DHANUNJAY VENIGANDLA

12-03-22

Outline

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

Executive Summary

❑ Summary of methodologies

- Data Collection
- Data Wrangling
- EDA with visualisation
- Interactive map with Folium
- Building a Dashboard with Plotly
- Predictive analysis (Classification model)

❑ Summary of all results

- Exploratory Data Analysis
- Interactive analytics
- Predictive analysis results

Introduction

- **PROJECT BACKGROUND AND CONTEXT**

SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- **PROJECT BACKGROUND AND CONTEXT**

–

- Determining the inter relationships within the variables and it's effect on the outcome i.e Landing successfully

- Conditions and parameters required to achieve the best results and ensure the best rocket success landing rate.

Section 1

Methodology

Methodology

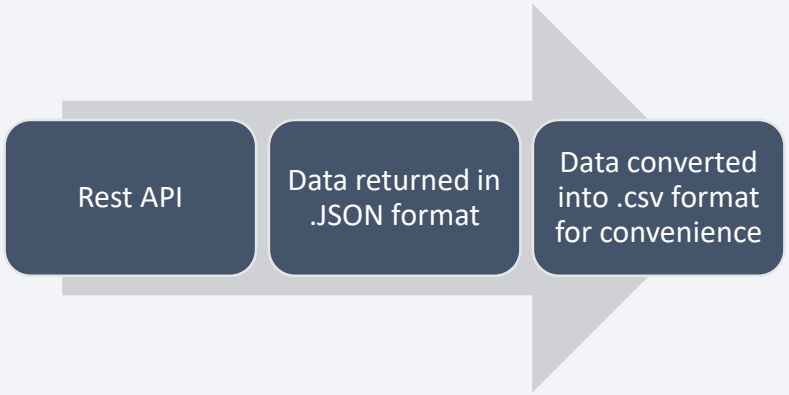
Executive Summary

- Data collection methodology:
  - SpaceX Rest API, (Web Scrapping) from Wikipedia
- Perform data wrangling
  - One Hot Encoding the feature variables and dropping irrelevant columns missing labels
- Perform exploratory data analysis (EDA) using visualization and SQL

  One Hot Encoding the feature variables and dropping irrelevant columns and data with missing labels
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Model Building ,Model Evaluation
  - Model tuning

Data Collection

- **THE FOLLOWING DATASETS WERE COLLECTED BY**
- We worked with SpaceX launch data that is gathered from the SpaceX REST API.
- This API will give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.
- Our goal is to use this data to predict whether SpaceX will attempt to land a rocket or not.
- The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
- Another popular data source for obtaining Falcon 9 Launch data is by web scraping Wikipedia using BeautifulSoup library.



Data Collection – SpaceX API

Github URL for notebook

1 .Getting Response from API

```
In [9]: spacex_url="https://api.spacexdata.com/v4/launches/past"
        response = requests.get(spacex_url)
```

2. Converting Response to a .json file

```
In [13]: # Use json_normalize meethod to convert the json result into a da
         data = pd.json_normalize(response.json())
```

```
In [20]: # Call getLaunchSite
         getLaunchSite(data)

         # Call getPayloadData
         getPayloadData(data)

         # Call getCoreData
         getCoreData(data)
```

3. Apply custom functions to clean data

```
In [23]: 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}
```

4. Assign list to dictionary then data frame.

```
In [35]: # Hint data['BoosterVersion']!='Falcon 1'
         data_falcon9=data.loc[data['BoosterVersion'] == 'Falcon 9']
         data_falcon9
```

Now that we have removed some values we should reset the FlgihtNumber column

```
In [36]: data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
         data_falcon9
```

5. Filter dataframe and export to flat file (.csv)

```
In [41]: data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

Github URL for notebook

1 .Getting Response from HTML

```
In [27]: # use requests.get() method with the provided static_url
         response = requests.get(static_url).text
         # assign the response to a object
```

2. Creating BeautifulSoup Object

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

3. Finding tables

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables=soup.find_all('table')
```

4. Getting column names

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
tryst=soup.find_all('th')

# Iterate each th element and apply the provided extract_column_from_header() to get a column name
for i in range(len(tryst)):
    try:
        name=extract_column_from_header(tryst[i])
        if name is not None and len(name)>0:
            column_names.append(name)
    except:
        pass
```

5. Creation of dictionary

