IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jean-Robert Richard
February 18 2023

Outline

Executive Summary

The goal of this research is to analyze SpaceX Falcon 9 data collected from various sources and use Machine Learning (ML) models to predict the success of first stage landing, which will be useful information to company Space Y, which similar to SpaceX and considering if they should compete directly against SpaceX after considering all the costs.

In order to collect and analyze the data, build and evaluate the ML models, as well as make predictions, the following methods were applied:

- Data collection through API and also web scraping using BeautifulSoup
- Data wrangling to transform the data (define Class 0 or 1 based on Outcome)
- Data analysis using SQL and Data visualization using Seaborn package
- Interactive Map generation using Folium package to analyze proximity of launch site to coast and nearby city
- Dashboard creation using Plotly Dash to seek further insights into launch records
- ML model building to predict success likelihood of first stage of SpaceX Falcon 9 along with evaluation of the models

In addition to highlight the methodology used, the report also provides results from the data analysis showing that from the dataset there is a ~67% overall success rate for first stage landings. However, one will see that there is a higher success rate in the more recent flights (indicated by higher flight numbers), with some negative correlation between successful landing and low payload masses (<2000 kg). Engaging with the interactive map, one will notice that the launch sites are relatively close to the coastline, railroads and highway but not too close to nearby cities. With respect to the problem statement, although the ML models built show ~83% accuracy against the test data, the sample size for the test data was small (18) as well as that used to build the model (72), and caution should be exercised in adopting it wholeheartedly.

Introduction

Background/Context

We are already living in the commercial space-age. Companies like Virgin Galactic and Blue Origin are working to make space travel affordable for all. The most successful of these companies, SpaceX, currently has a competitive advantage in the industry due to the lower cost of its launches at $62 million versus the competition cost of more than $165 million per launch.

SpaceX achieves its cost savings advantage through it ability to re-use the first stage rocket after it has landed successfully. The ability to predict successful landings of he first stage can help in the determination of the predicted cost of launches and can show the feasibility of competing against SpaceX.

Problems to be answered

Through the report, one should get answers to the following questions:

- What are the variables/factors that influence a successful first stage rocket landing?
- What is the impact of different variables and their interaction on the success rate of first stage rocket landings? e.g. Launch site, Payload Mass, Booster Version, Orbit type etc
- Is there a correlation between launch site and success rate?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected after a request to the SpaceX API (v4), with the response being captured in json format, normalized and then loaded into a Pandas Dataframe
  - Data was also web scraped through a request to the Wikipedia site called “List of Falcon 9 and Falcon Heavy Launches”, parsing using BeautifulSoup, afterwards loading created dictionaries into a Pandas Dataframe
- Perform data wrangling
  - One-hot encoding was used for converting mission outcomes (0 – unsuccessful, 1-successful) to determine labels that were used during supervised learning
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Column for ‘Class’ was created; Data was standardized and transformed; Data was split in train/test data; Various classification models were built (Logistic regression, SVM, Decision Tree and KNN) and used against the test data and evaluated to determined the best one in terms of different metrics (F1-score, Jaccard, Accuracy)

Data Collection

The dataset for this project was collected via two methods:

- Data was collected after a request to the SpaceX API (v4), with the response being captured in json format, normalized and then loaded into a Pandas Dataframe
- Data was also web scraped through a request to the Wikipedia site called “List of Falcon 9 and Falcon Heavy Launches”, parsing using BeautifulSoup, afterwards loading created dictionaries into a Pandas dataframe

Using SpaceX API

API request	• Request launch data from SpaceX API
Read API response	• Read .json response and convert to Dataframe (normalized), called data
Initial Data subset and build dataset	• Subset data to keep only the desired features and perform initial feature engineering, then build dictionary using helper functions to construct the dataset
Wrangle Data and Convert to CSV	• Convert dictionary to Dataframe, address missing values with data wrangling and convert into .csv format for next step

Web scraping data from Wikipedia website

HTTP GET request	• Request Falcon 9 html page using the HTML GET method as an HTTP response
Create BeautifulSoup object	• Use the HTML response to create a BeautifulSoup object
Extract HTML tables; Parse data to dictionary	• Collect all relevant headers as column names and update dictionary values using the column names as keys
Convert to Pandas Dataframe and then to CSV	• Build Dataframe from dictionary and export as .csv format for next step

Data Collection – SpaceX API

[Click here to link to GitHub URL of completed SpaceX API calls notebook]

1. API Request of rocket launch data and read json response into Pandas Dataframe (data)
2. Declare global variables
3. Use helper functions with API calls to update global variables
4. Construct dataset using data combining columns into a dictionary
5. Convert dictionary into Pandas Dataframe (launch_df), filter dataframe to only include Falcon 9 launches, address missing values and export to CSV

1. Create API GET request, normalize data and read into Pandas Dataframe (data).

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

response = requests.get(spacex_url)
```

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

2. Declare global variables to store data from requests in helper functions which have additional API GET requests

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

3. Use helper functions to get relevant data about launches using the IDs given for each launch. Only selected features were included

```
# Call getBoosterVersion
getBoosterVersion(data)
```

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

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

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

4. Construct dataset using data combining column to 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}
```

5. Convert dictionary to Pandas Dataframe (launch_df), filter on Booster version not equal to Falcon 1 to get Falcon 9, deal with missing values and export to CSV.

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

# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9= launch_df[launch_df['BoosterVersion']!='Falcon 1']
data_falcon9.head()
```

```
# Calculate the mean value of PayloadMass column
plm_mean=data_falcon9['PayloadMass'].mean()

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

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

Data Collection - Scraping

[Click here to link to GitHub URL of completed web scraping notebook]

1. Obtain HTTP response by performing HTTP GET method to request Falcon9 Launch HTML page
2. Create BeautifulSoup object from response and extract column/variable names from HTML table header in third table, representing first launch table
3. Create empty dictionary with keys from extracted column names
4. Use helper functions to process web scraped table and fill dictionary with launch records
5. Convert dictionary into Pandas Dataframe (df), and export to CSV

1. Obtain HTTP response by performing HTTP GET method to request Falcon9 Launch HTML page

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
✓ 0.4s

# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
✓ 6.4s
```

2. Create BeautifulSoup object from response and extract column/variable names from HTML table header in third table which represents first launch table, using helper function

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
✓ 2.8s

# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
✓ 0.1s

# Let's print the third table and check its content
first_launch_table = html_tables[2]
  column_names = []

  # Apply find_all() function with `th` element on first_launch_table
  # Iterate each th element and apply the provided extract_column_from_header() to get a column name
  # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
  for row in first_launch_table.find_all('th'):
      name = extract_column_from_header(row)
      if name!=None and len(name) >0:
          column_names.append(name)
```

3. Create empty dictionary with keys from extracted 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 value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

4. Use helper functions to process web scraped table and fill dictionary with launch records

```
def date_time(table_cells):

def booster_version(table_cells):

def landing_status(table_cells):

def get_mass(table_cells):
```

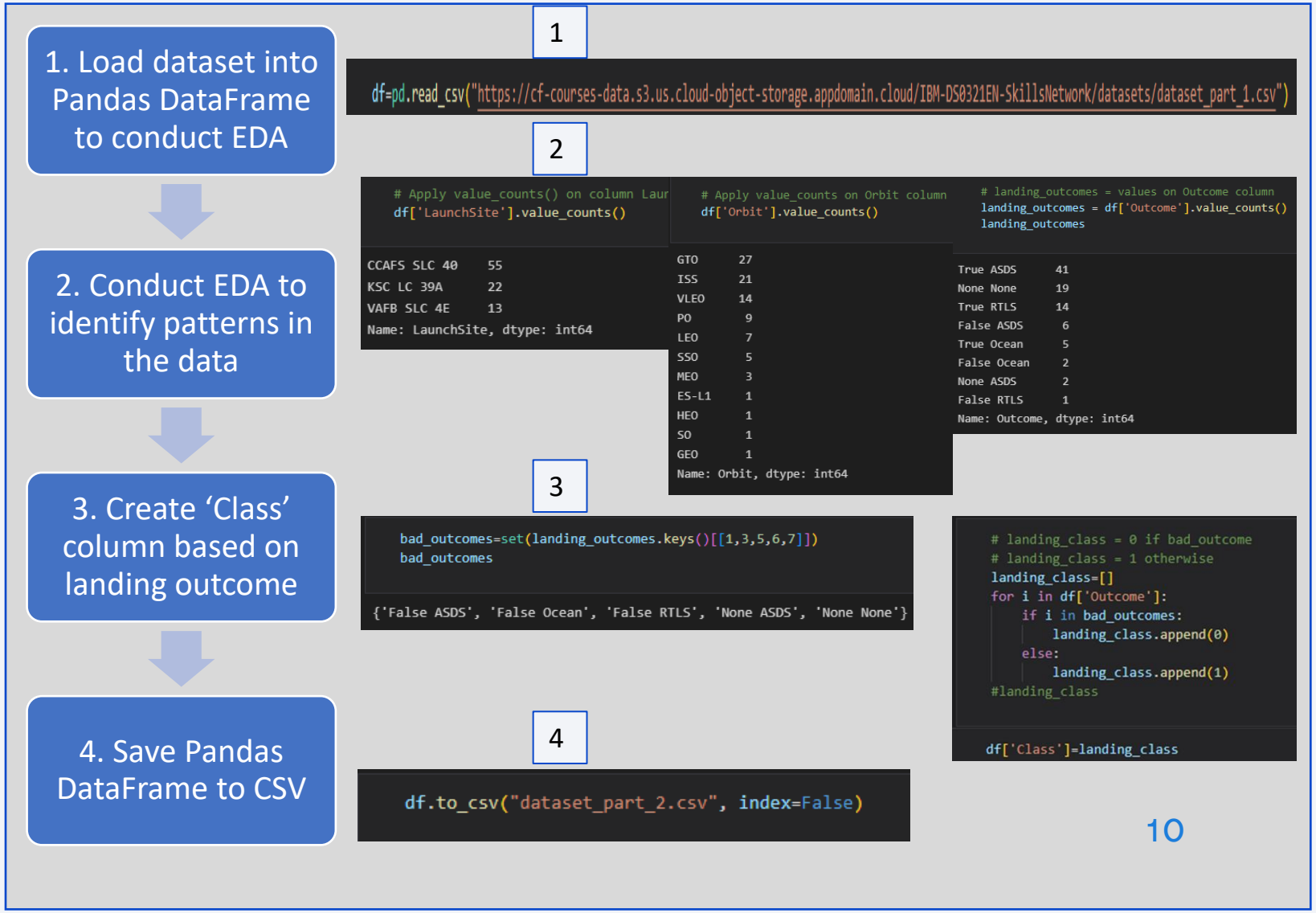
5. Convert dictionary to Pandas Dataframe (df), and export to CSV.

```
df=pd.DataFrame(launch_dict)
✓ 0.1s

df.to_csv('spacex_web_scraped.csv', index=False)
✓ 0.1s
```

Data Wrangling

[Click here to link to GitHub URL of completed data wrangling notebook]

- After loading dataset into Pandas DataFrame, Exploratory Data Analysis (EDA) was conducted to find patterns in data and define labels for training the supervised models.
- Patterns searched for included: number of launches per site, number of the different types of orbits and number and occurrences of mission outcomes
- The 'Class' column was created to be used as a label and it was based on the Outcome column, with descriptors having the word 'None' or 'False' being identified as bad outcomes (0). All others were considered successful landings (1).



