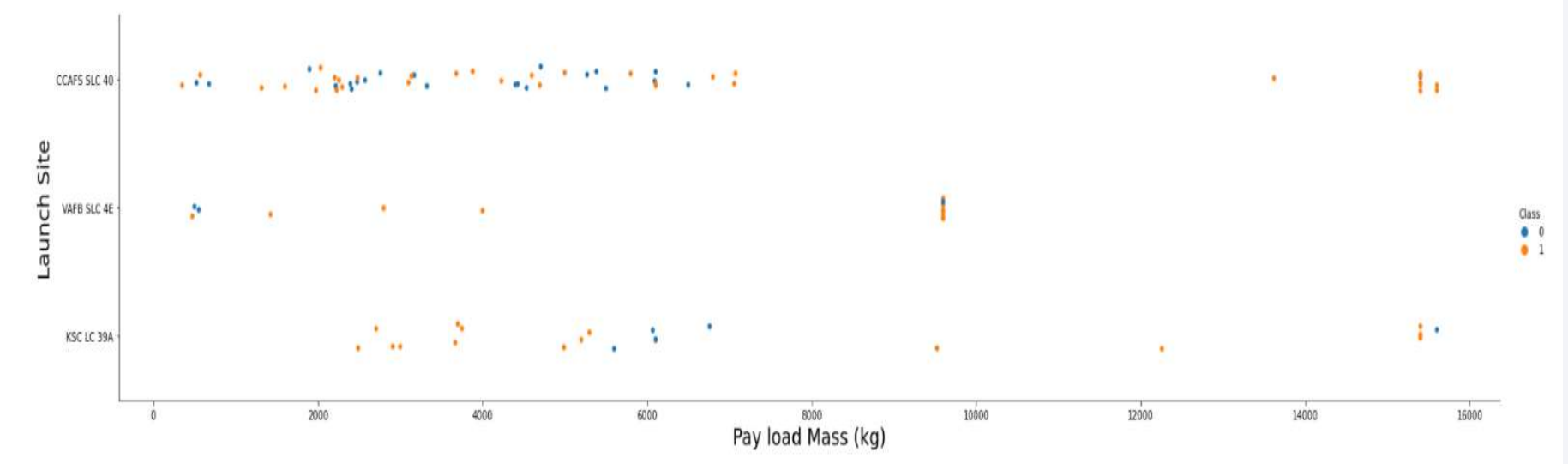
Insights drawn from EDA

Flight Number vs. Launch Site



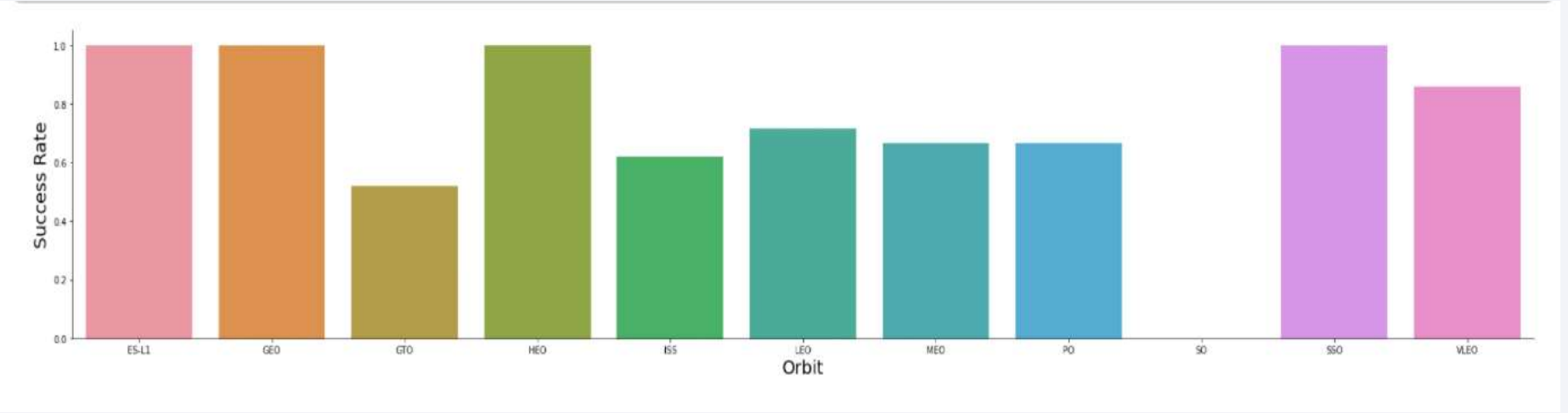
The scatter plot represents unsuccessful landing of class 0 (blue) and successful landing of class 1 (orange).
Both VAFB SLC 4E and CCAFS SLC 40 indicate that there are higher amount of successful landing after flight number 20.
KSC LC 39A successful landing seems to be less affected by later flight numbers as compared to the former.

Payload vs. Launch Site



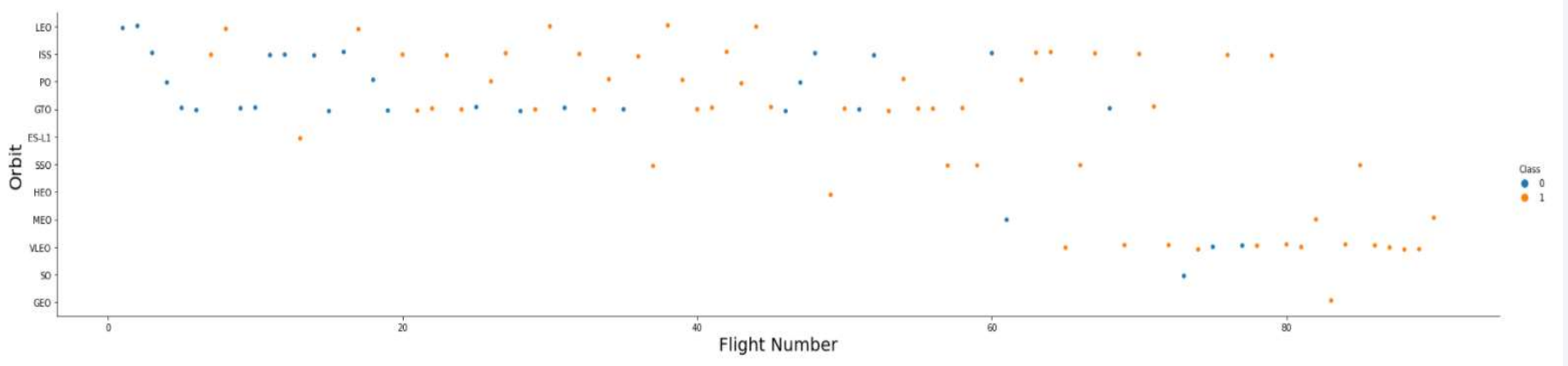
The scatter plot represents unsuccessful landing of class 0 (blue) and successful landing of class 1 (orange). As Payload mass increases, the are lesser launches. A higher success rate is evident in CCAFS SLC 40 as pay load launch increased dramatically from about 13000 kg onwards.
KSC LC 39 A and VAFB SLC 4E indicate full success rate between around 1800kg to 5000 kg of payload.