```
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']=[]
```

6. Converting dictionary to dataframe and saving to .csv format

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

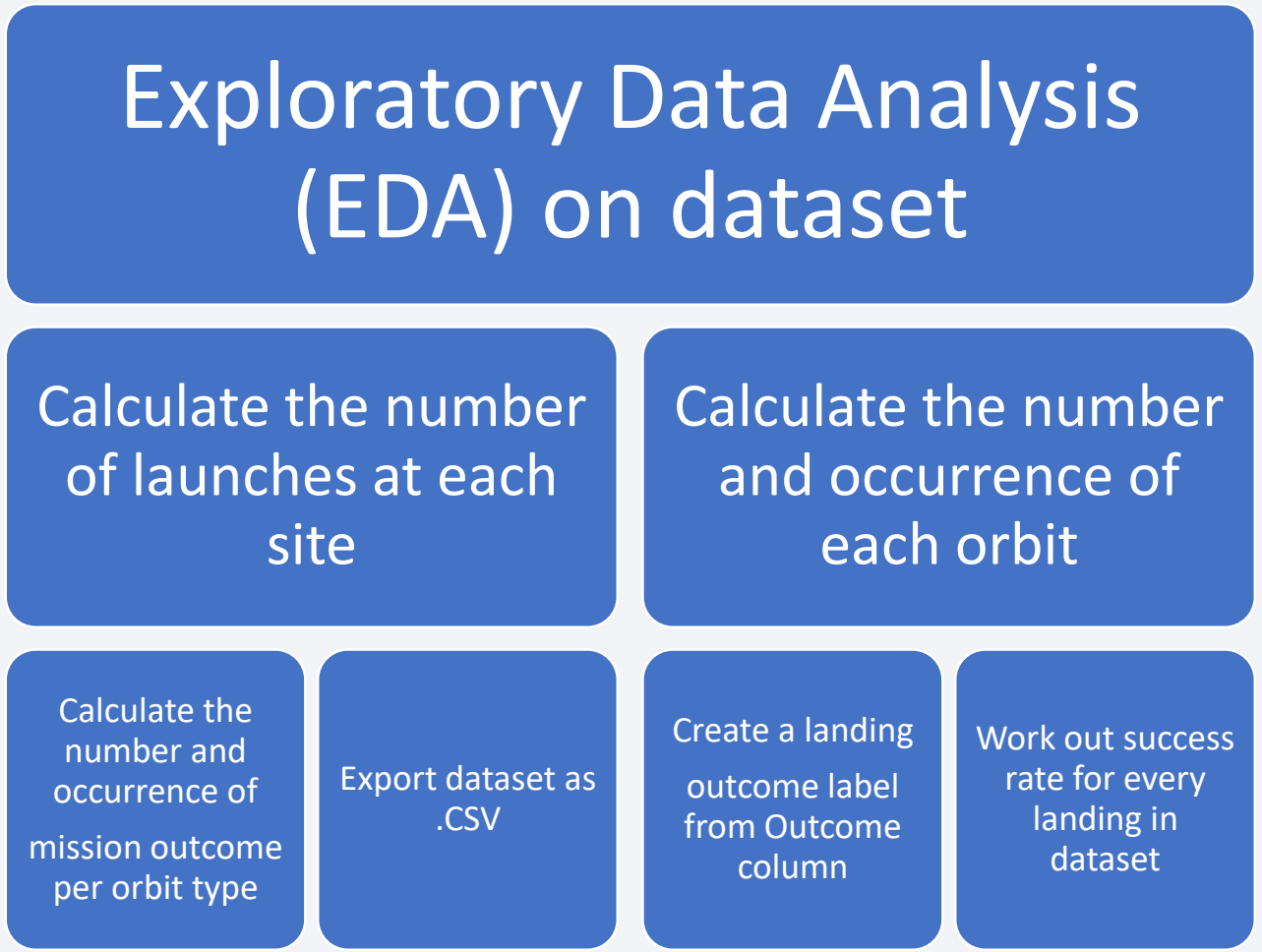
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

Github URL for notebook

- **Introduction**

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad. True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship. We mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.



EDA with Data Visualization

Github URL for notebook

SCATTER GRAPHS:

1. Flight Number VS. Payload
2. Mass Flight Number VS. Launch Site
3. Payload VS. Launch Site
4. Orbit VS. Flight Number Payload VS. Orbit
5. Type Orbit VS. Payload Mass

BAR GRAPH:

1. Mean VS. Orbit

LINE GRAPH:

1. Success Rate VS. Year

A scatter plot is **a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data**.

A bar graph is **a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent**.

A line graph is **a type of chart used to show information that changes over time**. We plot line graphs using several points connected by straight lines.

EDA with SQL

Github URL for notebook

SQL queries to obtain info. From dataset:

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC’
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster versions which have carried the maximum payload mass.
- Listing the records which will display the month names, successful landing outcomes in ground pad ,booster versions, launch site for the months in year 2017
- Ranking the count of successful landing outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

Build an Interactive Map with Folium

Github URL for notebook

- To visualize the Launch Data into an interactive map. We took the Latitude and Longitude Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.
- We assigned the dataframe launch outcomes(failures, successes) to classes 0 and 1 with Green and Red markers on the map in a MarkerCluster()
- Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. Lines are drawn on the map to measure distance to landmarks

Example of some trends in which the Launch Site is situated in.

- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Build a Dashboard with Plotly Dash

GRAPHS:

Pie Chart showing the total launches by a certain site/all sites

- display relative proportions of multiple classes of data.
- size of the circle can be made proportional to the total quantity it represents.

Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions –

- It shows the relationship between two variables.
- - It is the best method to show you a non-linear pattern. - The range of data flow, i.e. maximum and minimum value, can be determined.
- - Observation and reading are straightforward.

Predictive Analysis (Classification)

Github URL for notebook

BUILDING MODEL

• Load our dataset into NumPy and Pandas
• Transform Data
Split our data into training and test data sets
• Pick an estimator to use
Set our parameters and algorithms to GridSearchCV
• Fit our datasets into the GridSearchCV objects and train our dataset

EVALUATING MODEL

• Check accuracy for each model
Get tuned hyperparameters for each type of algorithms
• Plot Confusion Matrix

IMPROVING MODEL

• • Feature Engineering
• Algorithm Tuning

BEST CLASSIFICATION MODEL

• • The model with the best accuracy score wins the best performing model

• There is a dictionary of algorithms with scores at the bottom of the notebook

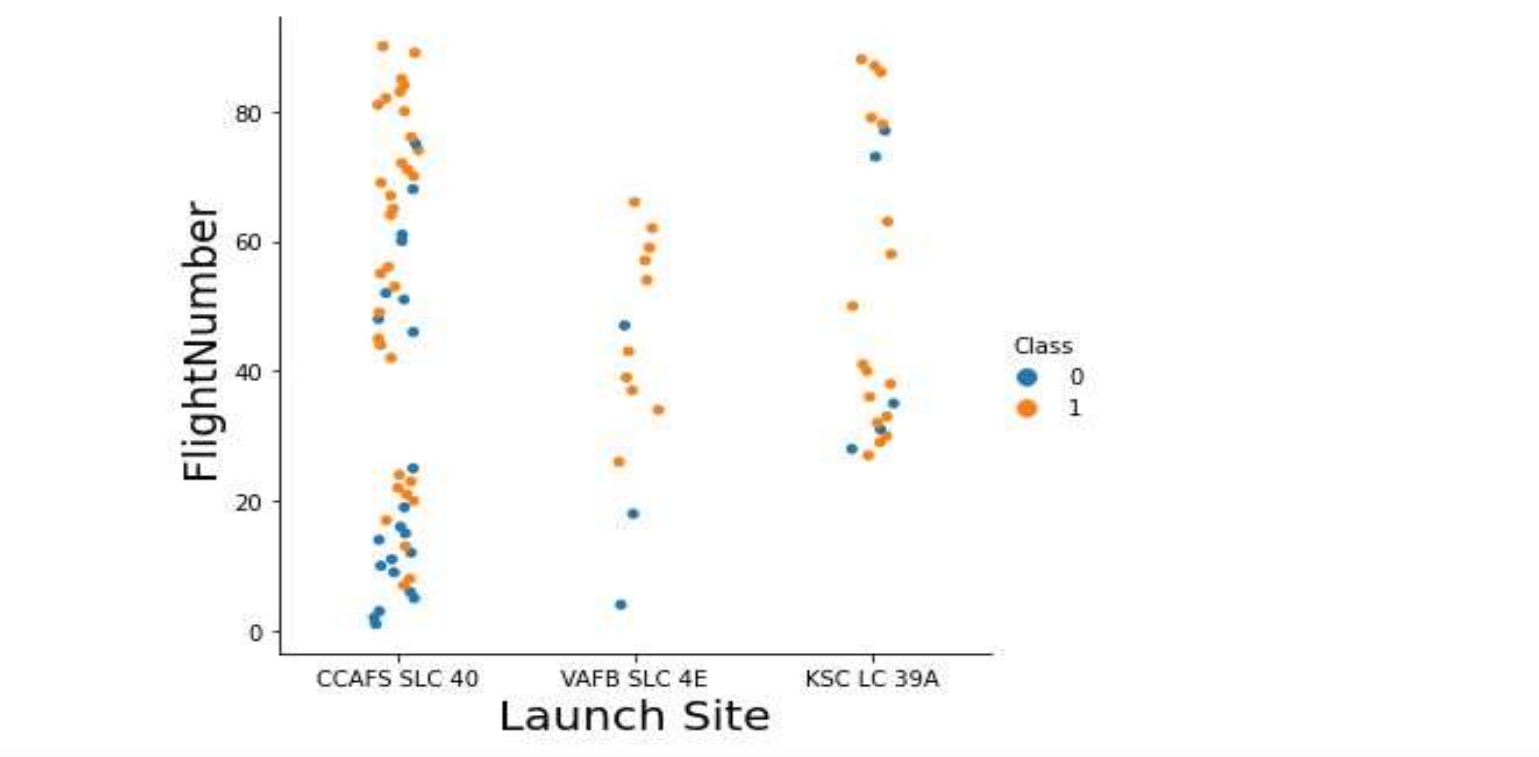
Results

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

Section 2

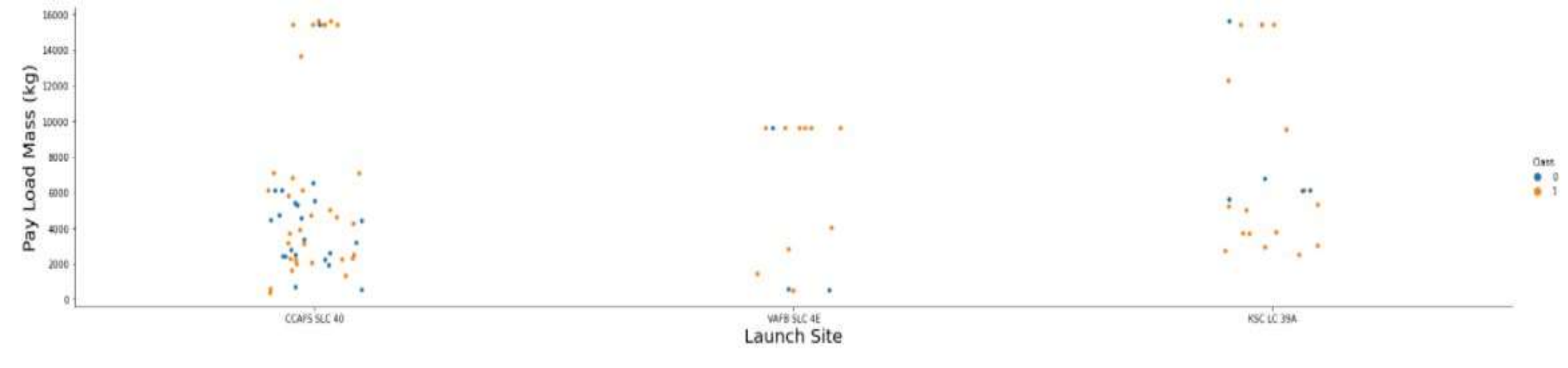
Insights drawn from EDA

Flight Number vs. Launch Site



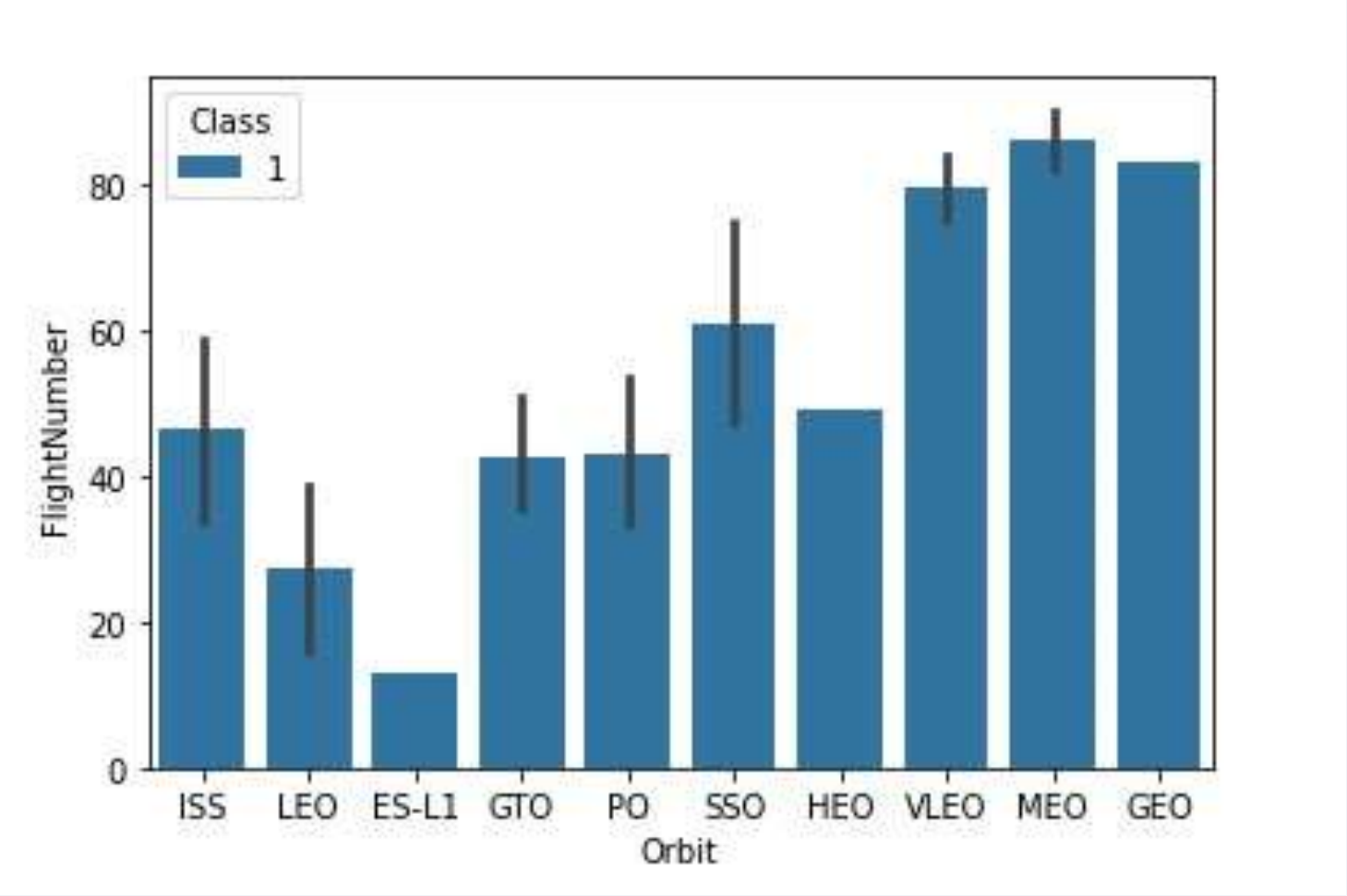
Greater the amount of flights launched from a launch site, greater is the success rate

Payload vs. Launch Site



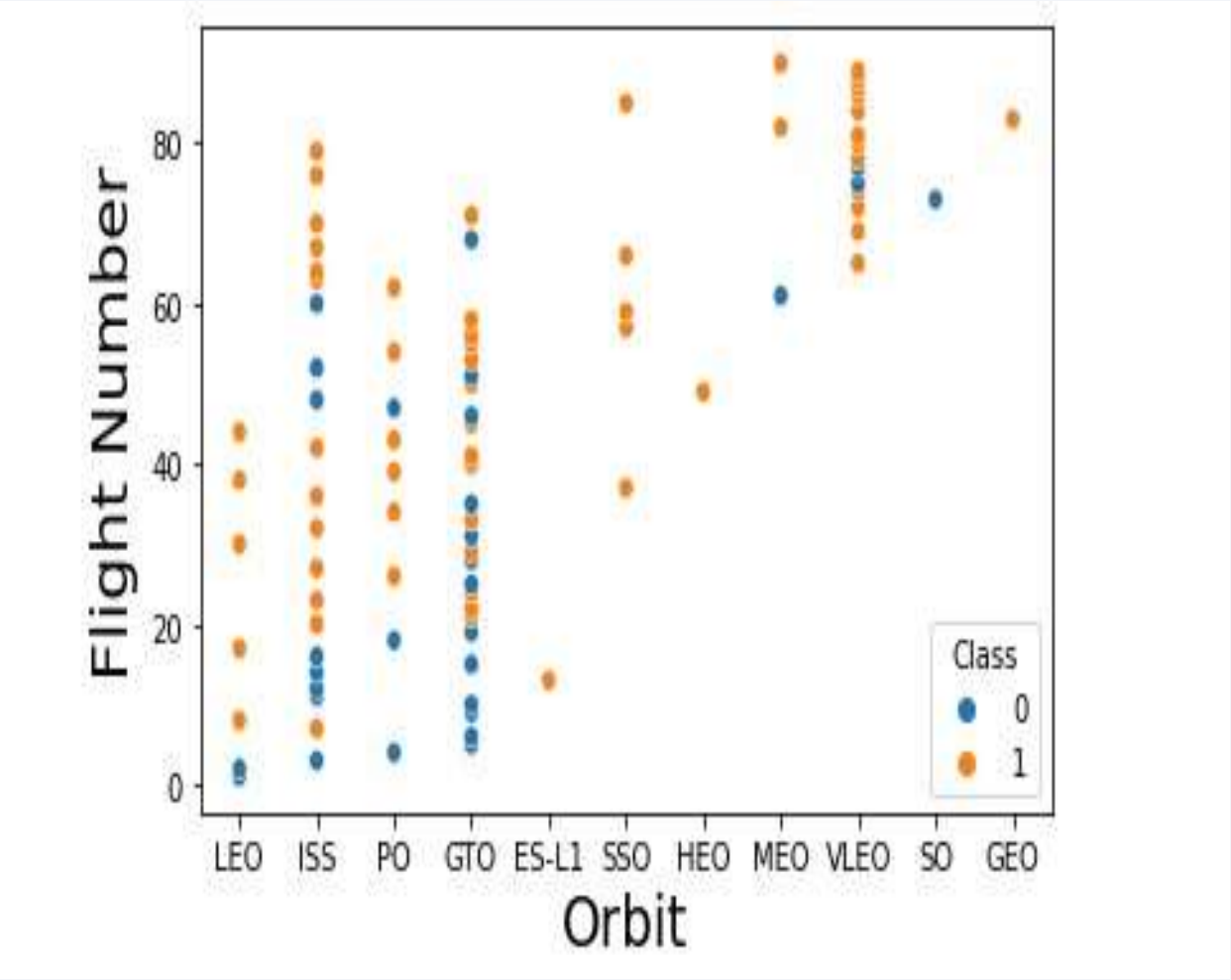
The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependant on Pay Load Mass for a success launch.

Success Rate vs. Orbit Type



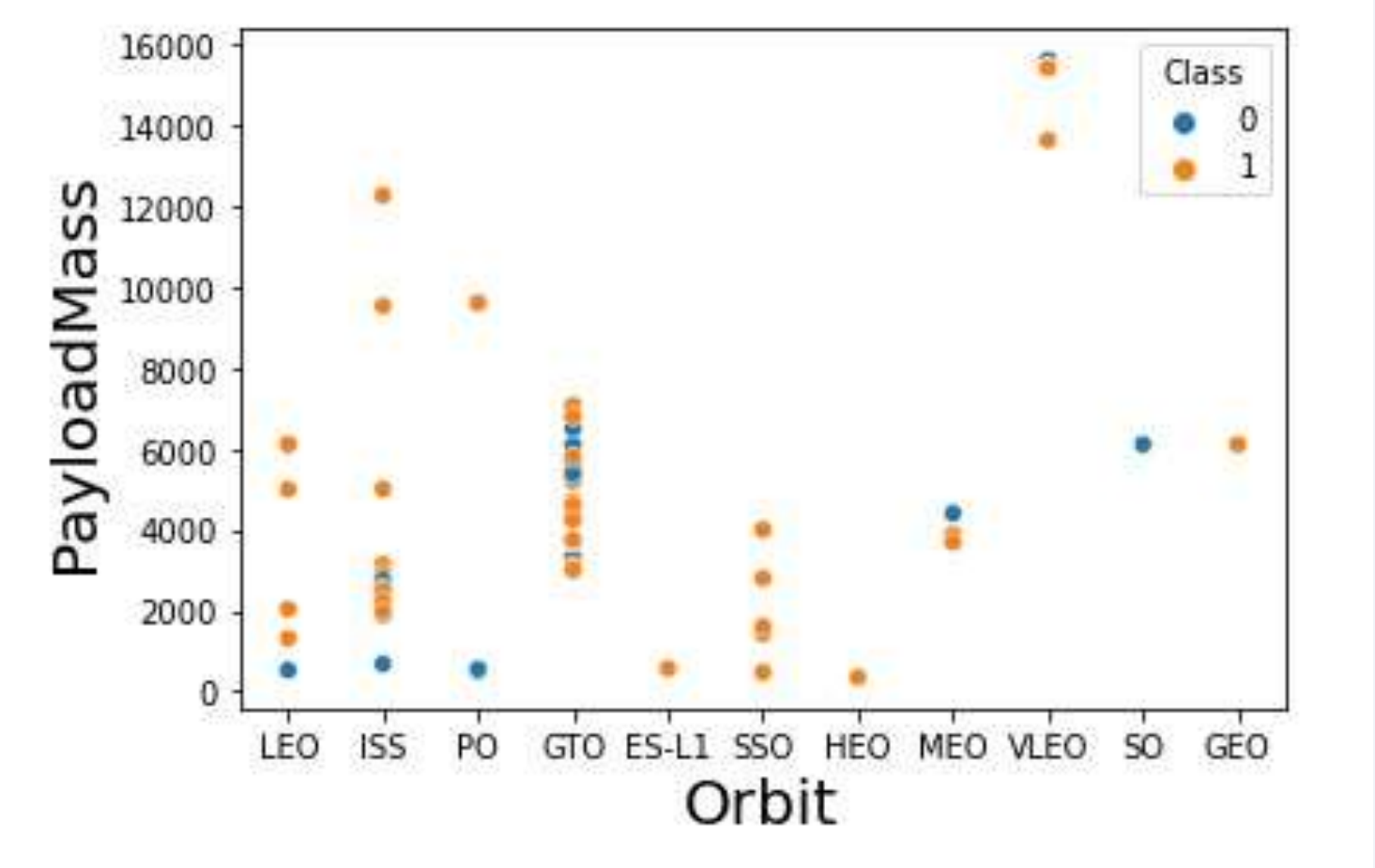
Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

Flight Number vs. Orbit Type



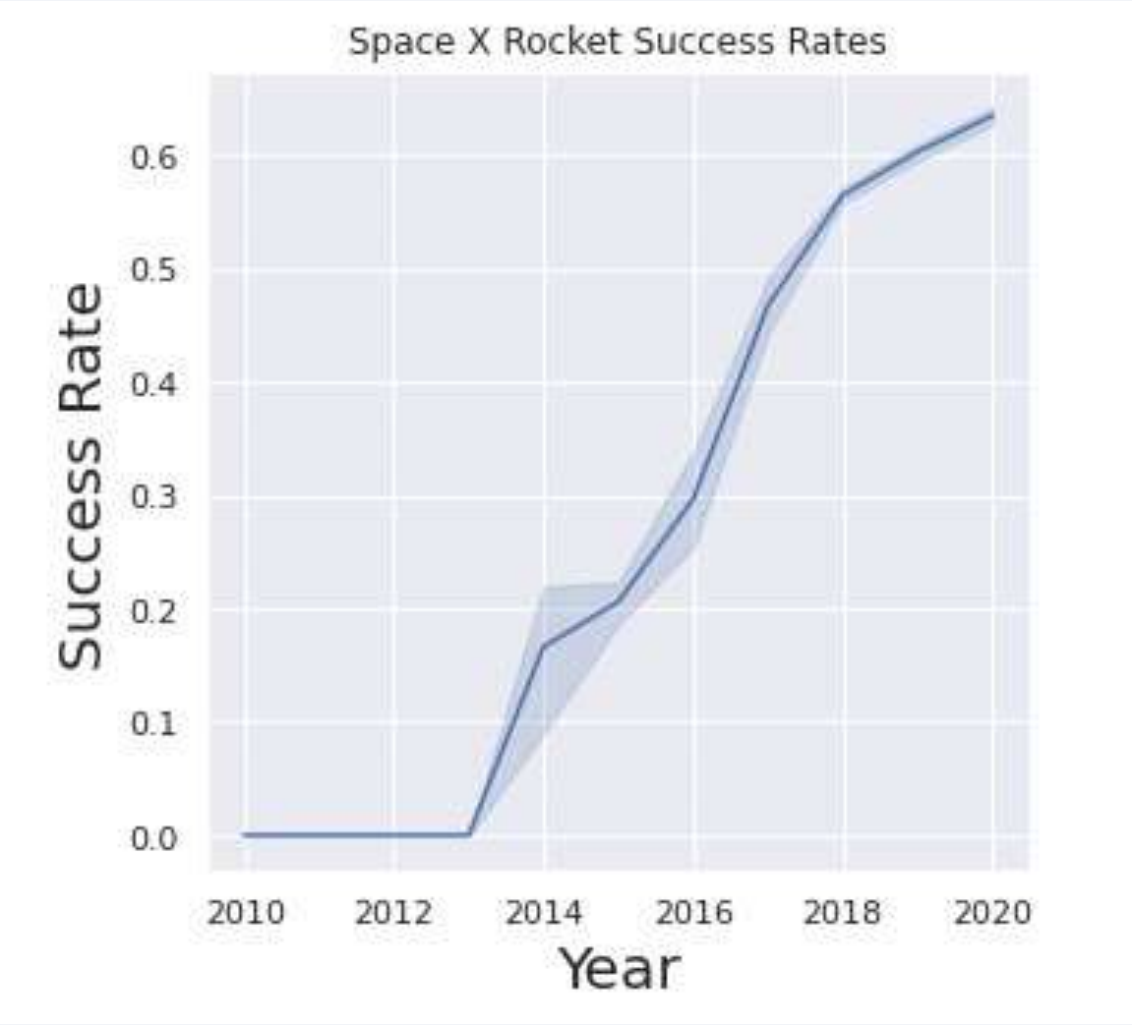
As the no. of ships increase, the success rate of LEO increases where there is no significant relationship for GTO and VLEO

Payload vs. Orbit Type



Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

Launch Success Yearly Trend



It has been an increasing trend from 2013 to 2020. A positive sign indicating a strong improvement.

All Launch Site Names

