IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Denis O'Byrne
2/20/2022

Outline

Executive Summary

Introduction

Methodology

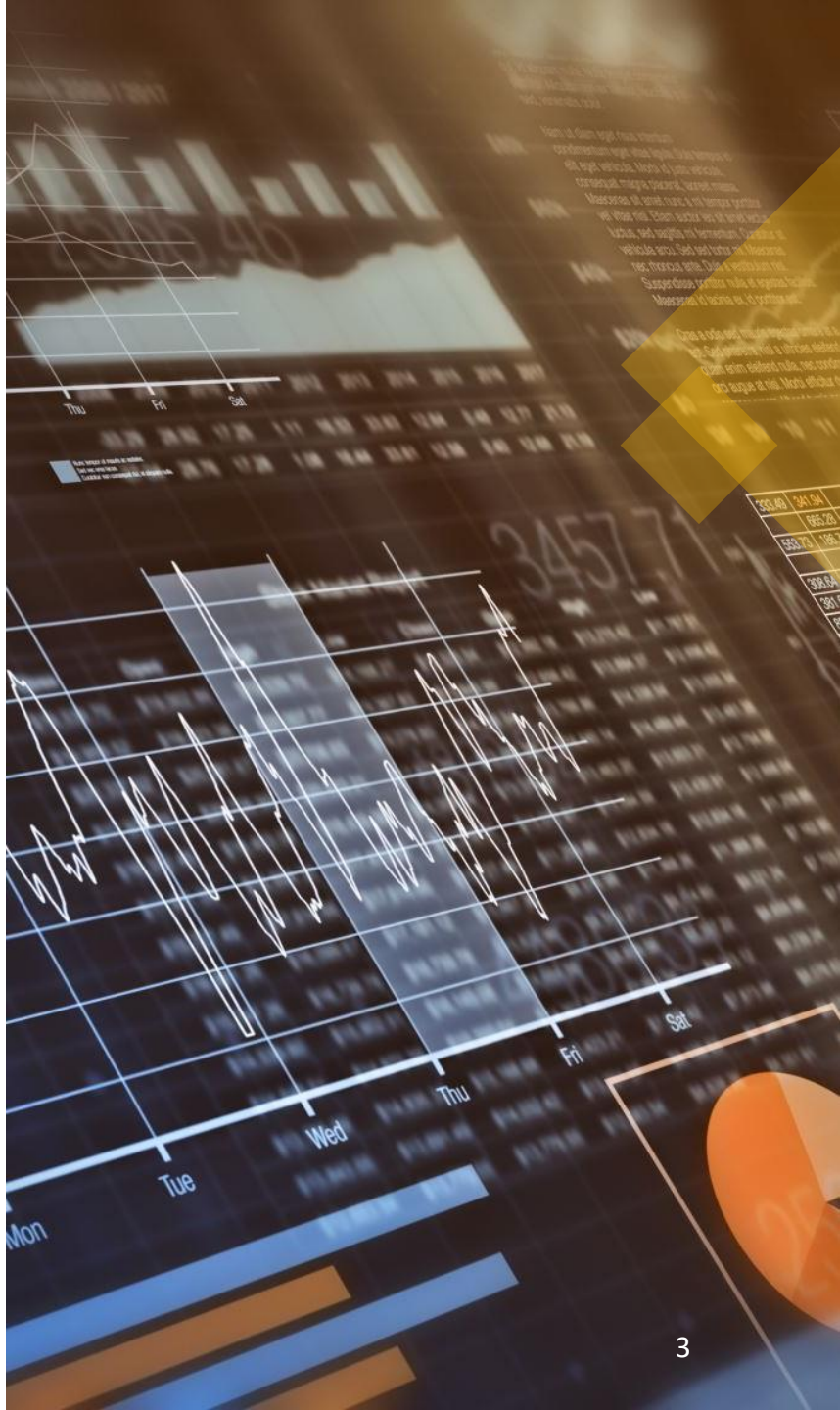
Results

Conclusion

Appendix

Executive Summary

- Summary of methodologies:
  - Data Collection using Webscraping and REST API queries
  - Data Wrangling to Classify Lauches based on Success and transform data into standardized numeric form
  - Exploratory Data Analysis using SQL and Visualization packages for Python
  - Interactive Plotly Web App to visualize payload and success launch data at each Launch Site
  - Exploring Launch Sites using interactive Folium Maps
  - Predictive analysis for classification of Rocket Landing Success
- Summary of all results:
  - Exploratory Data Analysis Results
  - Predictive Analysis Results



Introduction

- 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. Thus it is advantageous to be able to predict whether the Falcon 9 first stage will land successfully for new missions.
- To make valuable predictions we must solve the following:
  - What factors of a mission influence Falcon 9 launch success?
  - What conditions must be met by SpaceX to ensure the highest probability of success for a given mission?