Success Rate vs. Orbit Type



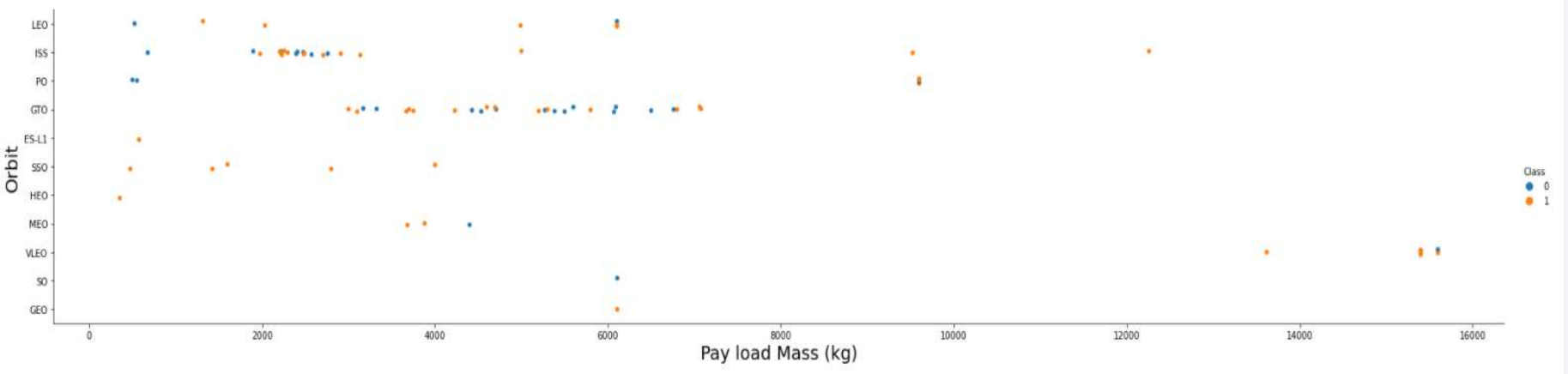
High success rate for Orbit ES-L1, GEO, HEO, SSO and VLEO. SO has no success at all while the rest of the orbits indicate low success rate.

Flight Number vs. Orbit Type



ES-L1,SSO, GEO, HEO indicate full success rate in its flights against Orbit Type.
LEO, ISS and PO seems to have concentrated success rate between flight number 20 to 40 while GTO and VLEO exhibits higher success rate as flight number increases. SO Seems to have 0 success rate,

Payload vs. Orbit Type

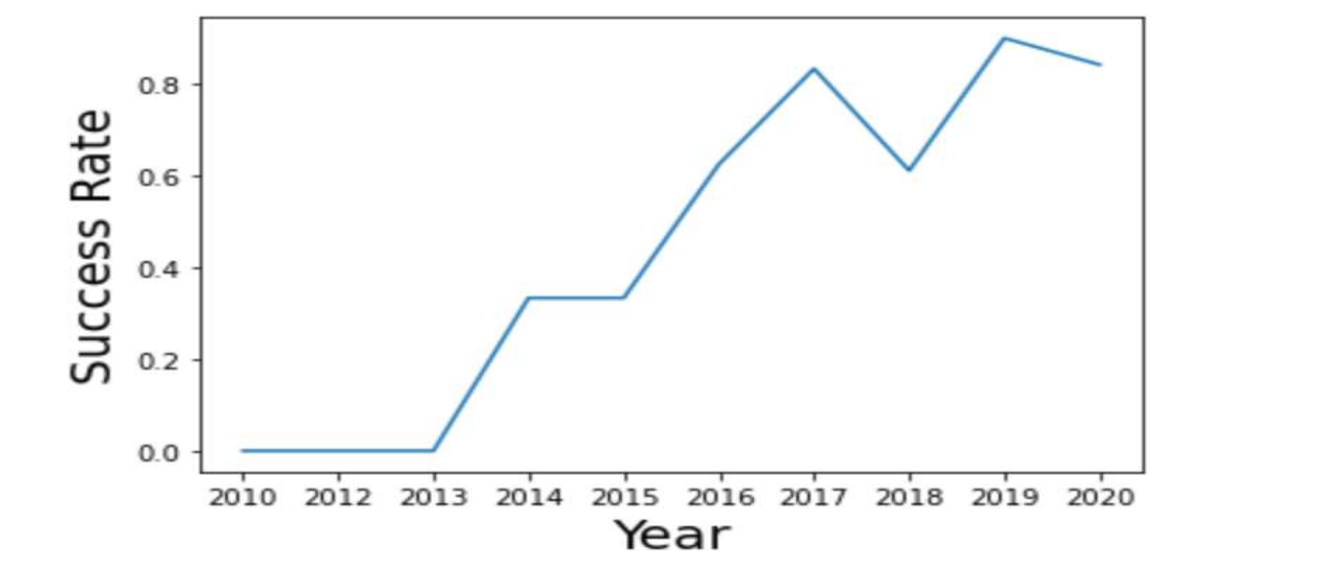


The heavier the payload, the lower the success rate for GTO NEO and VLEO. PO seems unaffected by Payload Mass. Analysis on 100% successful orbits not meaning for payload vs orbit type scatter plot.