```
In [13]: %%sql
         SELECT DISTINCT("LAUNCH_SITE") FROM SPACEXDATA;
          * ibm_db_sa://fbh92374:***@2d46b6b4-cbf6-40eb-bbce-6251e
         Done.
```

Out[13]:

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

Using the word DISTINCT in the query means that it will only show Unique values in the Launch_Site column from SPACEXDATA

Launch Site Names Begin with 'CCA'

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

```
 * ibm_db_sa://fbh92374:***@2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io90l08kqb1od8lcg.data
Done.
```

DATE	Time (UTC)	booster_version	launch_site	payload	payload_
0001-01-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	
0001-01-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	
0001-01-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	
0001-01-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	
0001-01-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	

Using the word LIMIT 5 in the query means that it will only show 5 records from SPACEXDATA and LIKE keyword has a wild card with the words ‘CCA%’ the percentage in the end suggests that the LaunchSite name must start with CCA.

Total Payload Mass

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS "total mass" FROM SPACEXDATA WHERE CUSTOMER = 'NASA (CRS)'
```

```
 * ibm_db_sa://fbh92374:***@2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io90l08kqb1od8lcg.data
Done.
```

total mass
45596

Using the function SUM summates the total in the column PAYLOAD_MASS_KG_
The WHERE clause filters the dataset to only perform calculations on Customer NASA (CRS)