EDA with Data Visualization

Click here to link to GitHub URL of completed EDA with data visualization notebook

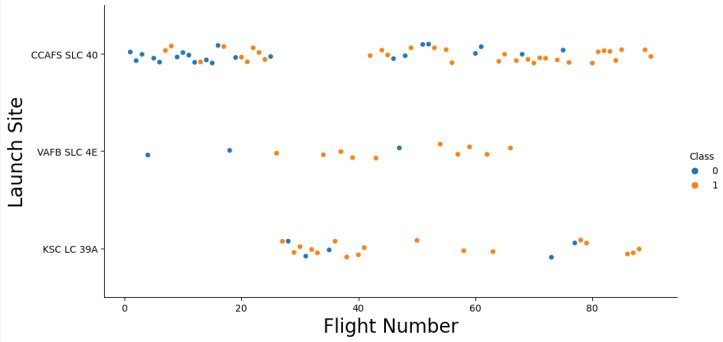
- As part of Exploratory Data Analysis, the following charts were used to gain further insights into the data and to visually represent the data:

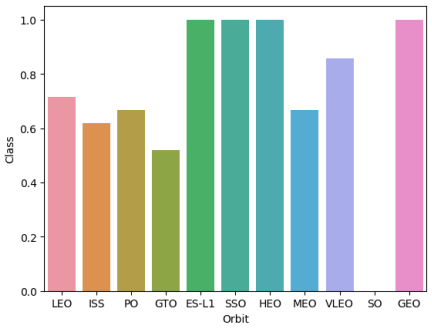
a) Scatterplots (using Seaborn catplot) to look for correlation between two variables and how this relates to successful/unsuccessful first stage landings

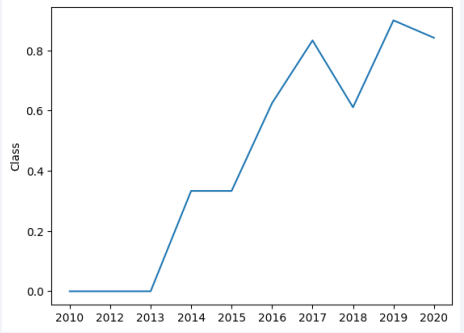
  - Variable-pairs explored for correlation: Flight number vs Payload Mass, Flight number vs Launch Site, Payload Mass vs Launch Site, Flight number vs Orbit type, Payload Mass vs Orbit Type

b) Bar chart (using Seaborn barplot) to look for the rate of successful first stage landings correlated against Orbit type

c) Line chart (using Seaborn lineplot) to observe the trend of rate of successful launches over time (year to year trending)







EDA with SQL

Click here to link to GitHub URL of completed EDA with SQL notebook

- As part of Exploratory Data Analysis, after the CSV file was converted to a Pandas Dataframe and loaded to the SQLite database, using SQL commands/statements, the database was queried to display and list the information below:
  - Unique Launch Site (using DISTINCT)
  - Launch Sites with 'CCA' in name (using WHERE and LIKE)
  - Total Payload Mass for specific NASA (CRS) Customer (using SUM)
  - Average Payload Mass for Booster Version F9 v1.1 (using AVG)
  - Date of first successful landing outcome in ground pad (using MIN(Date))
  - Booster Version names for successful drone ship landing with payload mass between 4000 kg and 6000 kg (using BETWEEN)
  - Total number of successful and failure mission outcomes (using GROUP BY)
  - Booster Version names for those carrying maximum payload mass (using subquery with another SELECT MAX statement)
  - Months, Failure Landing outcomes in Drone Ship, Booster Version and Launch Site in 2015 (using AND statement)
  - Count of successful landing outcomes between June 4 2010 and March 20 2017 in descending order (using WHERE, LIKE, AND, BETWEEN, GROUP BY, ORDER BY and DESC)

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

```
%%sql
SELECT STRFTIME("%Y-%m-%d" ,Date) AS Date, "Landing _Outcome", COUNT(*) AS Successful_Landings
FROM SPACEXTBL
WHERE "Landing _Outcome" LIKE '%Success%'
AND Date BETWEEN '2010-06-04' AND '2017-03-20'
Group By "Landing _Outcome"
ORDER BY Date DESC;
```

Date	Landing _Outcome	Successful_Landings
2016-04-08	Success (drone ship)	5
2015-12-22	Success (ground pad)	3

Build an Interactive Map with Folium

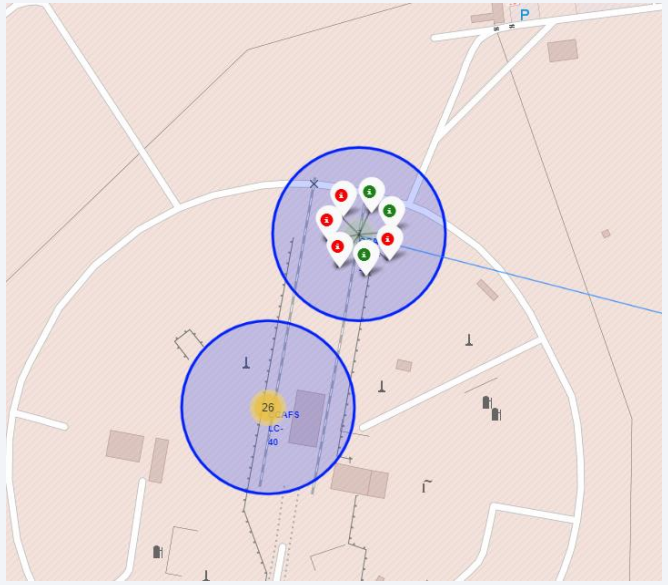
Click here to link to GitHub URL of completed interactive map with Folium map notebook (trust file to see map rendering)

- An map was built using the Folium package to allow for interactive visual analytics to look for more insights related to launch site location.

a) Map objects were created and added to the map to depict:

  - Launch Sites on the map – using folium.circle(*arg) and folium.Marker(*arg) to highlight circle area and have a text label over each launch site
  - Successful and Unsuccessful launch/landings – MarkerCluster() to show markers green for successful launch site/landing and red for unsuccessful landing outcomes
  - Distance and lines between launch sites and nearby landmarks like coastline, railroad, highway and major cities
    - MousePosition (*arg) used to get location of landmark
    - Used folium.Marker(*arg) and folium.Polyline(*arg) to add distance between site and landmark, and also draw a line between site and landmark respectively

b) To calculate the distance, a function was used to find distance between launch sites and the landmarks and helped answer questions about proximity of laucch sites to landmarks



Build a Dashboard with Plotly Dash

Click here to link to GitHub URL of completed Plotly Dash lab python script

- The requirements were to build a Plotly Dash web application to perform interactive visual analytics on SpaceX launch data in real-time.
- Several components were added to allow for insights through the real-time interaction
  a) Added a Launch Site Drop-down Input component to the dashboard to provide an ability to filter Dashboard visual by all launch sites or a particular launch site
  b) Added a Pie Chart to the Dashboard to show total success launches when 'All Sites' is selected and show success and failed counts when a particular site is selected
  c) Added a Range slider to the Dashboard to easily select different payload ranges to look for visual patterns with payload
  d) Added a Scatter Plot chart to observe how payload may be correlated with mission outcomes for selected site(s). The color-label Booster version on each scatter point identified each booster version and the location on the scatter plot (1 or 0) indicated success or not
- The visual analysis from the dashboard permitted the following questions to be answered:
  1. Which site has the largest successful launches? **KSC LC-39A with 10 launches out of total 24 for all sites**
  2. Which site has the highest launch success rate? **KSC LC-39A at 76.9% (10/13)**
  3. Which payload range(s) has the highest launch success rate? **3000-4000 kg (70% success rate)**
  4. Which payload range(s) has the lowest success rate? **<2000 kg and >5000kg**
  5. Which F9 Booster version (v1.0, v1.1, FT, B4, B5) has the highest launch success rate? **FT**

Predictive Analysis (Classification)

Click here to link to GitHub URL of completed predictive analysis lab as a notebook

1. Load datasets into Dataframes and create a numpy array from 'Class' column in **data** Dataframe
2. Standardize the data in the **X** Dataframe
3. Split **X** and **Y** in training and testing data sets using train/test split
4. For different classification algorithms, create and refine models
5. Determine best performing model by comparing scores

1a. Load datasets in Pandas Dataframe

```
data = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/api/dataset_part_2.csv')

X = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/api/dataset_part_3.csv')
```

1b. Create Numpy array from 'Class' column in **data** Dataframe and assign t **Y**

```
Y = data['Class'].to_numpy()
```

2. Standardize data in **X** Dataframe using StandardSclaer, reassigning it to **X**

```
transform = preprocessing.StandardScaler()
X = transform.fit(X).transform(X)
```

3. Split **X** and **Y** in training and testing data sets using train/test split function from Sci-kit learn package

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
```

4a. Create object (e.g Logistic Regression object) and then create GridSearchCV object with pre-defined parameters for the cross valiation. Fit train data into GridSearchCV object to train the model

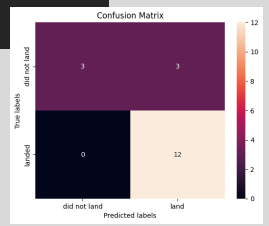
```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```

4b. Find and show best hyperparameters and accuracy score using **.best_params_** and **.best_score_** methods

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

4c. Check accuracy of model against test data using **.score** method and confusion matrix to compare True and Predicted labels

```
lraccuracy = logreg_cv.score(X_test, Y_test)
logreg_yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,logreg_yhat)
```



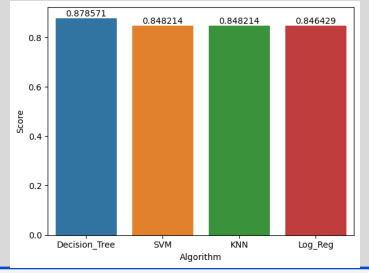
4d. Repeat above steps for different classification algorithms (SVM, DecisionTree, K-Nearest Neighbors)