Section 1

Methodology

Methodology

Data collection methodology:

- Requested past launch data from SpaceX's Rest API https://api.spacexdata.com/v4
- Webscraped tabular data on SpaceX launches from

Perform data wrangling

- Dropped data on non Falcon 9 launches
- Used One Hot Encoding to transform categorical variables into factors
- Transformed factors to integers
- Replaced missing numerical data for payload masses with the sample mean
- Classified data as 1 for Successful Landing or 0 for failed landing

Performed exploratory data analysis (EDA) using visualization and SQL:

- Used Scatter Plots and Bar Graphs to visualize relationships between variables
- Used SQL Queries to understand the data collected

Methodology Continued

Perform interactive visual analytics using Folium and Plotly Dash

- Interactive Plotly Web App to visualize payload and success launch data at each Launch Site
- Exploring Launch Sites using interactive Folium Maps

Perform predictive analysis using classification models:

- Built and tuned multiple classification models to predict landing success
- Used Grid Search and Cross Validation to find the best model parameters for each model tested (Logistic, SVM, Decision Tree, and KNN)
- Split Data into testing and training to test model accuracy resilience on Out of Sample Data
- Selected top performing Model on both testing and training set based on accuracy of predictions

Data Collection

- Used SpaceX REST API to gather data on rocket launches:
  - https://api.spacexdata.com/v4
- API provides data on rockets used, launch dates, payload masses, launch success or failure, launch site name and location (latitude and longitude), booster version (note for this experiment we are only interested in Falcon V9 boosters), landing outcome, etc. (47 columns of data for each launch in total)
- Our Goal is to Predict the Landing outcome using the other variables
- Falcon 9 launch data was also collected via Webscraping Wikipedia using BeautifulSoup from the page below:
  - https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

Data Collection Steps – SpaceX API

Github URL to Notebook :
Space X API Data Collection Notebook by Denis O'Byrne

1. Get Response from API

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

response = requests.get(spacex_url)
```

Note for this project a static version of the data is used from the following url:
https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json

2. Convert Response to JSON File then read as Pandas data frame

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(requests.get(static_json_url).json())
```

3. Clean data to extract desired information using custom functions

```
# Call getBoosterVersion
getBoosterVersion(data)

# Call getLaunchSite
getLaunchSite(data)

# Call getPayloadData
getPayloadData(data)

# Call getCoreData
getCoreData(data)
```

4. Assign cleaned data to a new data frame using dictionary of lists

```
# Create a data from launch_dict
data=pd.DataFrame.from_dict(launch_dict,orient='index').transpose()
```

```
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. Filter data for Falcon V9 boosters only then export to csv

```
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

Data Collection Steps – Webscraping Wikipedia

Github URL to Notebook : Webscraping Notebook by Denis O'Byrne

1. Get Response from HTML and create Soup object

```
# assign the response to a object
response = requests.get(static_url)
```

Note for this project a static version of the data is used from the following url:
https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

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

2. Find Tables

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

3. Extract column names

```
column_names = []

ths = first_launch_table.find_all('th')
for i in range(0,len(ths)-1):
    name = extract_column_from_header(ths[i])
    if (name is not None and len(name) > 0):
        column_names.append(name)
```

4. Create Launch Data 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']=[]
```

5. Extract table data to dictionary

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
```

6. Create data frame from dictionary

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

7. Export to CSV

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

Data Wrangling – Training Labels

Plan:

- 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.
- To convert those outcomes into Training Labels with `1` means the booster successfully landed `0` means it was unsuccessful

Steps:

1. Load data

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
```

2. Identify Landing outcomes as good and bad

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

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

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 None ASDS
6 False Ocean
7 False RTLS
```

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

3. Create a list of training labels

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class=[]
for i in range(0,len(df['Outcome'])):
    if df['Outcome'][i] in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

4. Add list to data frame in new column "Class"

```
df['Class']=landing_class
```

Github URL to Notebook : [Data Wrangling Notebook by Denis O'Byrne]

Data Wrangling – Imputing Missing Data

In the earlier SpaceX API notebook we performed data wrangling to impute missing data

Steps:

1. Check for Missing Values
   - Note we found 5 missing values for Payload Mass which should not be missing and nonzero. We will replace theses missing values with the Sample Mean
   - Note we found 26 entries of None for Landing Pad which is informative indicating there was no landing pad to land the rocket on for the mission meaning we should not change this column

```
data_falcon9.isnull().sum()
```

```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       5
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad       26
Block             0
ReusedCount       0
Serial            0
Longitude         0
Latitude          0
dtype: int64
```

2. Calculate Sample Mean For Payload mass

3. Replace nas for Payload mass with sample mean

```
# Calculate the mean value of PayloadMass column
m_plm = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, m_plm)
```