Average Payload Mass by F9 v1.1

```
: %%sql
  SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXDATA WHERE BOOSTER_VERSION = 'F9 v1.1'

   * ibm_db_sa://fbh92374:***@2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io90l08kc
  Done.

17]:  1
      2928
```

Using the function AVG works out the average in the column PAYLOAD_MASS_KG_
The WHERE clause filters the dataset to only perform calculations on Booster version F9 v1.1

First Successful Ground Landing Date

```
: %%sql
  SELECT MIN(DATE) FROM SPACEXDATA WHERE LANDING__OUTCOME LIKE 'Success%';

   * ibm_db_sa://fbh92374:***@2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io
  Done.

18]:        1
  0001-01-01
```

Using the function MIN works out the minimum date in the column Date
The WHERE clause filters the dataset to only perform calculations on Landing_Outcome Success (drone ship)

Successful Drone Ship Landing with Payload between 4000 and 6000

```
: %%sql
SELECT BOOSTER_VERSION FROM SPACEXDATA WHERE (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000) AND (LANDING__OUTCOME = 'Success (drone ship)');
```

```
 * ibm_db_sa://fbh92374:***@2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32328/bludb
Done.
```

19]:

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

Selecting only Booster_Version
The WHERE clause filters the dataset to Landing_Outcome = Success (drone ship)
The BETWEEN AND clause specifies additional filter conditions
PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000

Total Number of Successful and Failure Mission Outcomes

```
%%sql
SELECT COUNT(*) FROM SPACEXDATA WHERE (LANDING__OUTCOME LIKE 'Succ%') OR (LANDING__OUTCOME LIKE 'fail%');
```

```
 * ibm_db_sa://fbh92374:***@2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io90l08kqb1od8lcg.databases.appdomain.cl
Done.
```

]:

1
61

Calculated the total no. of outcomes with help of wild card operator and the OR operator for term succ% (starting with) or fail% (starting with)

Boosters Carried Maximum Payload

```
%%sql
SELECT DISTINCT(BOOSTER_VERSION),PAYLOAD_MASS__KG_ FROM SPACEXDATA WHERE (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXDATA)
```

```
 * ibm_db_sa://fbh92374:***@2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32328
Done.
```

8]:

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1048.5	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1049.7	15600
F9 B5 B1051.3	15600
F9 B5 B1051.4	15600
F9 B5 B1051.6	15600
F9 B5 B1056.4	15600
F9 B5 B1058.3	15600

DISTINCT for obtaining only unique elements; we used the sub-query in WHERE clause to obtain the max. Payload with help of MAX function from SPACEXDATA