5a. Compute various scores (Jaccard, F1 and Accuracy (train and test) and enter in Dataframe

```
from sklearn.metrics import jaccard_score, f1_score

jaccard_scores = [
                  jaccard_score(Y_test, logreg_yhat),
                  jaccard_score(Y_test, svm_yhat),
                  jaccard_score(Y_test, tree_yhat),
                  jaccard_score(Y_test, knn_yhat)
                  ]

f1_scores = [
             f1_score(Y_test, logreg_yhat),
             f1_score(Y_test, svm_yhat),
             f1_score(Y_test, tree_yhat),
             f1_score(Y_test, knn_yhat)
             ]

test_accuracy_scores = [lraccuracy, svm_accuracy, tree_accuracy, knn_accuracy]

train_accuracy_scores = [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_]
Metrics2 = np.array([jaccard_scores, f1_scores, test_accuracy_scores, train_accuracy_scores])
Report2 = pd.DataFrame(Metrics2, index=['Jaccard_Score', 'F1_Score', 'Test Accuracy', 'Train Accuracy'], columns = ['LogReg', 'SVM', 'Decision_Tree', 'K-Nearest_Neigh'])
Report2
```

	LogReg	SVM	Decision_Tree	K-Nearest_Neigh
Jaccard_Score	0.800000	0.800000	0.800000	0.800000
F1_Score	0.888889	0.888889	0.888889	0.888889
Test Accuracy	0.833333	0.833333	0.833333	0.833333
Train Accuracy	0.846429	0.848214	0.878571	0.848214

5b. Create bar plot for comparison on train accuracy (test accuracy was the same for all algorithms)

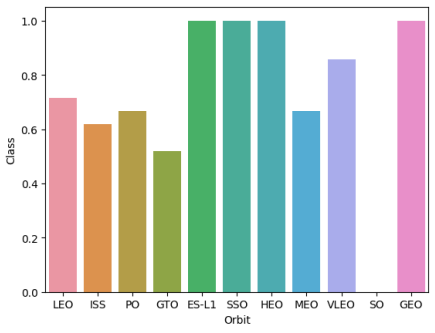


Results

The following will be shown in the Results section:

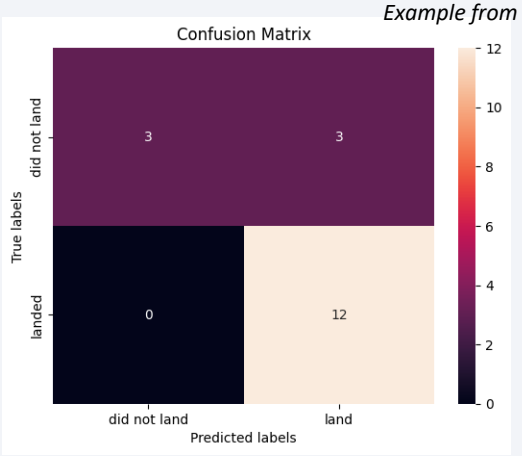
- Exploratory data analysis results
- Interactive analytics demo in screenshots e.g.
- Predictive analysis results

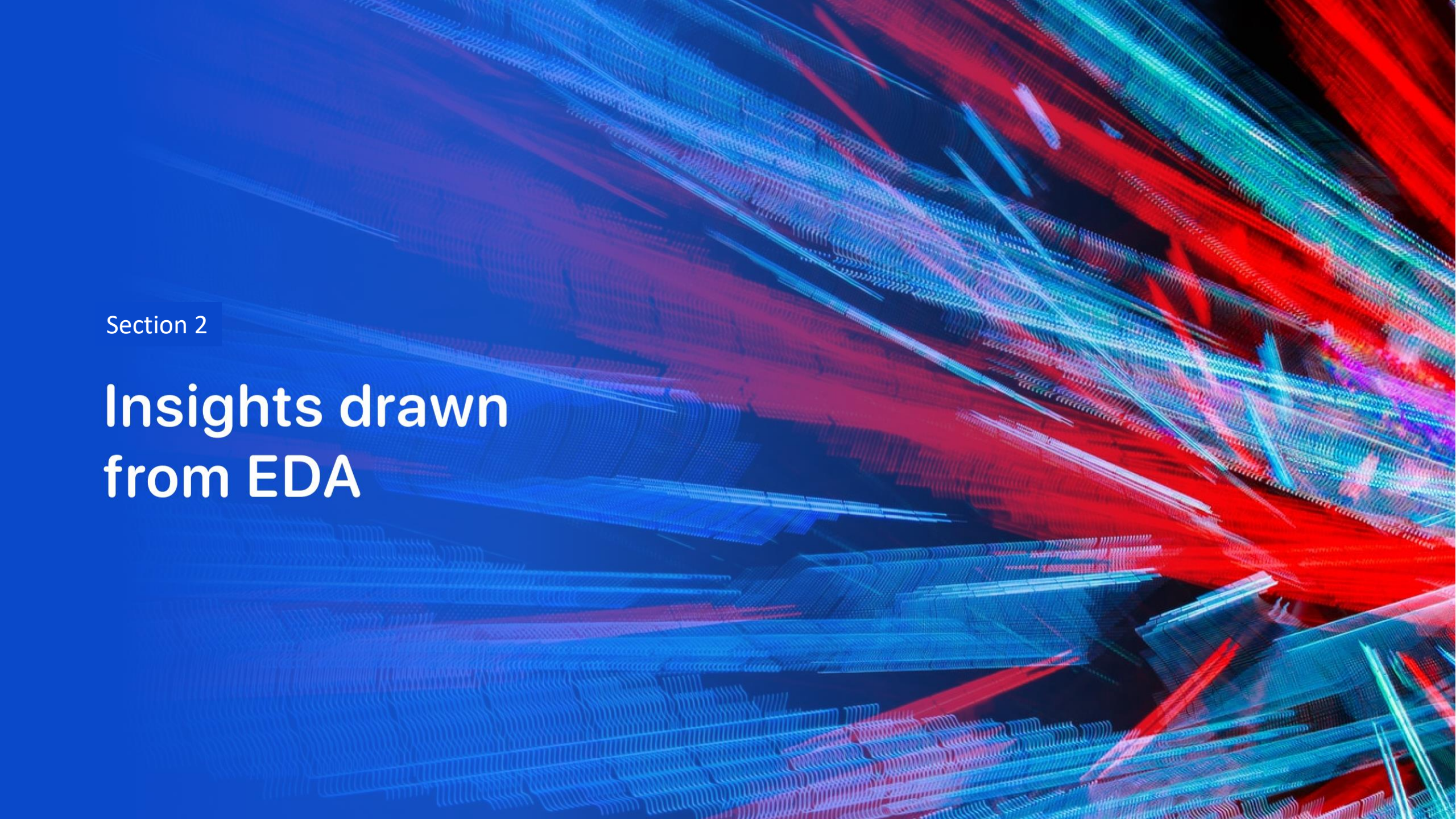
Example from Exploratory Data Analysis



Example from Interactive analytics

Example from Predictive analysis




Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

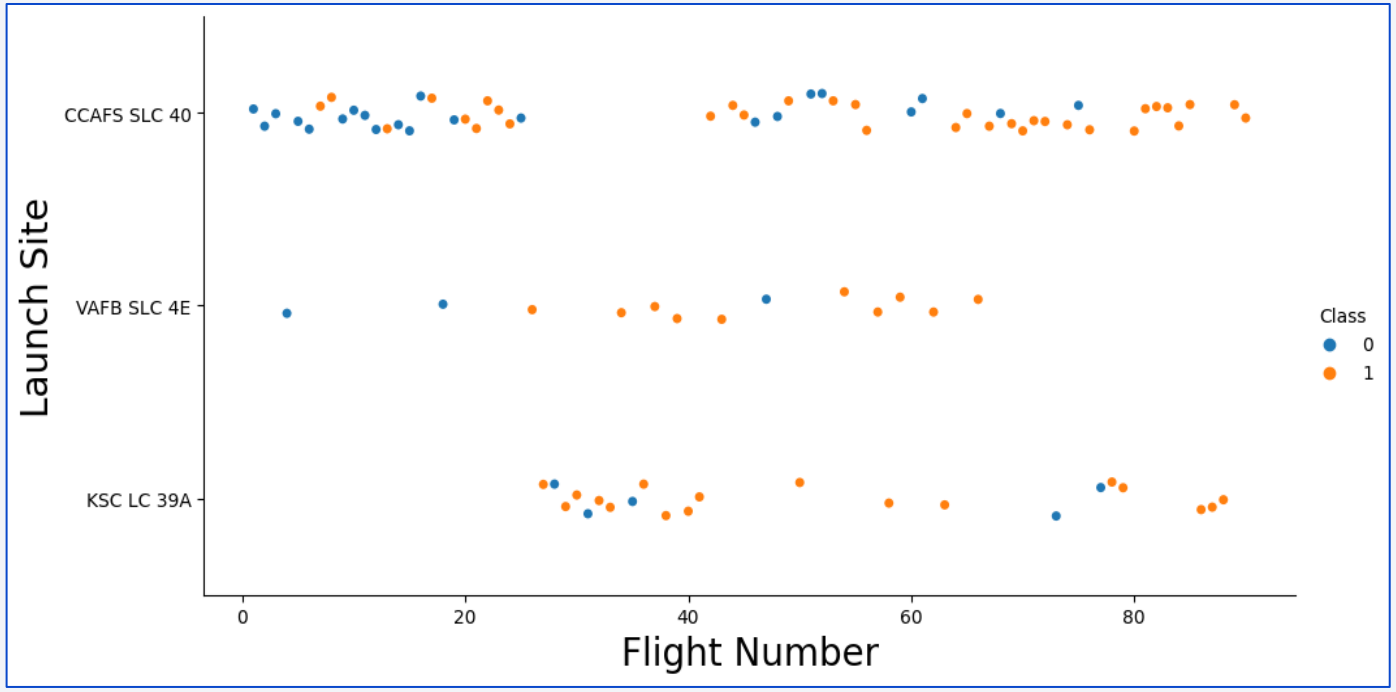


Chart 1 – Scatter plot of Flight Number vs Launch Site

- Success frequency (class 1) increases with the higher flight numbers, which also take place later in time. This is supported line trend analysis which shows higher success rate over time
- Launch Site **KSC LC 39A** shows no record of use before Flight number 27