Launch Success Yearly Trend

Launch success yearly trend:

Success rate constant increase until 2017 and reach its peak around 2019



All Launch Site Names

Display the names of the unique launch sites in the space mission

```
%sql SELECT DISTINCT(launch_site) FROM SPACEXDATAIBM
```

```
 * ibm_db_sa://fkv43691:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

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

There four launch sites as shown above, in the data set SQL table SPACEXDATAIBM. Note the SQL Clause Distinct is used to return unique values of launch_site.

Launch Site Names Begin with 'CCA'

```
%sql SELECT * FROM SPACEXDATAIBM WHERE launch_site LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://fkv43691:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

Note the “%” sign after the LIKE clause which represents names beginning with CCA. Only 5 rows are returned due to the LIMIT 5 clause.

Total Payload Mass

```
%sql SELECT SUM(payload_mass__kg_) FROM SPACEXDATAIBM WHERE customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://fkv43691:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

1
45596

Notice that no Group By Clause is required as “SUM” function returns scalar value for payload_mass_kg only, of NASA (CRS) customers.

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(payload_mass__kg_) FROM SPACEXDATAIBM WHERE booster_version = 'F9 v1.1'
```

```
 * ibm_db_sa://fkv43691:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

1
2928

For booster_version F9 v1.1, the average payload mass is 2928 kg. The AVG function returns average of column, payload mass kg.

First Successful Ground Landing Date

```
%sql SELECT MIN(date) FROM SPACEXDATAIBM WHERE landing__outcome = 'Success (ground pad)'
```

```
 * ibm_db_sa://fkv43691:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

1
2015-12-22

The first successful landing is on 22nd Dec 2015. The Min function searches the date column for the lowest value after applying the WHERE filter on landing_outcome = ‘Success (ground pad)’.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT booster_version FROM SPACEXDATAIBM WHERE landing__outcome = 'Success (drone ship)' AND payload_mass__kg_ BETWEEN 4000 and 600
0;
```

```
 * ibm_db_sa://fkv43691:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

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

SQL Clause “BETWEEN” used to indicate range 4000 to 6000. “AND” clause combines WHERE filter into the query.

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT mission_outcome, COUNT(*) FROM SPACEXDATAIBM GROUP BY mission_outcome
```

```
 * ibm_db_sa://fkv43691:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

COUNT Function used to count the grouping sets enabled in the GROUP BY clause of mission_outcome. There are 100 successful mission outcome and 1 failure.

Boosters Carried Maximum Payload

```
%sql SELECT booster_version FROM spacexdataibm WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM spacexdataibm)
```

```
 * ibm_db_sa://fkv43691:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

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

MAX aggregate function applied to Payload_mass_kg to return scalar value “X”. SQL query applies filter with WHERE clause returning booster_version of payload_mass_kg = “X”,

2015 Launch Records

```
%sql SELECT landing__outcome, booster_version, launch_site FROM spacexdataibm WHERE landing__outcome = 'Failure (drone ship)' AND YEAR(DATE) = 2015
```

```
 * ibm_db_sa://fkv43691:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

There were only two failed drone ship landings in 2015 with landing outcome = ‘Failure (drop ship)’. The YEAR function is successfully applied to DATE column data type “Date”,

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

```
%sql SELECT landing__outcome, COUNT(landing__outcome) FROM spacexdataibm WHERE date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY landin
g__outcome ORDER BY COUNT(landing__outcome) DESC
```