Section 3

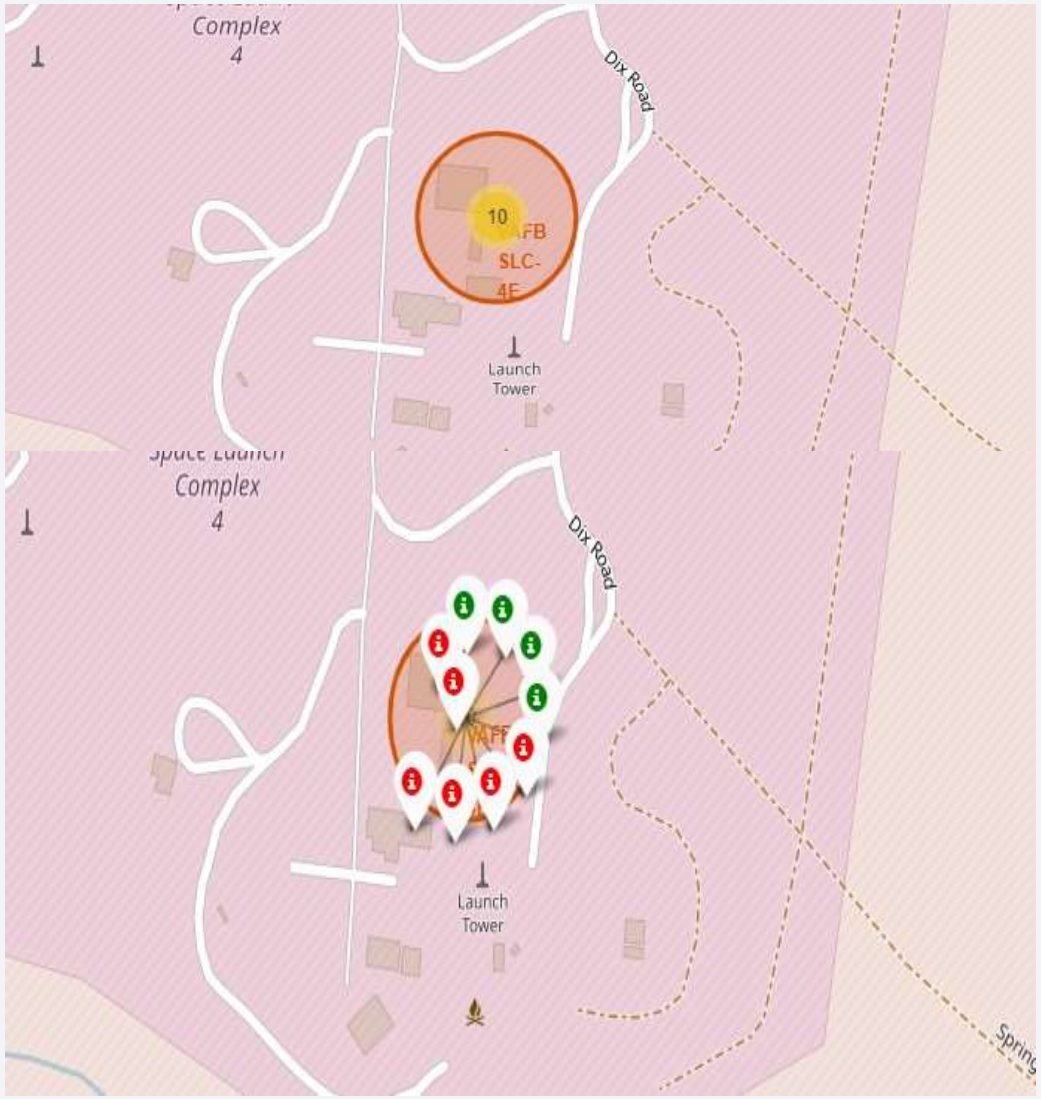
Launch Sites Proximities Analysis

Global Markers of Launch Sites

Launch sites located at Calfiornia and Florida. Both coastal areas

Color Labelled Markers

California

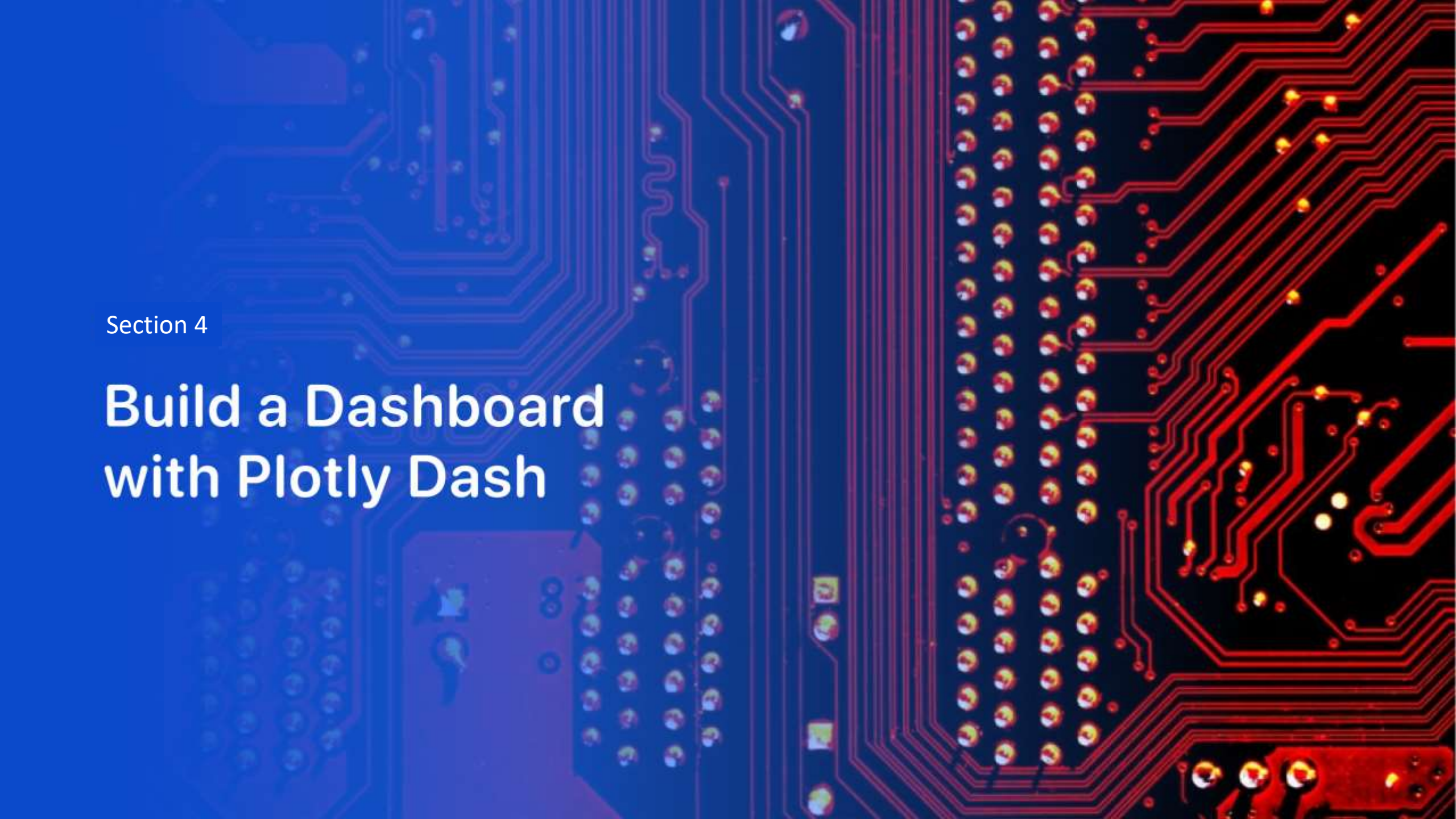


Florida

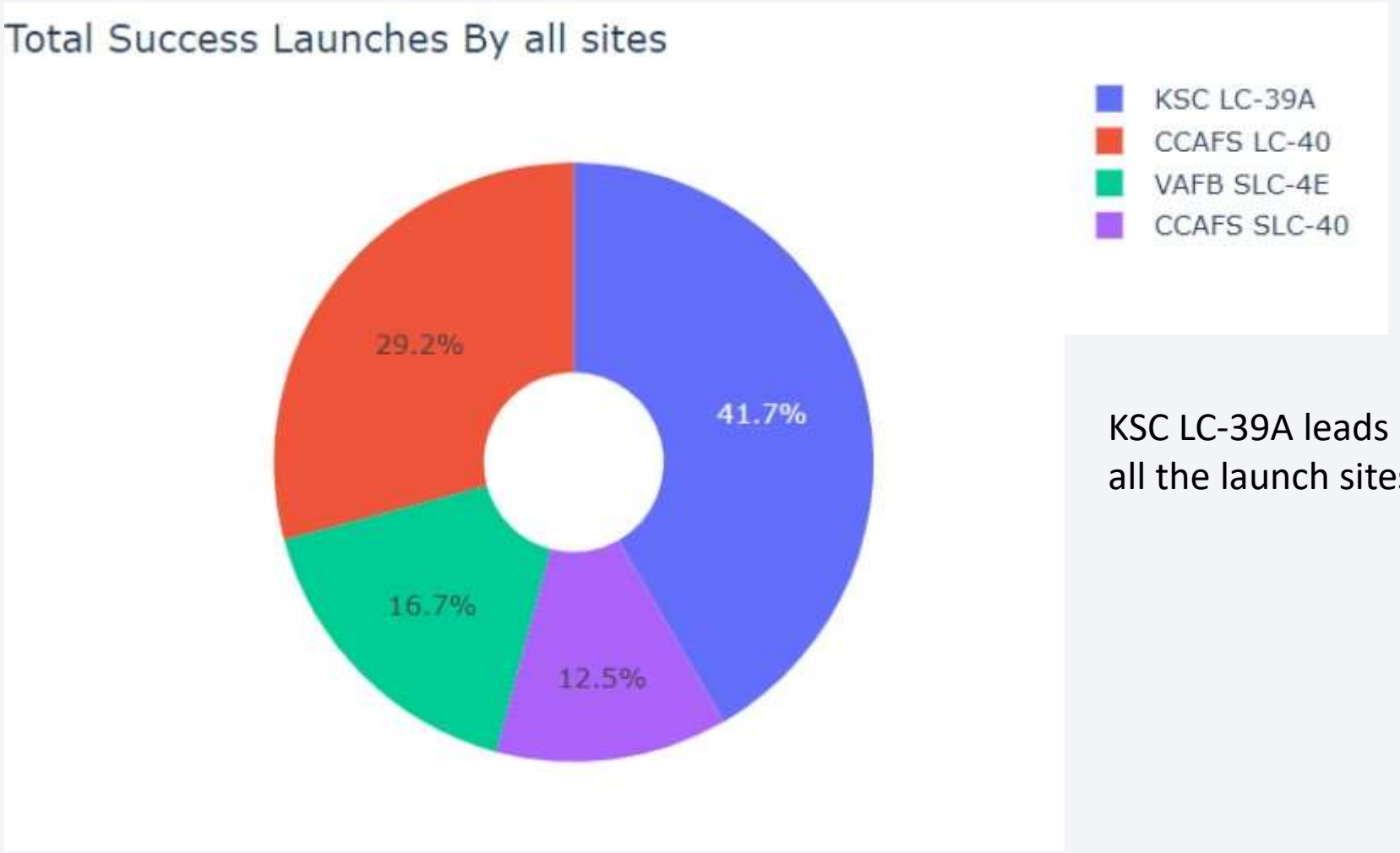
Green Marker shows successful Launches and Red Marker shows Failures

Launch Sites distance to landmarks to find trends using CCAFS-SLC-40 as a reference

Are launch sites in close proximity to railways? No
Are launch sites in close proximity to highways? No
Are launch sites in close proximity to coastline? Yes
Do launch sites keep certain distance away from cities? Yes