Payload vs. Launch Site

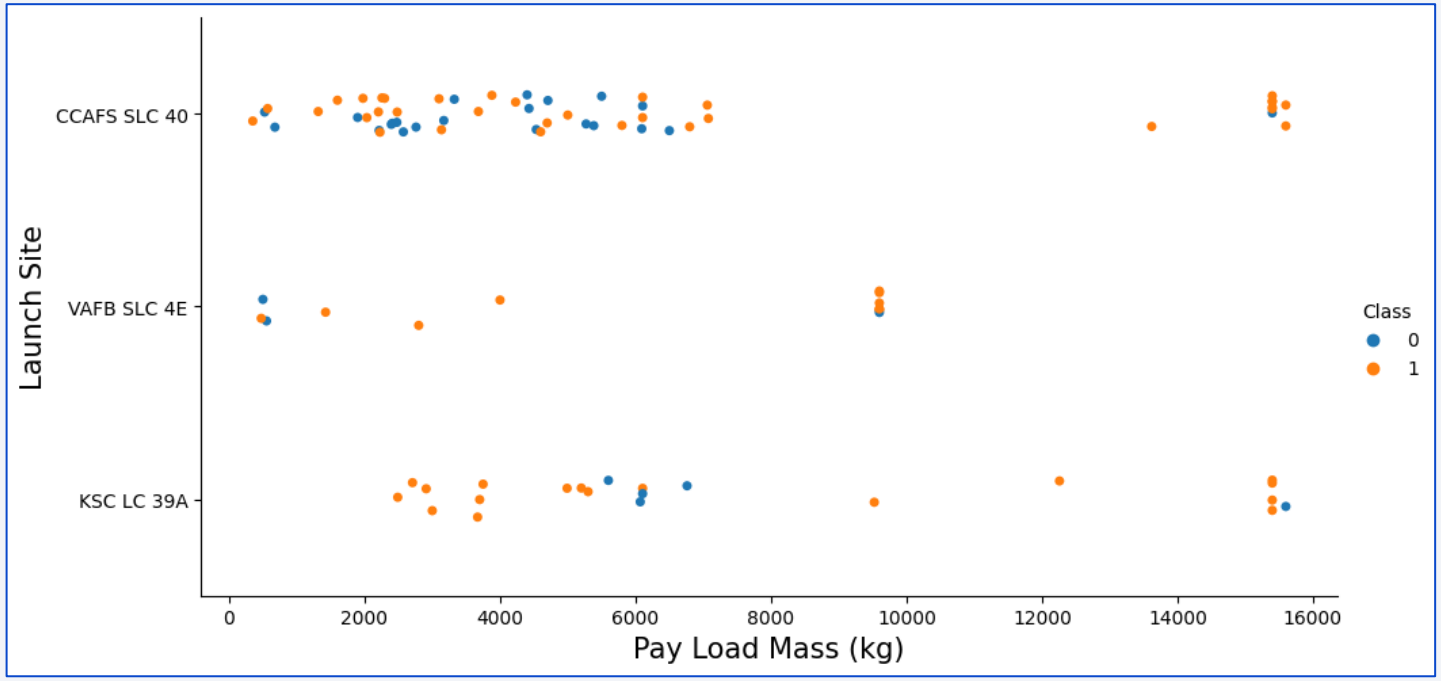


Chart 2 – Scatter plot of Payload Mass vs Launch Site

- For launch site **KSC LC 39A**, there seems to be some higher success rates with pay load masses between 2000 kg and 5000 kg. No real such link can be made for the other sites
- No payload masses above 10000 kg have been launched from site **VAFB SLC 4E**
- In general, it seems that payloads above 9000 kg have good success rates (only 3 failures), regardless of launch site

Success Rate vs. Orbit Type

- Success rate seems to be correlated to the type of orbit with ES-L1, SSO, HEO and GEO having the highest success rates
- The GTO orbit has the lowest success rate

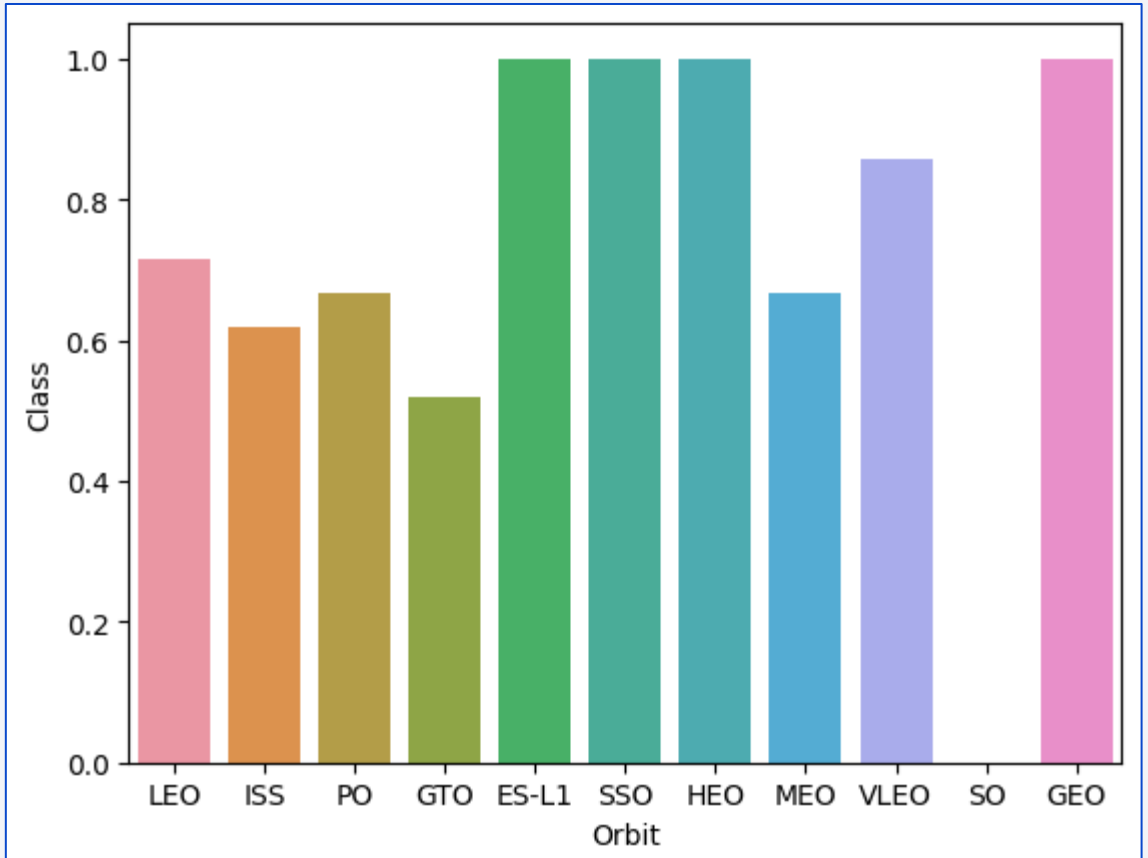


Chart 3 – Bar chart showing Orbit Type vs Success Rate (Class)

Flight Number vs. Orbit Type

- LEO orbit types have stopped after ~ flight number 50, with higher concentration of VLEO-type orbits in more recent flights with higher flight numbers
- VLEO and MEO type orbits started only after ~ flight 60 and only 1 GEO type orbit pursued (SO and SSO are same types, according to documentation)
- High success rates for ISS type orbit after ~ flight 60 as well as for VLEO orbit-type

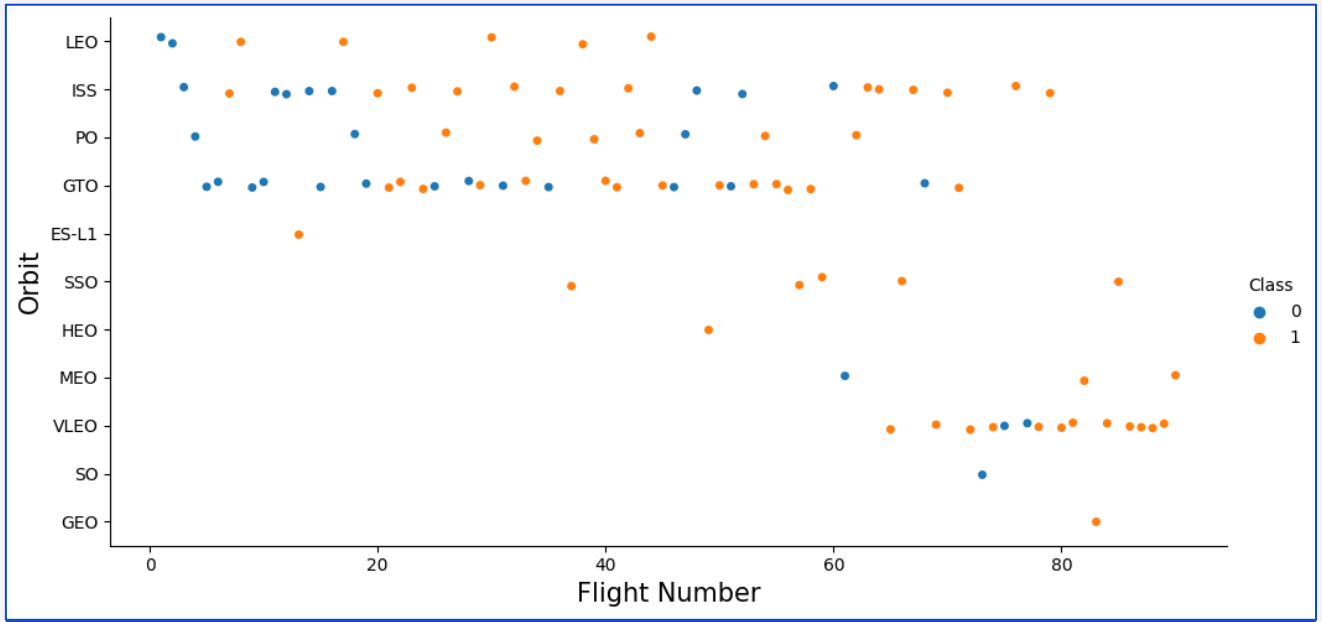


Chart 4 – Scatter plot of Flight Number vs Orbit Type

Payload vs. Orbit Type

- Higher Payload masses used for the VLEO orbit-type (>12500 kg), with mid range Payload masses used for the GTO and MEO orbit types (2500-7500 kg)
- Payload mass seems to have no correlation to success rate for GTO orbit
- Increased Payload mass seems to have positive correlation with ISS, PO and LEO orbit-type success for outcome but it is unclear for SSO (all successful)