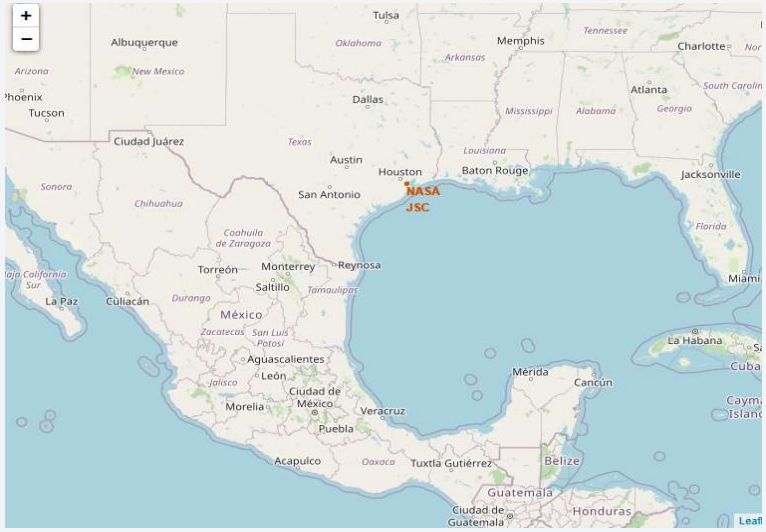
```
 * ibm_db_sa://fkv43691:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

landing__outcome	2
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Landing outcome is in descending order from no attempt with the highest count and precluded (drone ship with lowest count).

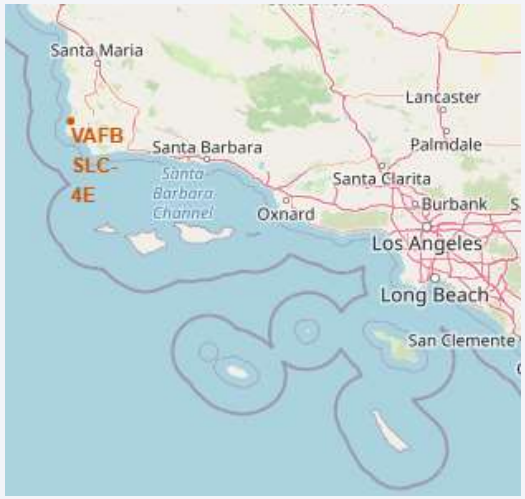

Section 4
Launch Sites
Proximities Analysis

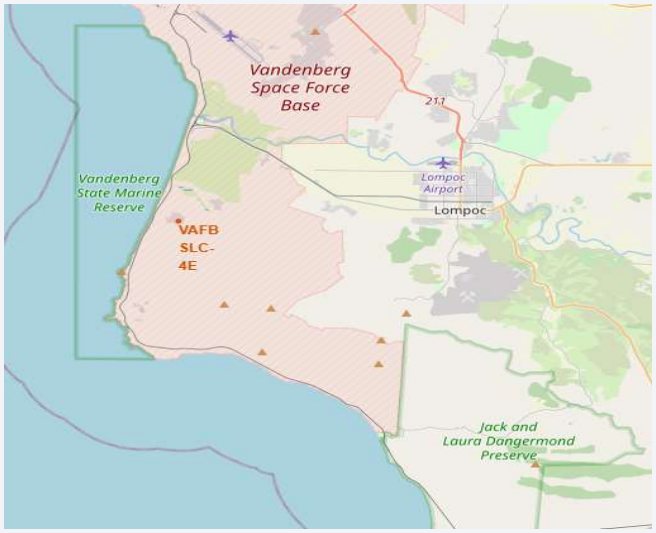
Folium Map Launch Sites 1 of 4: NASA JSC



NASA JSC is located in Houston at Lyndon B.Johnson space center.

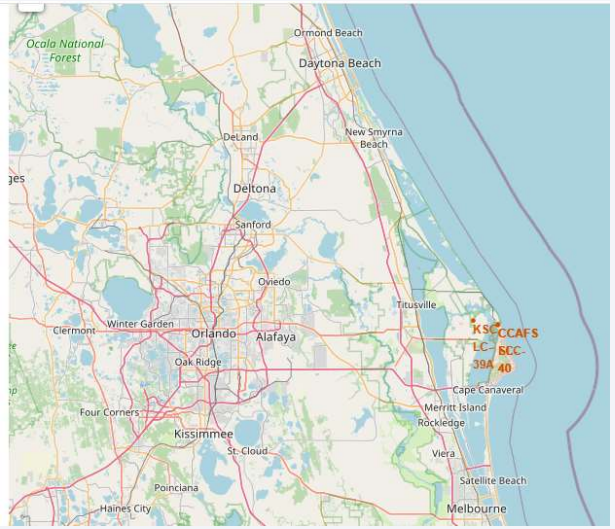
Folium Map Launch Sites 2 of 4 VAFB SLC – 4E :





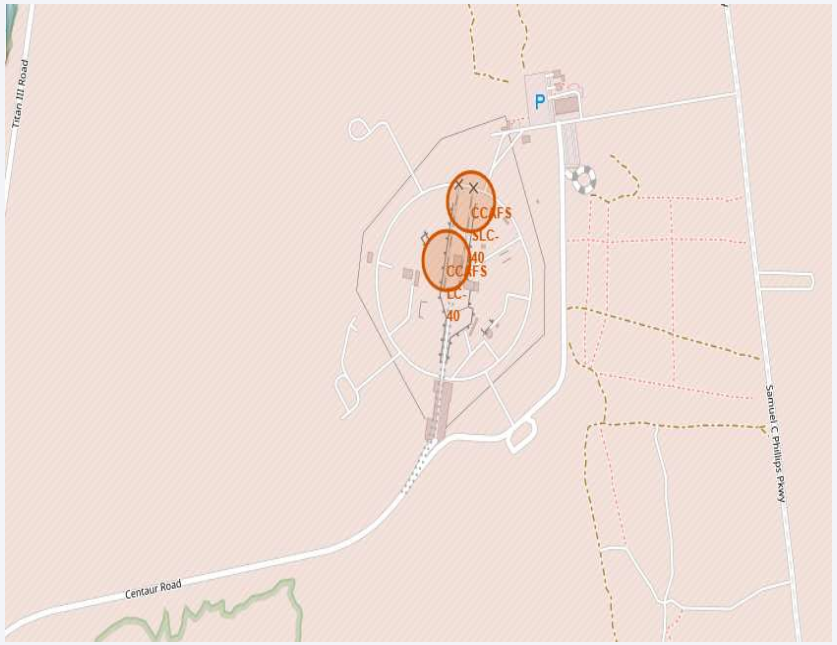
VAFB SLC – 4E is located around South of Santa Maria and West of Santa Barbara. The site is near to Vandenberg State Marine Reserve.

Folium Map Launch Sites 3 of 4 KSC LC 39A:



KSC LC 39 A is located on Merritt Island and new Cochran cove and Banana Creek.

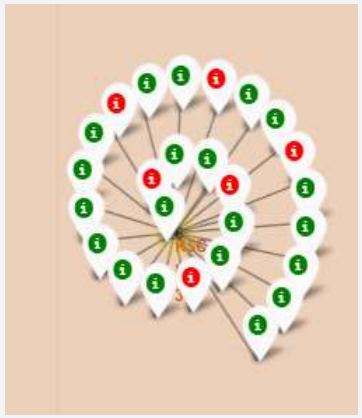
Folium Map Launch Sites 4 of 4 KSC LC 39A:



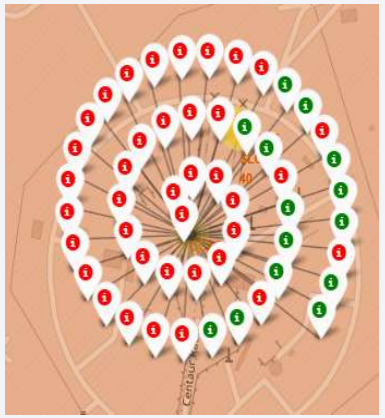
CCAFS SLC – 40 is located on Cape Canaveral, along Centaur Road.

Folium: Landing Outcomes

KSC LC 39 A

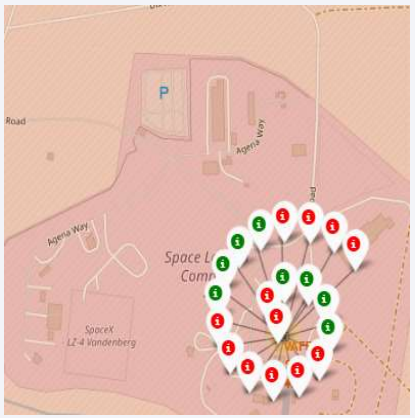


CCAFS LC 40



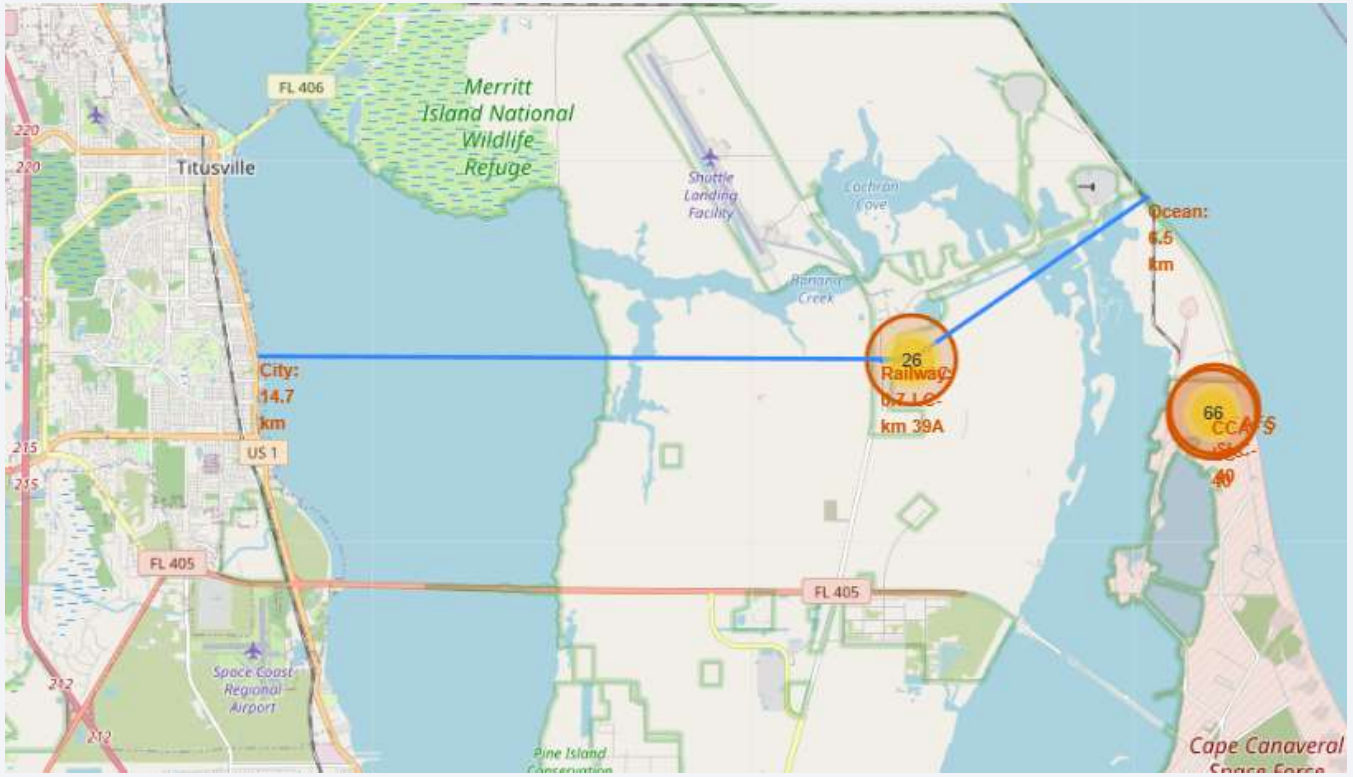
CCAFS SLC 40

CCAFS LC 40

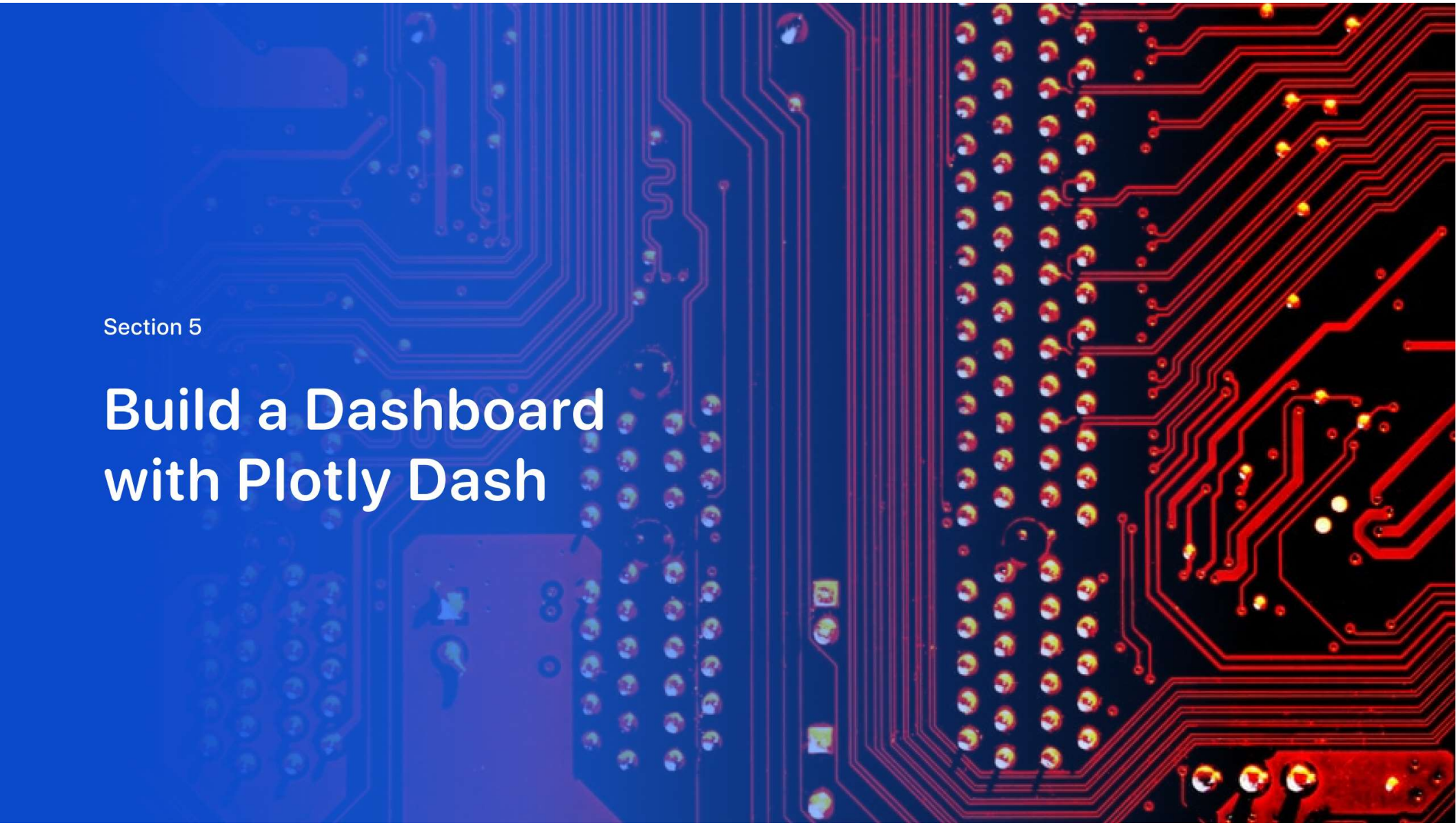


Green represents successful landing while Red represents unsuccessful landing.
CCAFS LC 40 has the most number of attempts, but did not result in high success rate of booster recovery.
KSC LC 39A has highest booster recovery rate.

Proximities of KSC Launch Site



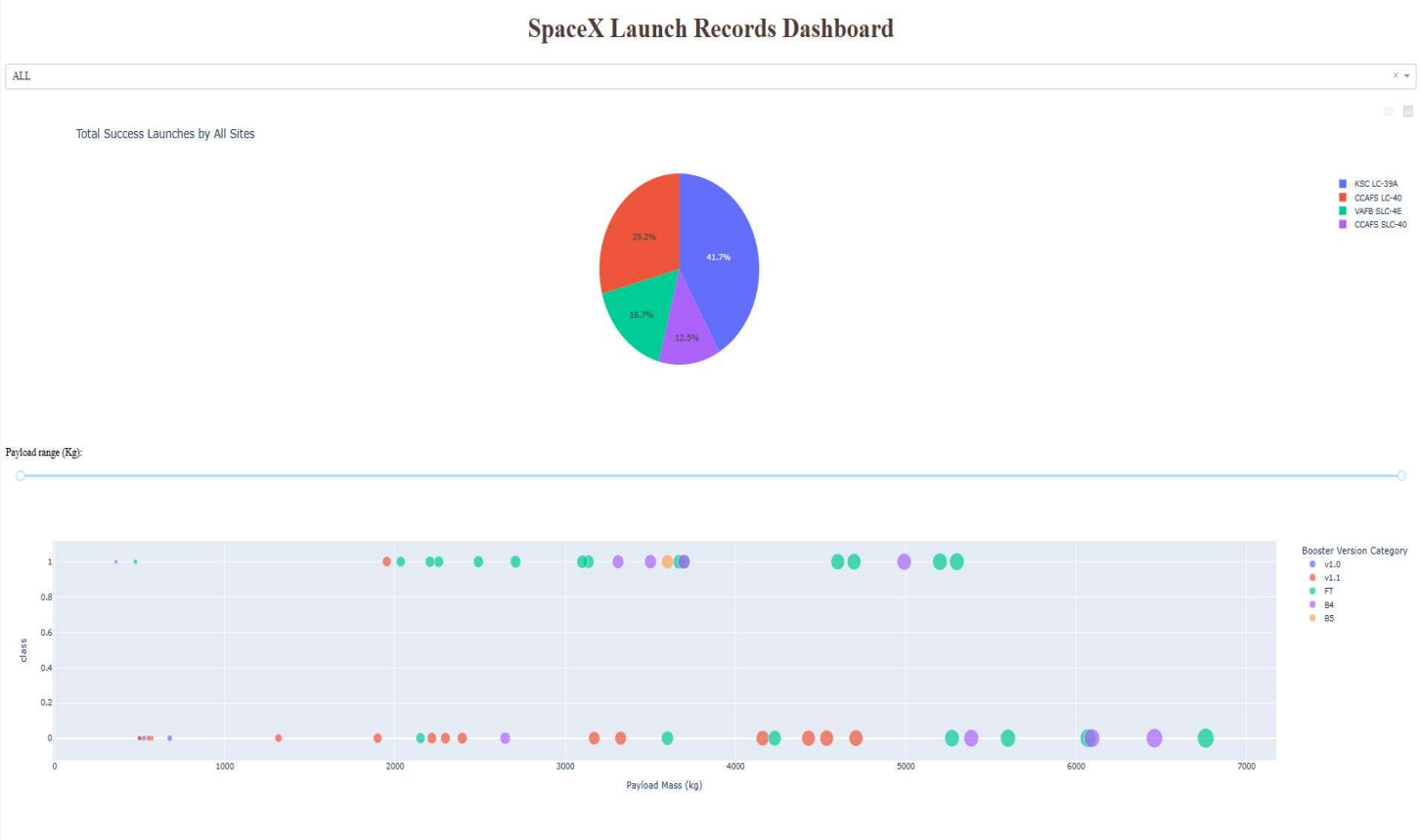