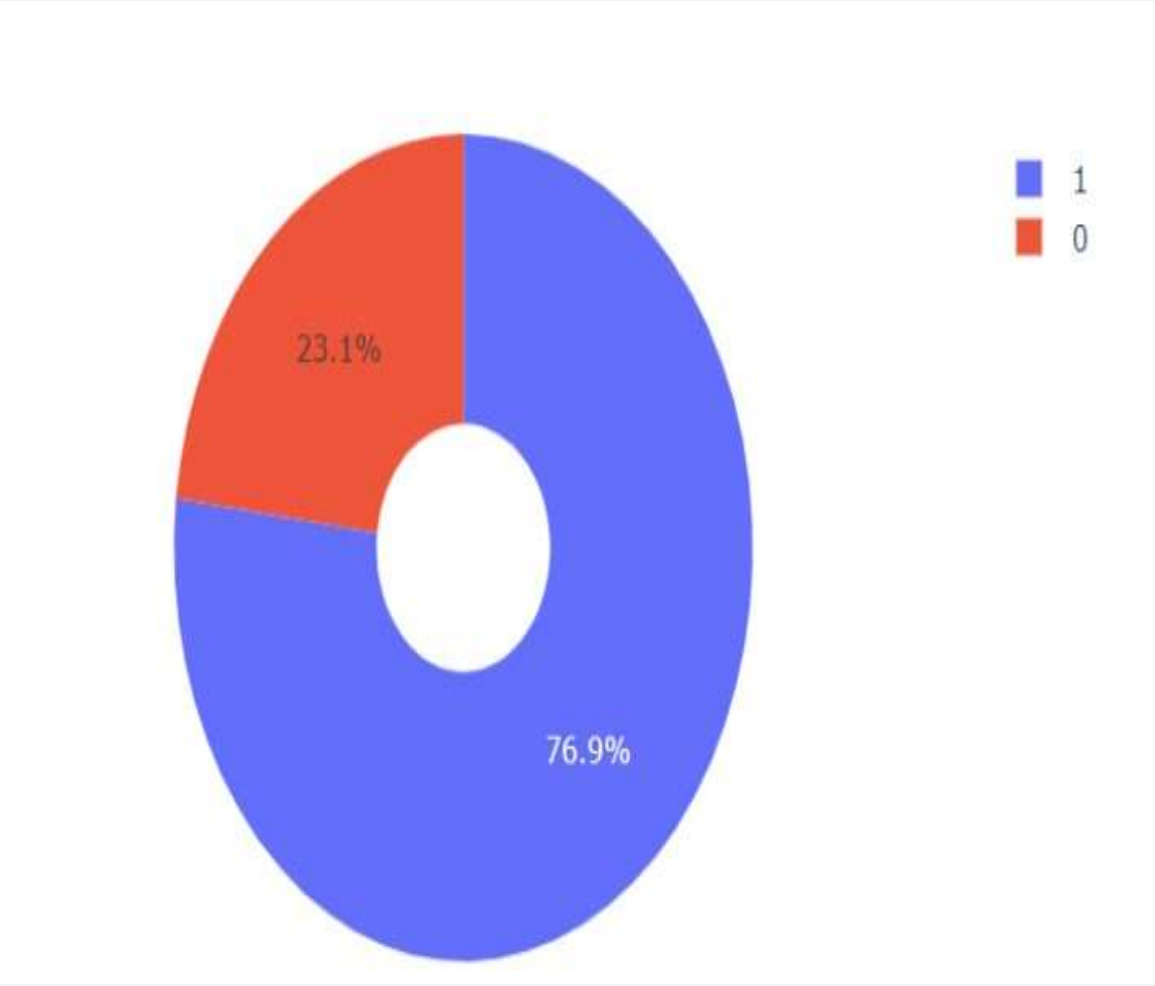

Section 4
Build a Dashboard with Plotly Dash

DASHBOARD –Success percentage achieved by each launch site



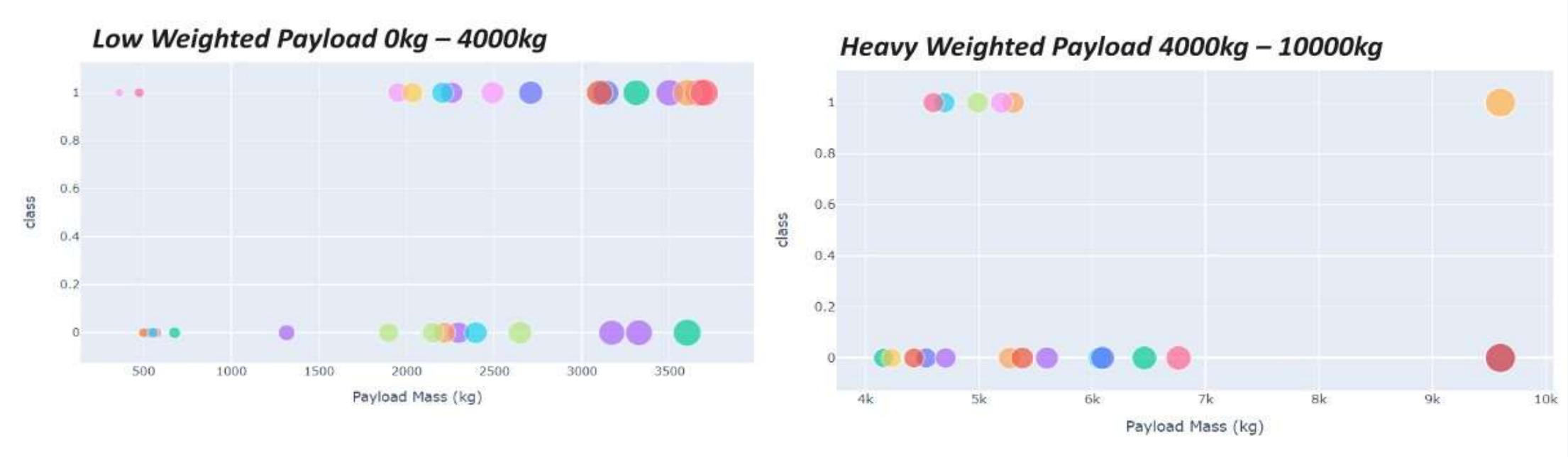
KSC LC-39A leads in the success rate among all the launch sites

Pie chart for the launch site(KSC LC-39A) with highest launch success ratio



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider



Lower payloads have a higher success rate than higher payloads which have lower success rate

Section 5

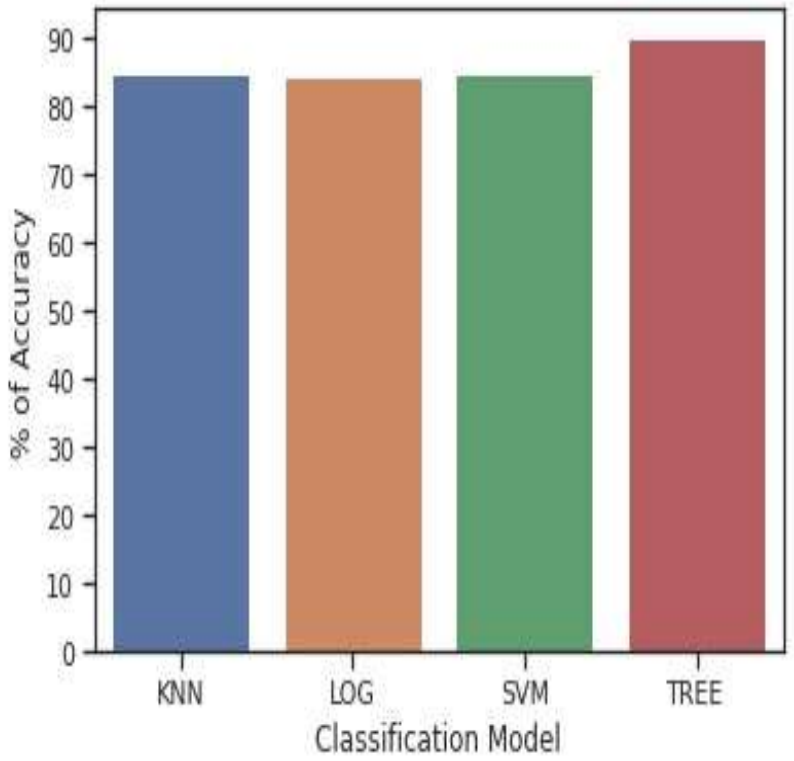
Predictive Analysis (Classification)