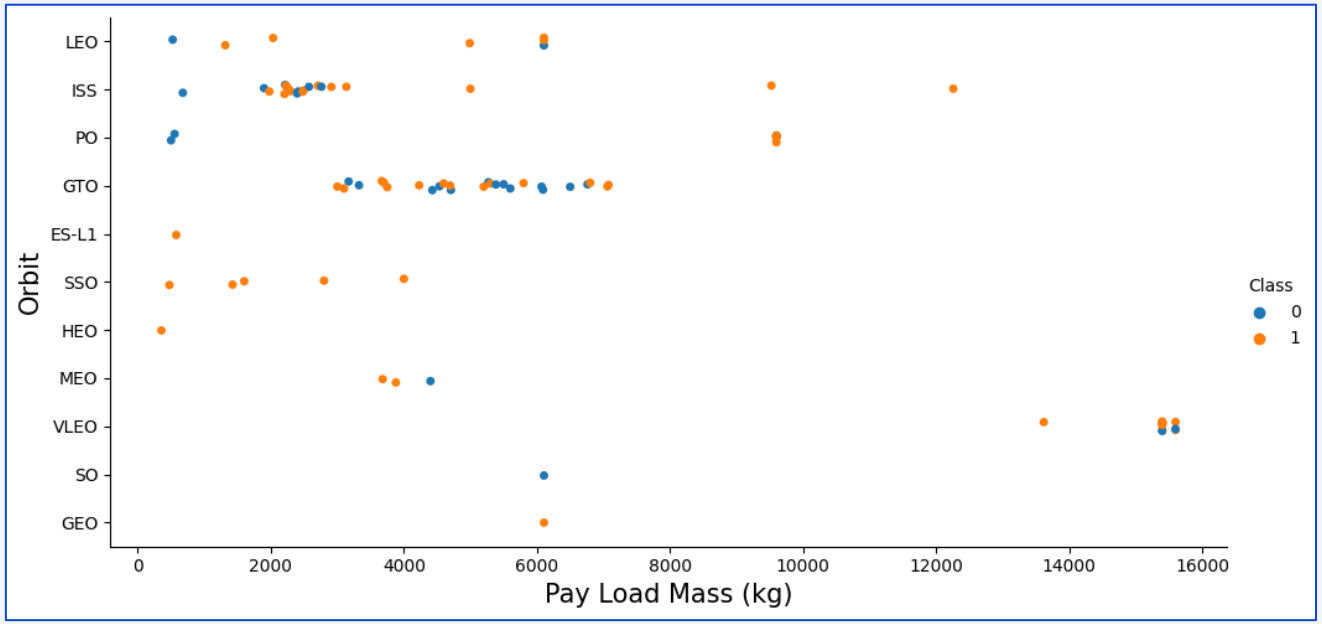


Chart 5 – Scatter plot of Payload Mass vs Orbit Type

Launch Success Yearly Trend

- Average success rate generally increases over time with small dip in 2018.
- Rate started at 0% during first 3years to progress to ~80% success rate in 2020

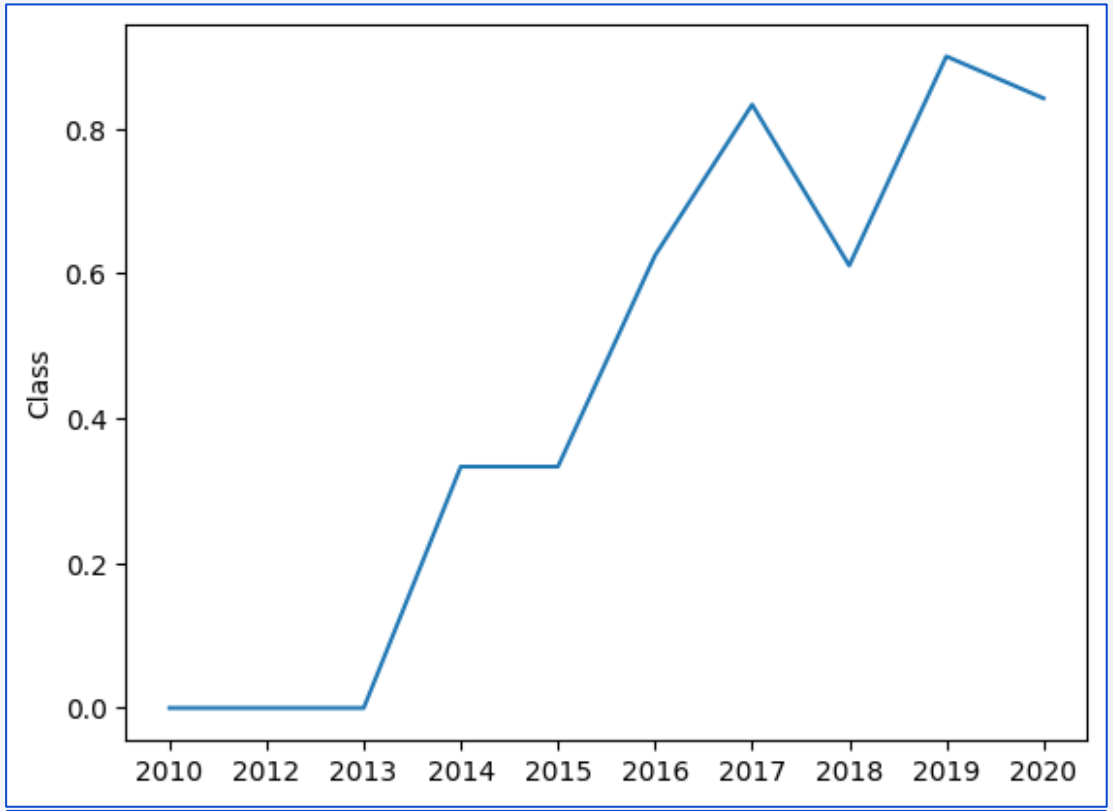


Chart 6 – Line plot of Yearly Average Launch Success rate trend

All Launch Site Names

QUERY

```
%%sql
SELECT DISTINCT "Launch_Site"
FROM SPACEXTBL
```

DESCRIPTION

- SELECT DISTINCT returns only unique values from the Launch_Site column

RESULT

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

Launch Site Names Begin with 'CCA'

QUERY

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

DESCRIPTION

- SELECT * to choose all columns
- LIKE to specify string to look for (CCA% in this case for starting with CCA and ending with anything)
- LIMIT 5 to limit the returned records to 5

RESULT

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
2010-06-04 00:00:00	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08 00:00:00	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 00:00:00	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 00:00:00	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01 00:00:00	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass for boosters launched by NASA (CRS)

QUERY

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_), Customer
FROM SPACEXTBL
WHERE Customer== 'NASA (CRS)';
```

DESCRIPTION

- SUM adds values in column **PAYLOAD_MASS__KG_** with WHERE used to identify **Customer** NASA (CRS)

RESULT

SUM(PAYLOAD_MASS__KG_)	Customer
45596	NASA (CRS)

Average Payload Mass by F9 v1.1

QUERY

```
%%sql
SELECT Booster_Version, AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE "Booster_Version" = 'F9 v1.1';
```

DESCRIPTION

- AVG keyword used to calculate average within **PAYLOAD_MASS__KG_** column
- WHERE specifies **Booster_Version** to perform calculation against (**F9 v1.1**)

RESULT

Booster_Version	AVG(PAYLOAD_MASS__KG_)
F9 v1.1	2928.4

First Successful Ground Landing Date

QUERY

```
%%sql
SELECT "Landing _Outcome", STRFTIME("%Y-%m-%d" ,MIN(Date)) AS First_Date
FROM SPACEXTBL
WHERE "Landing _Outcome" = 'Success (ground pad)';
```

DESCRIPTION

- Select relevant columns to display (aliased column name as **First_Date**)
- STRFTIME used to interpret string format of time as time format Year, Month, Date
- MIN looks for lowest value for time within the **Landing _Outcome** column under **Success (ground pad)**

***Note**: Prior to loading into SQL database, Date column was converted to datetime using Pandas to_datetime method.*

RESULT

Landing _Outcome	First_Date
Success (ground pad)	2015-12-22

Successful Drone Ship Landing with Payload greater than 4000 and less than 6000

QUERY

```
%%sql
SELECT "Landing _Outcome",Booster_Version, PAYLOAD_MASS__KG_
FROM SPACEXTBL
WHERE "Landing _Outcome" = 'Success (drone ship)'
AND PAYLOAD_MASS__KG_ BETWEEN 4001 AND 5999;
```

DESCRIPTION

- Select relevant columns to display
- WHERE to return results for when **Landing _Outcome** is for **Success (drone ship)**
- AND connect WHERE and BETWEEN
- BETWEEN for Payload mass range to search within

RESULT

Landing _Outcome	Booster_Version	PAYLOAD_MASS__KG_
Success (drone ship)	F9 FT B1022	4696
Success (drone ship)	F9 FT B1026	4600
Success (drone ship)	F9 FT B1021.2	5300
Success (drone ship)	F9 FT B1031.2	5200

Total Number of Successful and Failure Mission Outcomes

QUERY

```
%%sql
SELECT "Mission_Outcome", COUNT("Mission_Outcome") AS 'Outcome'
FROM SPACEXTBL
GROUP BY "Mission_Outcome" LIKE '%Success%';
```

DESCRIPTION

- GROUP BY to aggregate similar Mission outcomes
- LIKE for all with “Success” string in column **Mission_Outcome**
- COUNT for number of occurrences for all the aggregated outcomes done within GROUP BY

RESULT

Mission_Outcome	Outcome
Failure (in flight)	1
Success	100

Boosters Carried Maximum Payload

QUERY

```
%%sql
SELECT Booster_Version, PAYLOAD_MASS__KG_
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

DESCRIPTION

- Select relevant columns to display in records
- Subquery executed to SELECT MAX Payload mass within **PAYLOAD_MASS__KG_** column
- MAX used to identify maximum value in column
- WHERE returns records for **PAYLOAD_MASS__KG_** matching results from subquery

RESULT

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records for Failure in drone ship landings

QUERY

```
%%sql
SELECT STRFTIME("%m", Date) AS Month, "Landing _Outcome" AS Outcome, Booster_Version, Launch_Site
FROM SPACEXTBL
WHERE STRFTIME("%Y", Date) = '2015' AND "Landing _Outcome" = 'Failure (drone ship)';
```

DESCRIPTION

- Select relevant columns to display (aliased STRFTIME ("%m", Date) as **Month**)
- STRFTIME used to interpret string format of time as time format Year, Month, Date
- WHERE to filter year portion of date (STRFTIME ("%m", Date)) for year **2015**
- AND to intersect filter with Landing outcome being **Failure (drone ship)**

RESULT

Month	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 Successful Landing Outcomes Between 2010-06-04 and 2017-03-20

QUERY

```
%%sql
SELECT STRFTIME("%Y-%m-%d" ,Date) AS Date, "Landing _Outcome", COUNT(*) AS Successful_Landings
FROM SPACEXTBL
WHERE "Landing _Outcome" LIKE '%Success%'
AND Date BETWEEN '2010-06-04' AND '2017-03-20'
Group By "Landing _Outcome"
ORDER BY Date DESC;
```

DESCRIPTION

- Select relevant columns and alias as necessary
- COUNT (*) for number of records matching filters applied through GROUP BY, WHERE and BETWEEN
- GROUP BY to aggregate similar Landing outcomes
- WHERE to return results when **Landing _Outcome** has **Success** in string
- BETWEEN for Date range to search within
- ORDER BY and DESC starting with most recent date first