The proximity between KSC to city is 14.7 KM, while to the Ocean is 6.5 km and 0.7 km to the railway



Section 5

Build a Dashboard with Plotly Dash

<SpaceX Launch Records Dashboard>

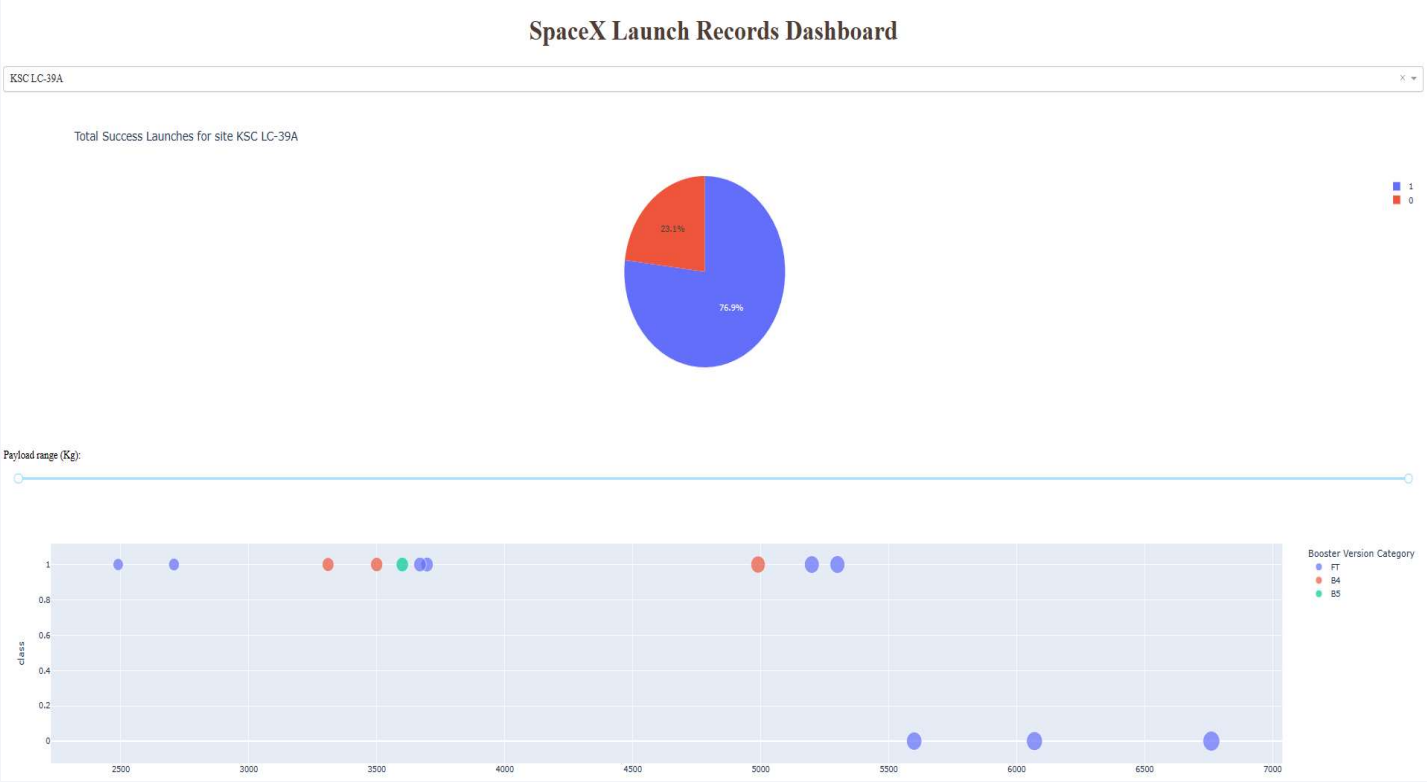


KSC LC-39A has the highest rate of successful launches while CCAFS SLC-40 has the lowest rate of successful launches.

B5 is 100% successful but has only one attempt.

FT has the highest successful landing rate.

<Highest Successful Launch KSC LC- 39A>



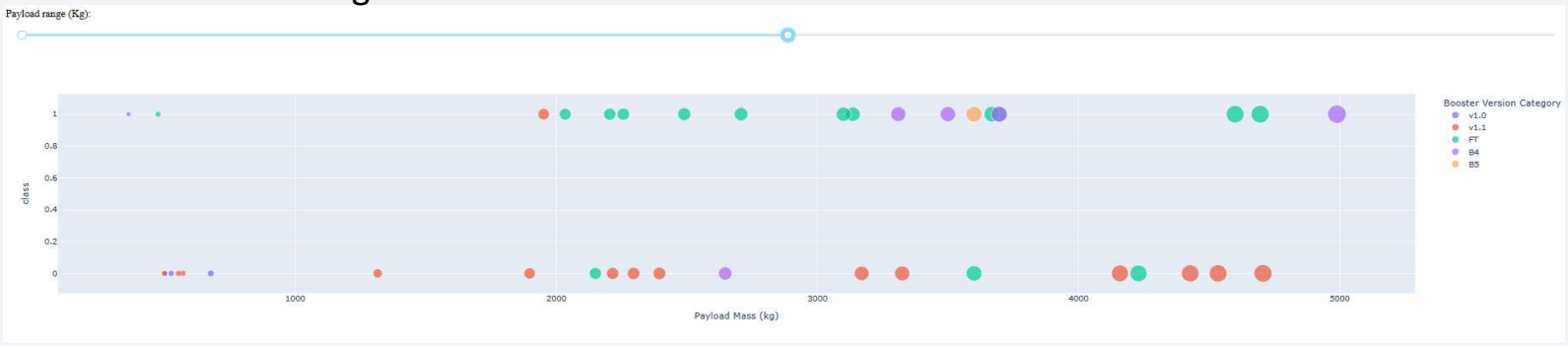
KSC LC 39A has three booster version of FT, B4 and B4.

FT success rate decreases as payload increase from 5500 kg onwards.

B4 yields 100 % successful rate while B5 has only one attempt and is successful

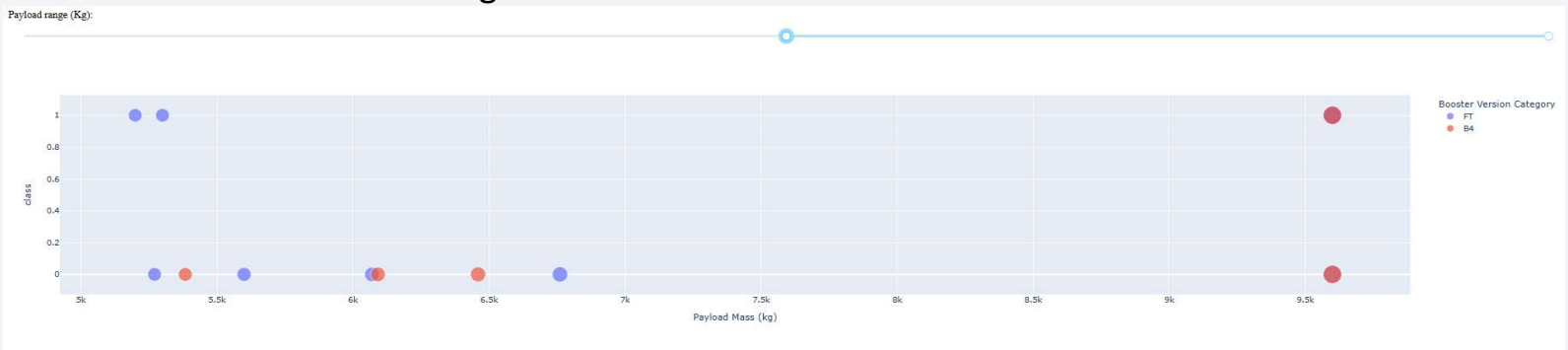
Payload range & Slider

First half: 0 to 5000 kg