Classification Accuracy

	Accuracy
KNN	0.848214
LOG	0.846429
SVM	0.848214
TREE	0.887500

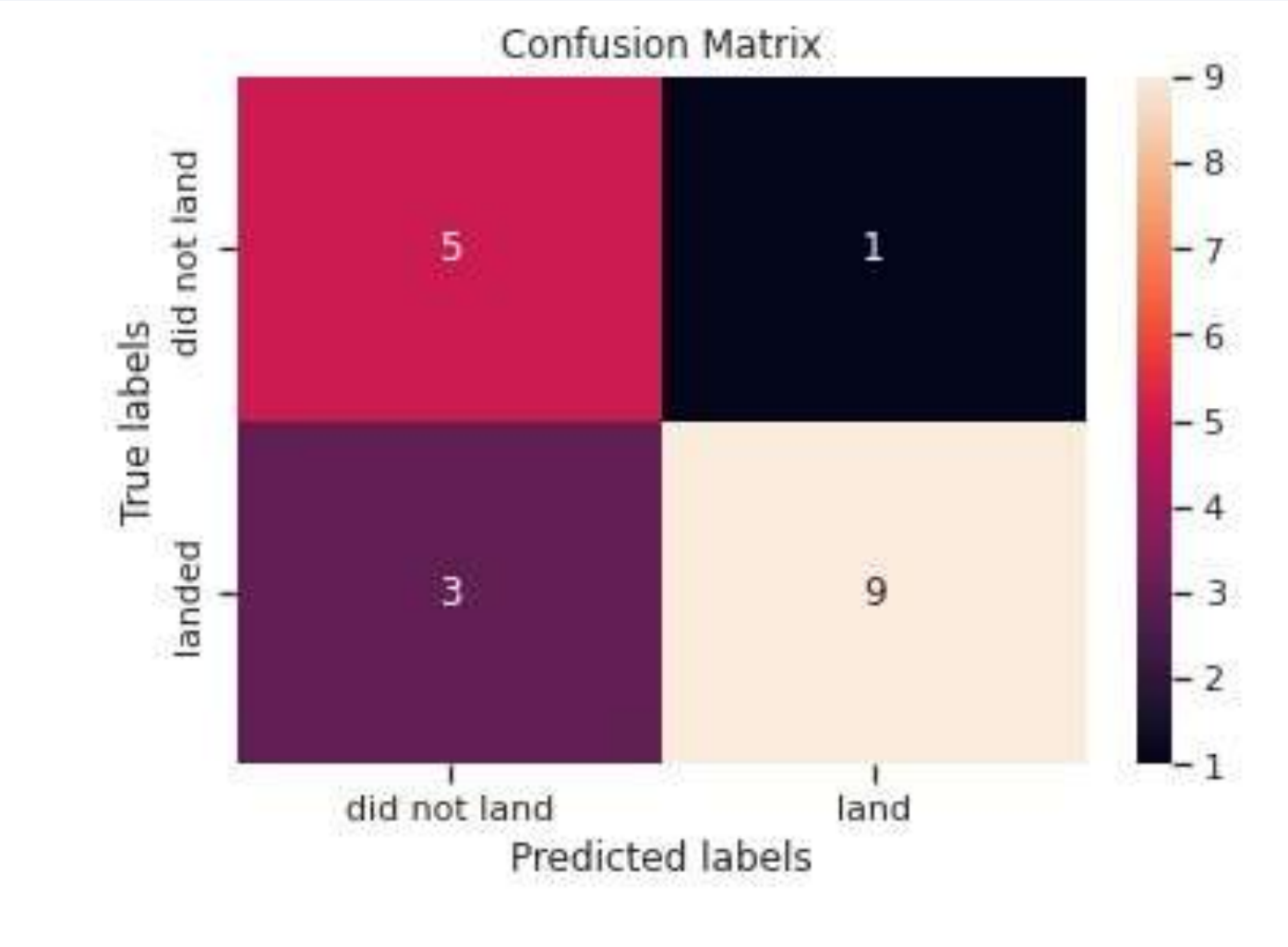
All the algorithms are super close but the "DECISION SCORE" algorithm has the higher accuracy score.

With the help of user defined function, we obtain the best performing model and it's corresponding hyper parameters.



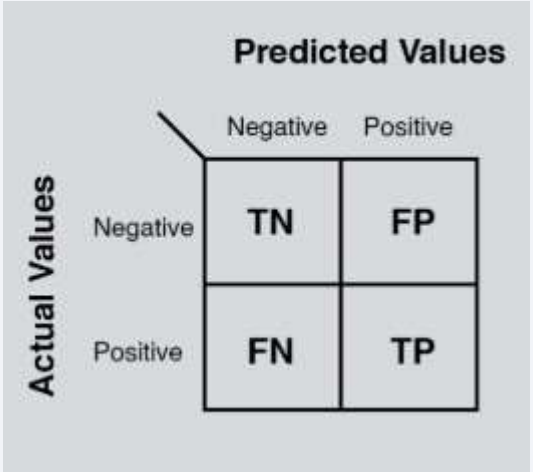

```
Best Algorithm is Tree with a score of 0.8875
Best Params is : {'criterion': 'entropy', 'max_depth': 18, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'rando
m'}
```

Confusion Matrix



The confusion matrix distinguishes True positive, True negatives, False Positive, False negative.

Reference table



Conclusions

- The Tree Classifier Algorithm is the best for Machine Learning for this dataset
- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches
- We can see that KSC LC-39A had the most successful launches from all the sites
- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

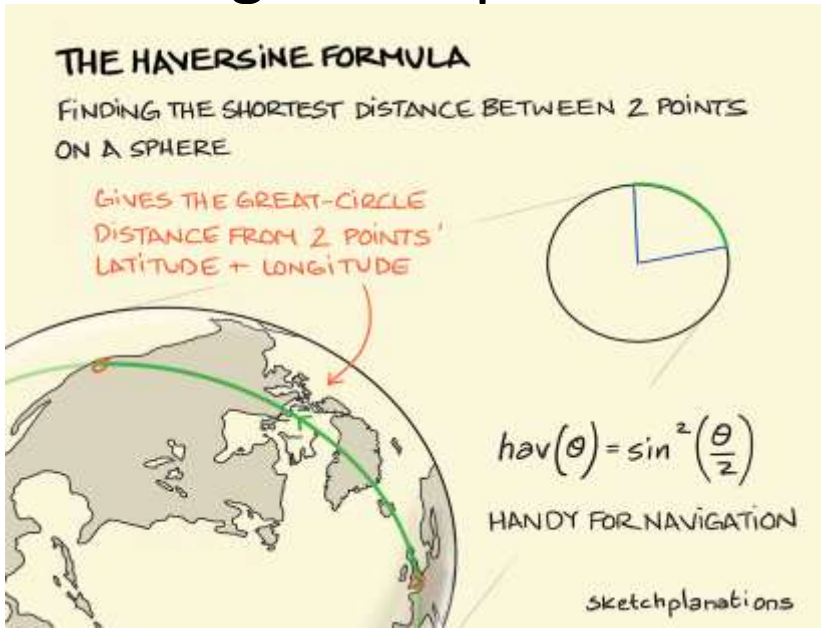
Appendix

- Haversine formula:

- IBM Cloud Db2 server

Haversine formula:

The haversine formula **determines the great-circle distance between two points on a sphere given their longitudes and latitudes**. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines, that relates the sides and angles of spherical triangles.




```python
from math import sin, cos, sqrt, atan2, radians

def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance
```

IBM Cloud Db2 server

In order to access SQL, I have uploaded the SpaceX.csv file into a cloud database. IBM Db2 is a free to use cloud database provided by IBM. We have used this to perform Exploratory data analysis through SQL.

Db2 Database

Export to CSV

DATE	Time (UTC)	BOOSTER_VERSION	LAUNCH_SITE	PAYLOAD	PAYLOAD_MASS__KG_	ORBIT	CUSTOMER	MISSION_OUTCOME
0001-01-02	03:50:00	F9 v1.1 B1014	CCAFS LC-40	ABS-3A Eutelsat 115 West B	4159	GTO	ABS Eutelsat	Success
0001-01-02	07:49:00	F9 B5B1051.1	KSC LC-39A	Crew Dragon Demo-1, SpaceX CRS-17	12055	LEO (ISS)	NASA (CCD)	Success
0001-01-02	20:30:00	F9 B4 B1039.2	CCAFS SLC-40	SpaceX CRS-14	2647	LEO (ISS)	NASA (CRS)	Success
0001-	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170	GTO	SES	Success

```
%sql ibm_db_sa://fbh92374:fqLdqiMXQE08ymjd@2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32328/bludb?security=SSL
```

IBM Db2 login access to Jupyter notebook

Thank you!