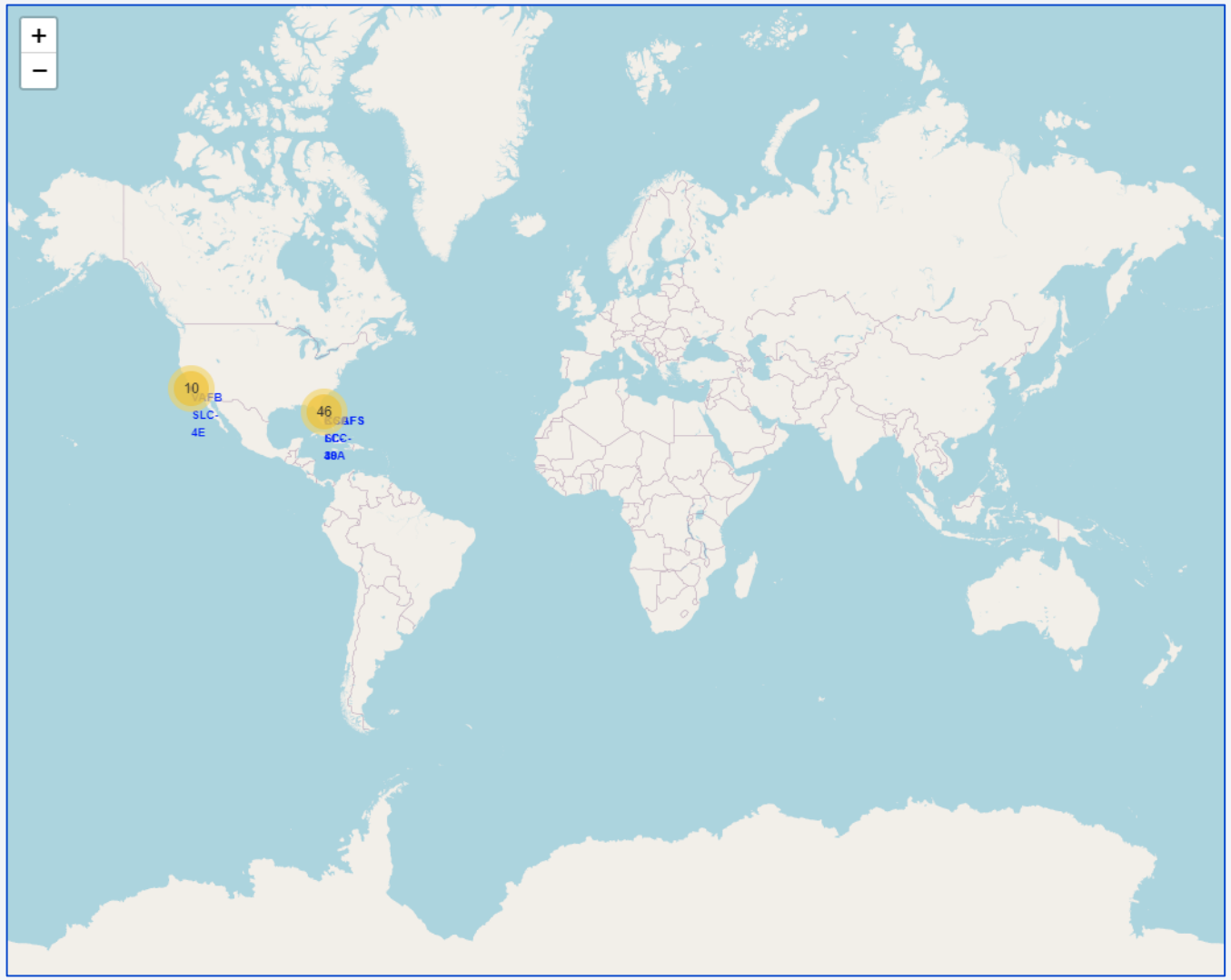
RESULT

Date	Landing _Outcome	Successful_Landings
2016-04-08	Success (drone ship)	5
2015-12-22	Success (ground pad)	3

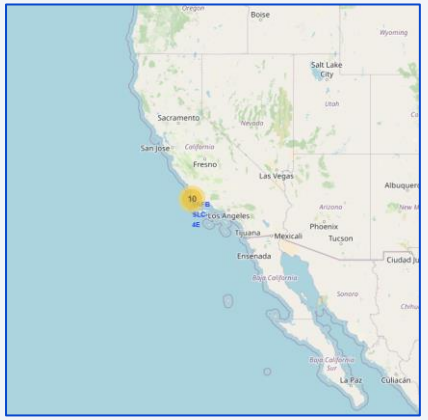
Section 3

Launch Sites Proximities Analysis

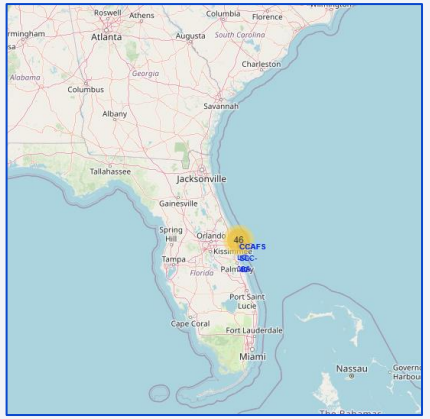
Map showing Space X Falcon9 launch sites location markers on the East and West coasts



- Launch sites on the global map on both coasts of the USA (Florida and California)
- Launch sites also located at lower part of the USA, where the weather is warmer with less likelihood of winter weather impacting launches



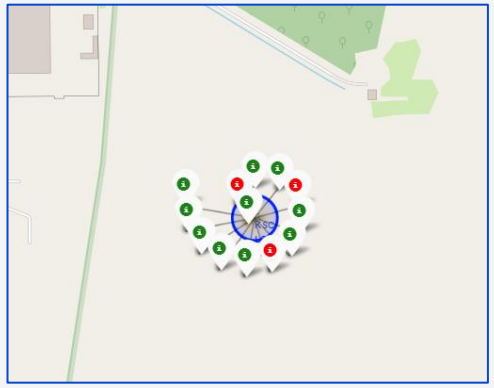
West coast (California) with 1 Launch site)



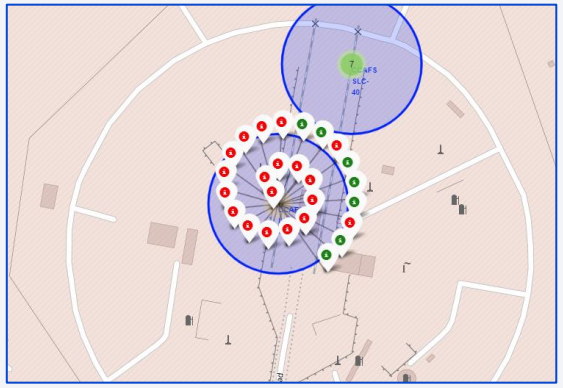
East coast (Florida) with 3 Launch sites)

SpaceX Falcon9 Successful/Failed Launch Outcome Map based on Launch sites

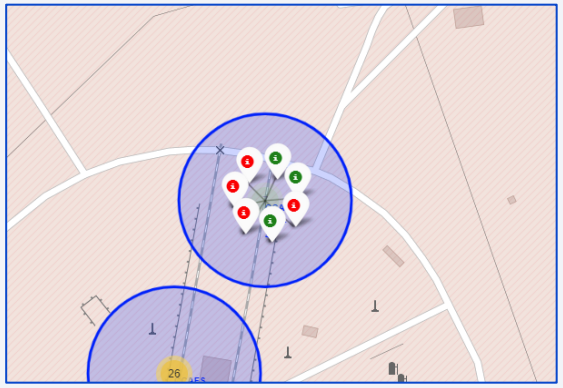
VAFB SLC - 4E launch site in California with success (green)/fail (red) markers



KSC LC-39A launch site in Florida with success (green)/fail (red) markers



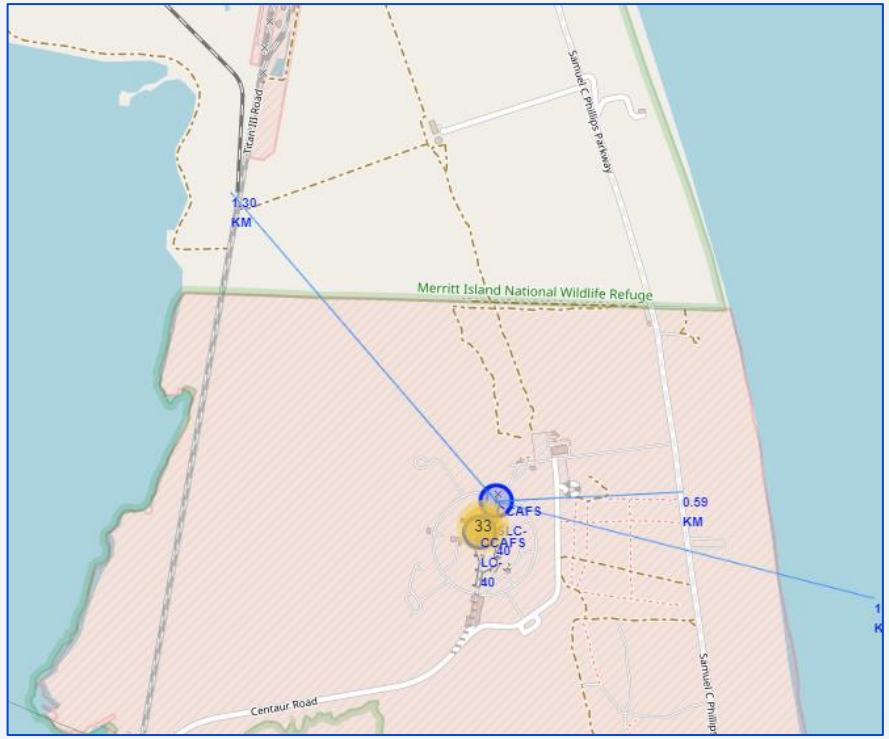
CCAFS LC-40 launch site in Florida with success (green)/fail (red) markers



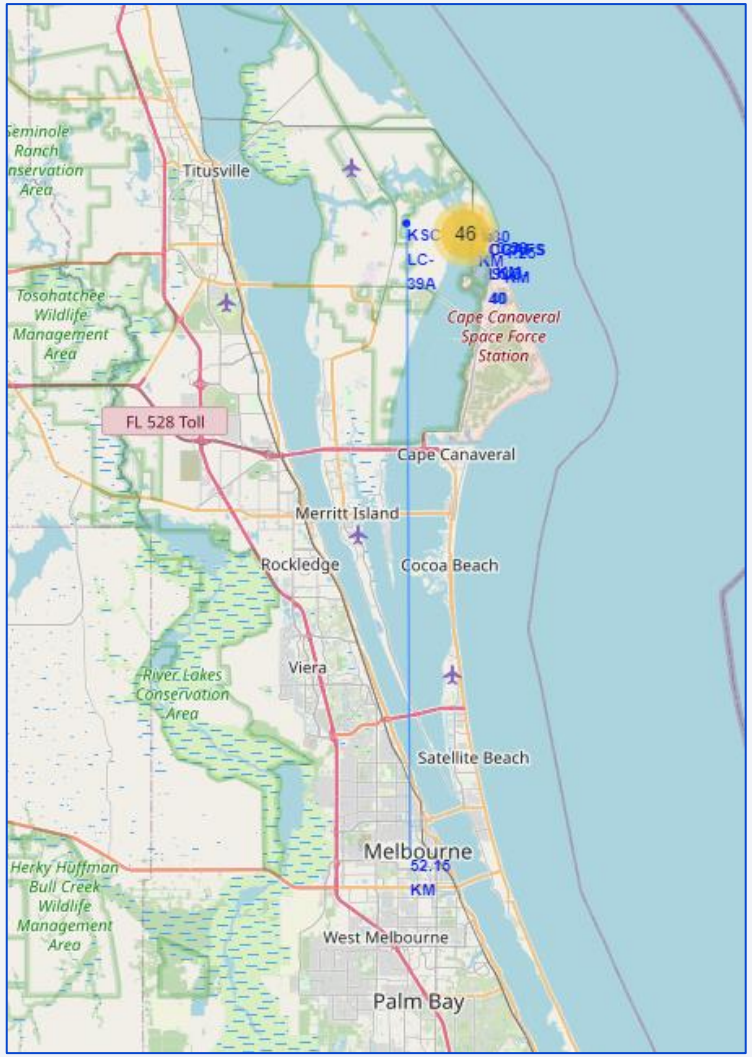
CCAFS SLC-40 launch site in Florida with success (green)/fail (red) markers