There is higher success rate during the first half, 0 to 5000 kg which mainly applies to FT followed by B4. v1.1 exhibits no reasonable relationship between success and payload.

Second half: 5000 to 10000 kg

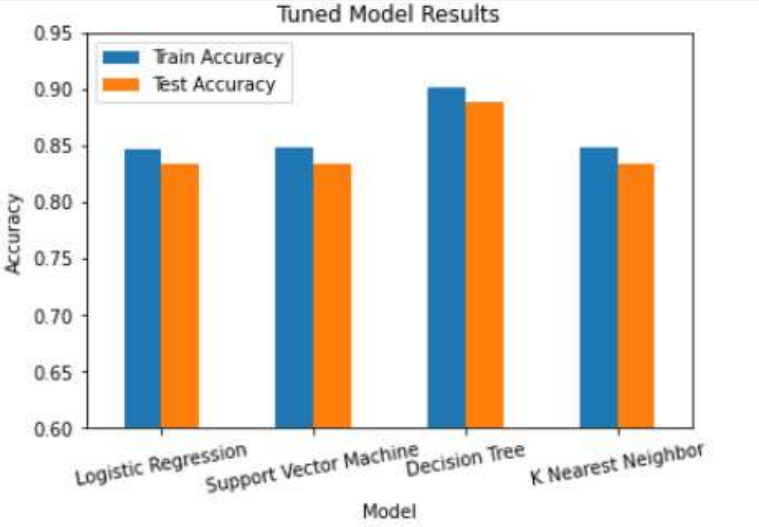


As payload increases above 5000 kg, Only FT and B4 boosters are available. Both which indicates a low success rate after 5500kg.

Section 6

Predictive Analysis (Classification)

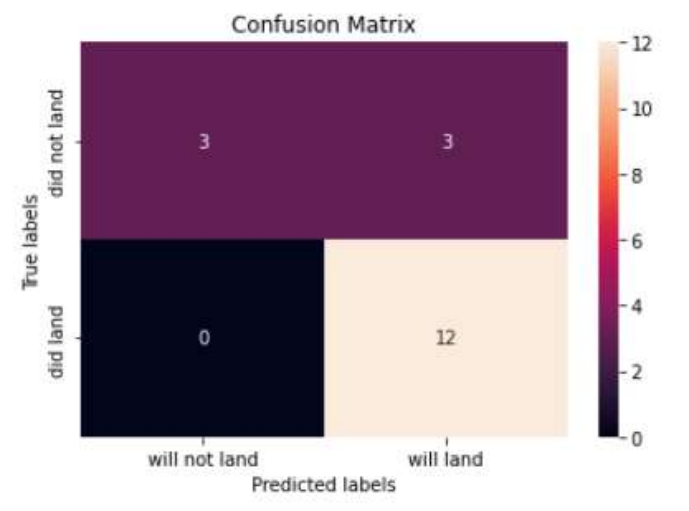
Classification Accuracy



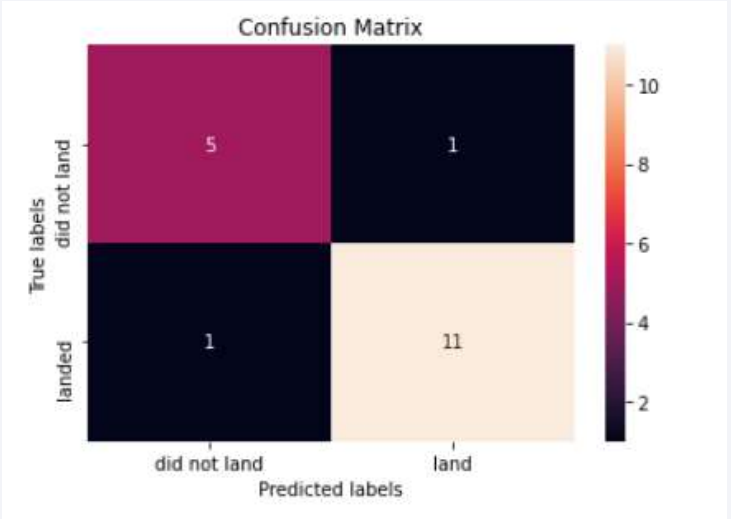
- Decision tree returns the highest train and test accuracy.
- Others return similar train/test accuracy.

Confusion Matrix

Logistic Regression, Support Vector Machine and K Nearest Neighbor return similar confusion Matrix



Decision Tree Classifier returns a confusion matrix which indicates high true positives and higher true negatives.



Conclusions

- Payload Mass, Orbit Type, and Launch Site may have an impact on the outcome
- A decision tree is recommended as it returns a highest train accuracy of 0.90 and test accuracy of 0.88
- Minimum train/test accuracy is at 83.3% for booster recover with Errors are mainly false positives
- Decreasing payload seems to yield higher success rates

Appendix

Libraries

Pandas: pandas.DataFrame — pandas 1.3.3 documentation (pydata.org)
Numpy: NumPy quickstart — NumPy v1.21 Manual
Scikit: API Reference — scikit-learn 1.0 documentation
Folium: folium · PyPI
MatplotLib: Pyplot tutorial — Matplotlib 3.4.3 documentation
Seaborn: seaborn: statistical data visualization — seaborn 0.11.2 documentation (pydata.org)

Machine Learning

Logistic regression – Wikipedia
Support-vector machine – Wikipedia
Decision tree learning – Wikipedia
k-nearest neighbors algorithm - Wikipedia

Thank you!