- Most launches took place at CCAFS LC-40 but a quick comparison shows KSC LC-39A having the highest success rate for launches (ratio of green markers vs total markers for site)

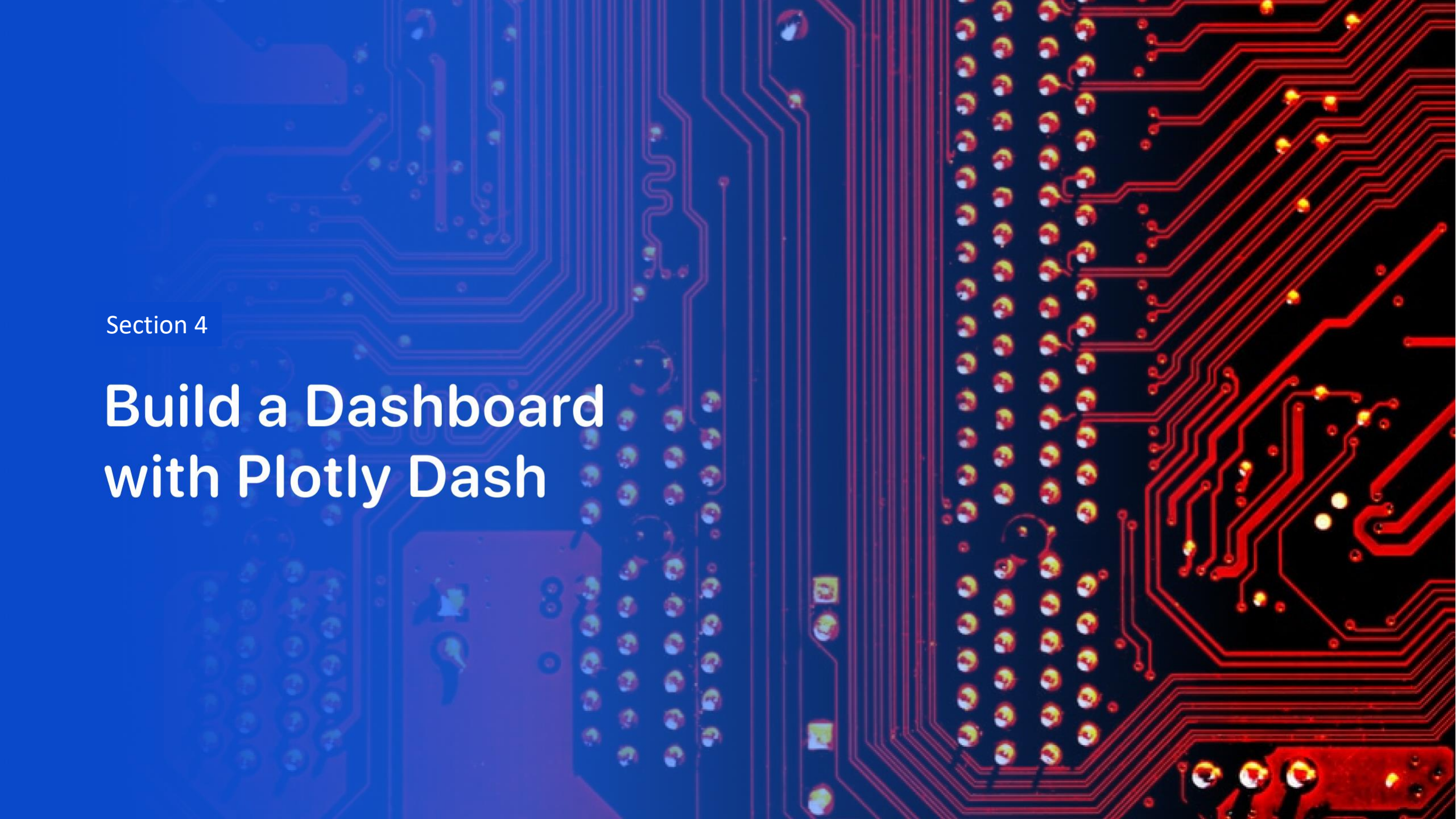
Map showing proximity of launch site(s) to selected landmarks



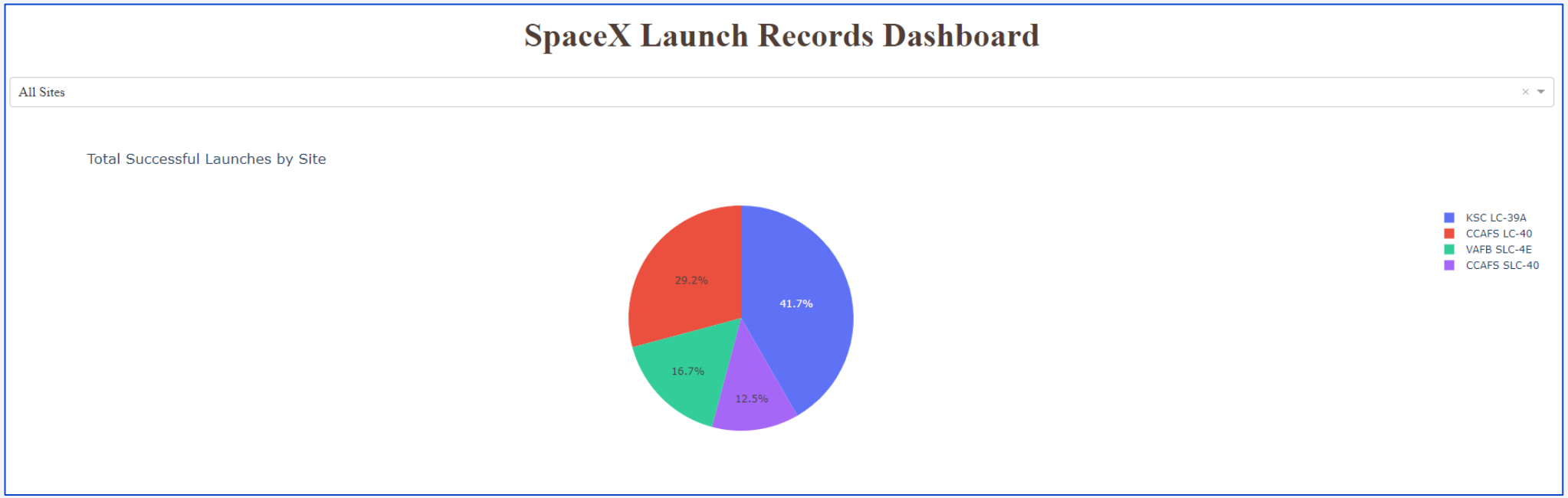
- Launch site in relatively close proximity to railroad, highway and coastline (>2km), which are all useful for transportation of people and material to the launch sites



- Launch site is located a significant distance away from major cities e.g. Melbourne being 52 km away, to minimize impact of potential accident at the launch site location

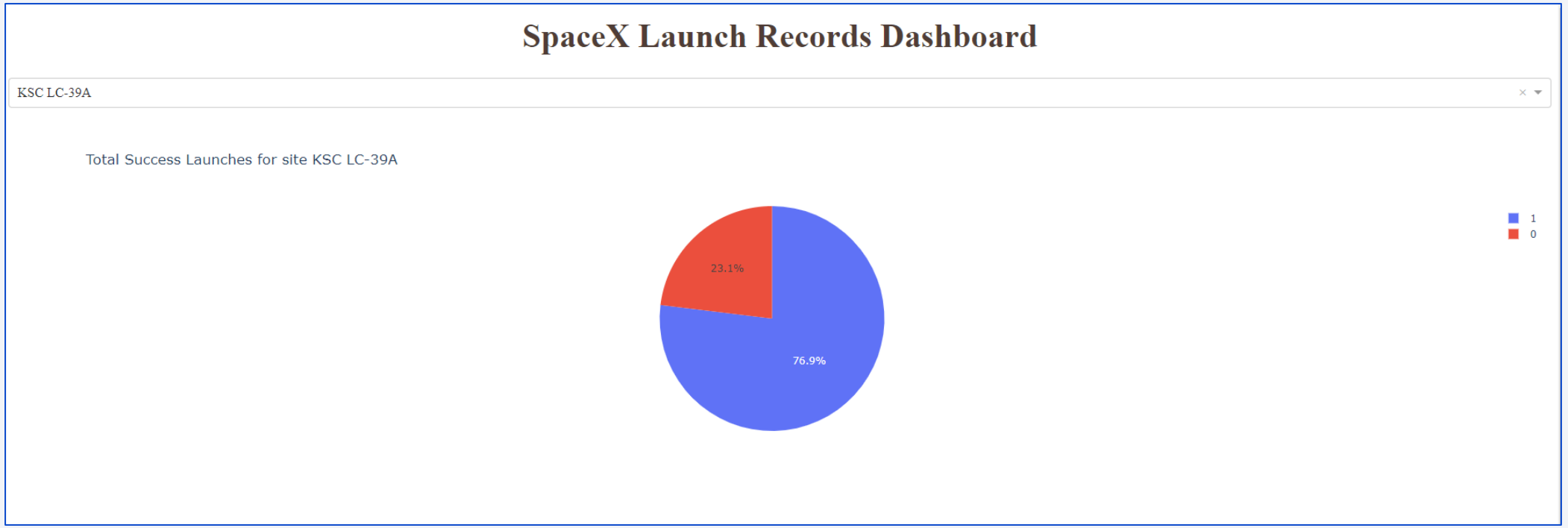

Section 4
Build a Dashboard with Plotly Dash

Percentage contribution by Launch Site of Total Successful Launches



- Launch site KSC LC-39A had the highest number of successful launches (10 successful launches or 41.7% of total 24)
- Launch site CCAFS SLC-40 had the lowest number of successful launches (3 successful launches or 12.5%)

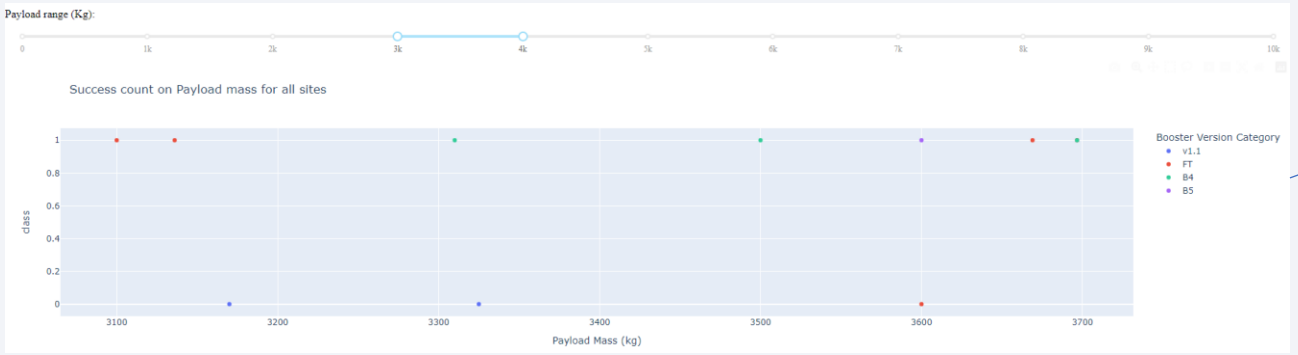
Success/Failure ratio for launches for site with highest ratio (KSC LC-39A)



- In addition to Launch site KSC LC-39A having the highest number of successful launches, it also had the highest launch success ratio at 76.9% (10 out of 13)

Note: Launch site CCAFS SLC-40 (not shown in pic) had the lowest launch success ratio at 42.9% (3 out of 7)

Correlation between Payload Mass and Mission Outcome for all sites considering Booster versions



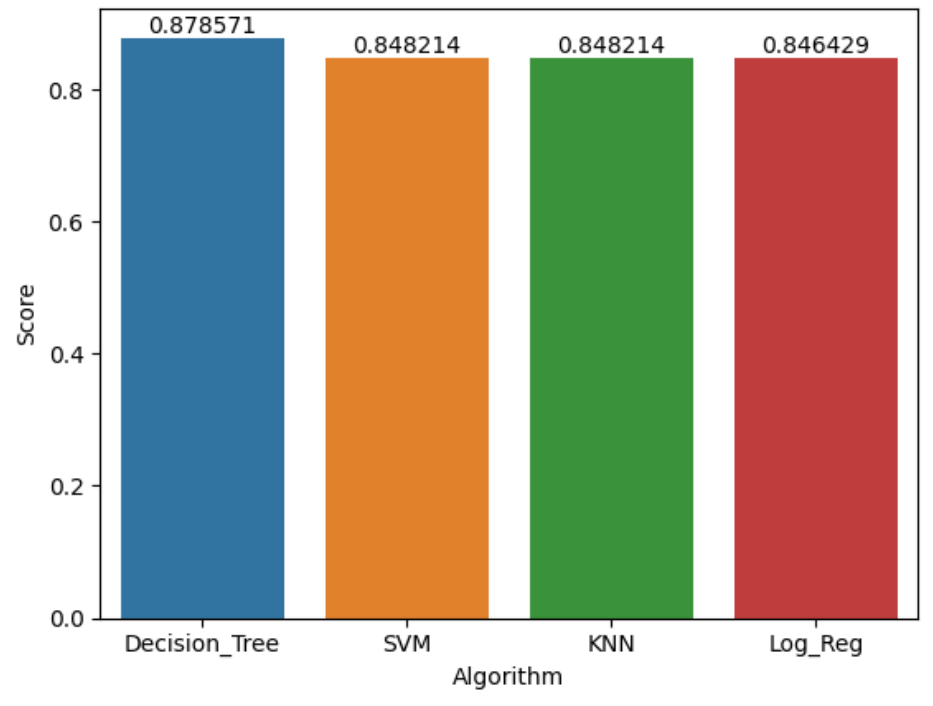
- Relatively low success rate below 2000kg as Payload mass, regardless of booster version
- Low success rate with Payload mass above 5000kg, though only two Booster versions are used

- Highest success rate between 3000 and 4000 kg for Payload mass for most booster versions
- Booster version v1.1 seems to be least successful throughout the range of Payload masses

Section 5

Predictive Analysis (Classification)

Classification Accuracy



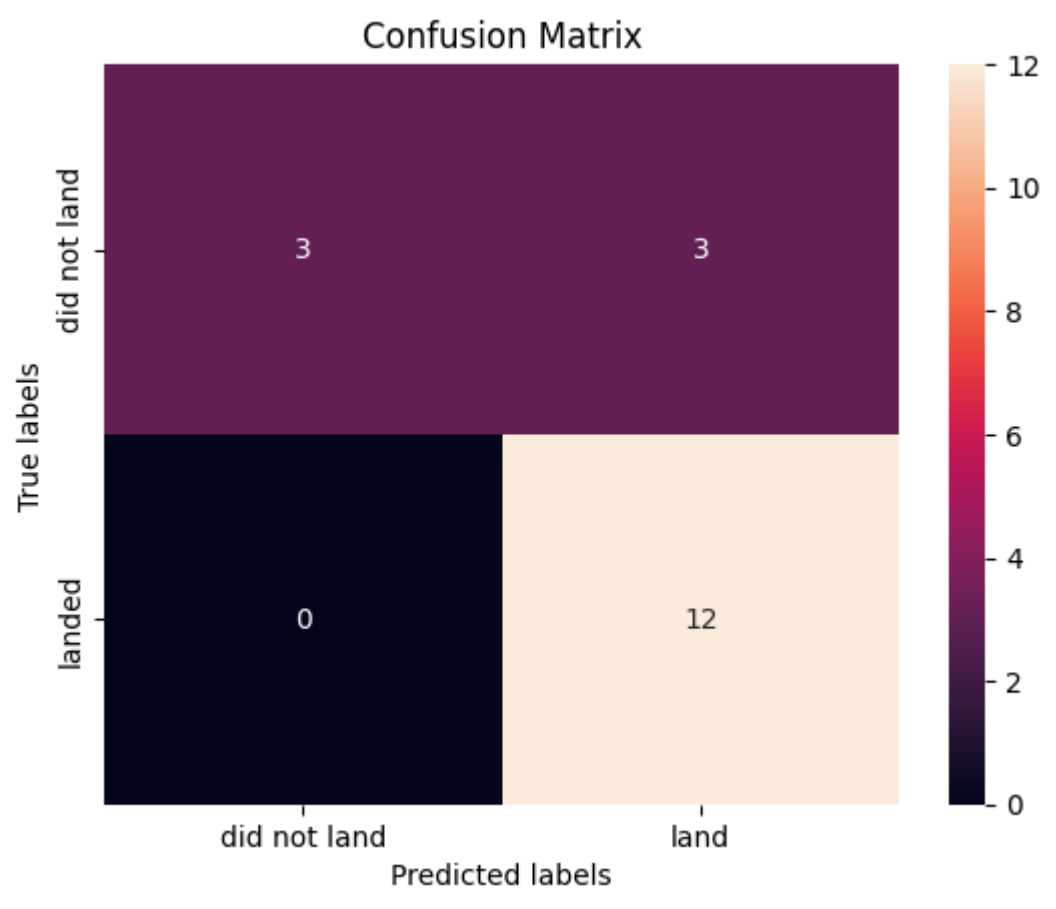
Accuracy based on train data

	Score	Algorithm
0	0.846429	Log_Reg
1	0.848214	SVM
2	0.878571	Decision_Tree
3	0.848214	KNN

- With respect to Train (data) Accuracy score, the **Decision Tree** algorithm performs the best with the highest accuracy score among the four with a value of 0.878571, though they are all quite close.
- However, when looking at the Test (data) Accuracy, all four algorithms return with a value of 0.833333, with no difference between the Jaccard scores or F1 scores among the algorithms either.

	LogReg	SVM	Decision_Tree	K-Nearest_Neigh
Jaccard_Score	0.800000	0.800000	0.800000	0.800000
F1_Score	0.888889	0.888889	0.888889	0.888889
Test Accuracy	0.833333	0.833333	0.833333	0.833333
Train Accuracy	0.846429	0.848214	0.878571	0.848214

Confusion Matrix



- Confusion matrix is the same for all the models evaluated (LogReg, SVM, Decision_Tree and KNN)
- Only 18 data points were used for the predictions, with 12 True positives, 3 True negatives, but there were also 3 False positives (Predicted label as ***land*** but True label is ***did not land***)
- For all the algorithms the prediction gave an accuracy of 83.3% on the test data ((TP+TN)/Total)
- The Precision for the ***landed*** label is 80% (TP/(TP+FP)), while the Recall for the same variable is 100% (TP/(TP+FN))

	LogReg	SVM	Decision_Tree	K-Nearest_Neigh
Jaccard_Score	0.800000	0.800000	0.800000	0.800000
F1_Score	0.888889	0.888889	0.888889	0.888889
Test Accuracy	0.833333	0.833333	0.833333	0.833333
Train Accuracy	0.846429	0.848214	0.878571	0.848214

Conclusions

- With more flights (indicated by increasing flight number), the rate of success has increased, as seen in the yearly trend line for average success rate. This seems to show lessons from previous missions are incorporated into subsequent launches to lead to more positive results.
- There seems to be a negative correlation with low payload masses (<2000 kg) and success rate (3/11). A more definitive conclusion can be drawn with ***additional data***.
- Launch Site 'KSC LC-39A' has the highest launch success rate while Launch Site 'CCAFS SLC- 40' has the lowest launch success rate. However the ***sample size is quite low*** (13 launches and 7 launches respectively) and so no definitive conclusion should be drawn from fact.
- Several Orbit types (ES-L1, GEO, HEO, and SSO) have high launch success rates while orbit GTO has the lowest However, when looked at in more recent times (flight number >50), there is a significantly higher success rate and it does not seem correlated to any orbit type.
- Launch sites are located strategically away from the cities, likely to prevent catastrophic fallout in the event of an accident, but they are closer to coastlines, railroads, and highways, which is likely deliberate as these can serve as transportation means to and from the launch sites.
- The best performing Machine Learning Classfication Model is the Decision Tree with an accuracy of about 87.9% on the training data. However, when the models were scored on the test data, the accuracy score was ~83.3% for all models. Given the proximity in values for all the models, ideally ***additional data*** should be obtained for further training and testing before deployment into production.

Appendix

- Link to spacex_web_scraped.csv, generated after webscraping exercise
- Link to support file for Plotly Dash dashboard, spacex_launch_dash.csv
- Link to spacex_launch_geo.csv file, from interactive map task

Thank you!