Github URL to Notebook : Space X API Data Collection Notebook by Denis O'Byrne

EDA with Data Visualization

Github URL to Notebook : EDA Data Visualization in Python by Denis O'Byrne

- Visualize Orbits of Rockets and Count Launches in each orbit
  - Larger orbits require more fuel which may impact launch landing success
- Bar Graph to show landing success percentage for given Orbit Launches
  - By grouping our success based on orbit we can see if our intuition and combining this chart with our orbits visualization we can see the relationship between orbit radius and landing success
- Scatter Plots to show the relationships between the following Variables and visualize their correlation:
  a) Flight Number vs. Payload Mass
  b) Flight Number vs. Launch Site
  c) Payload Mass vs Launch Site
  d) Orbit vs Flight Number
  e) Payload Mass vs. Orbit
- Line Graph to show landing success percentage with respect to the year of launch
  - We expect SpaceX Engineers to learn from their mistakes and improve over time but how fast are they improving

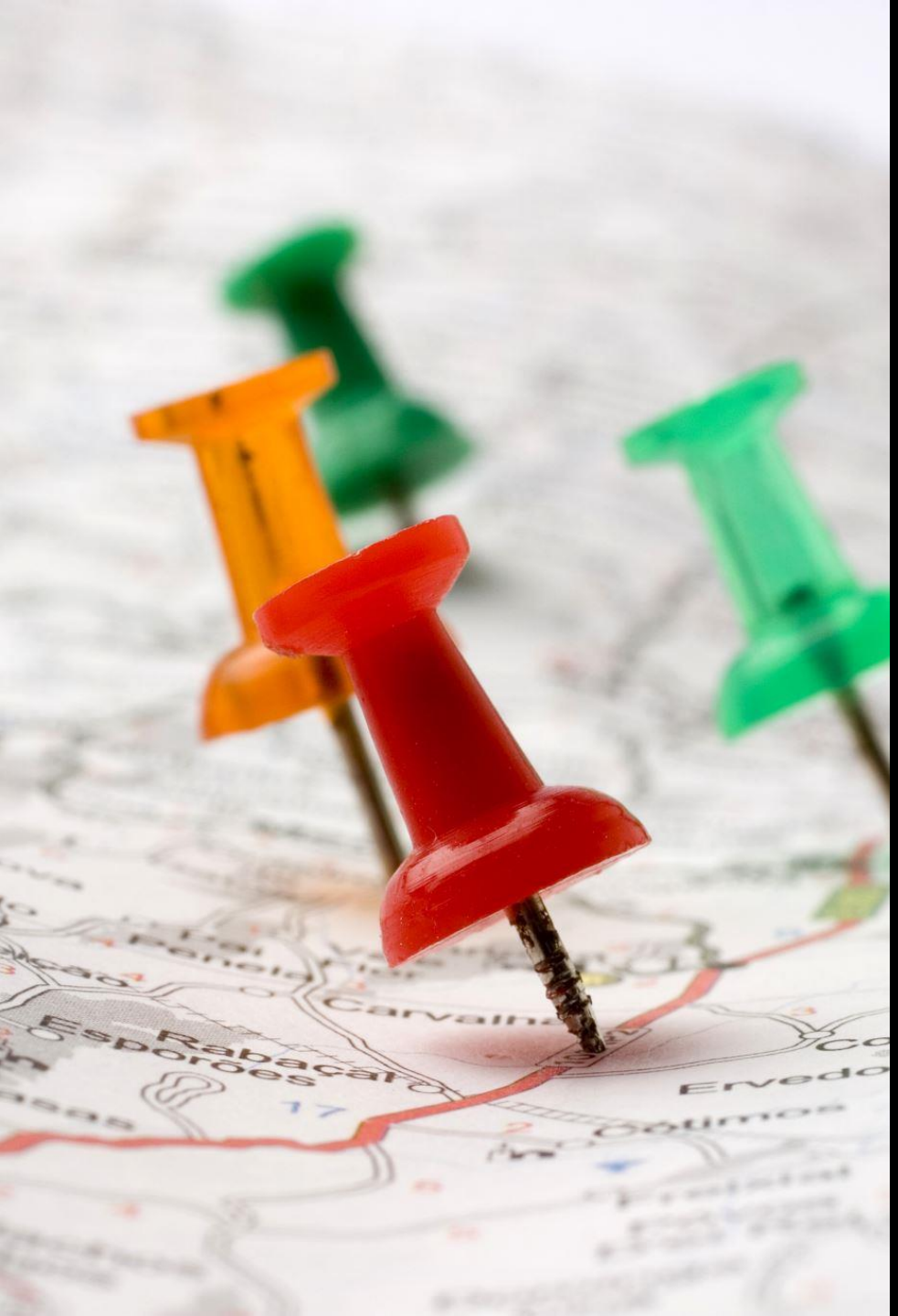
EDA with SQL

Used SQL Queries to an IBM DB2 instance to gain insight on the dataset

Desired Insight:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Github URL to Notebook:
SQL Query Notebook By Denis O'Byrne



Built an Interactive Map with Folium

Visualized Launch Data in an interactive Map

- Used Latitude and Longitude Coordinates of Launch Sites to Add Circle Markers with the site names labeled
- Assigned Launch Outcome (Success/Failure) from the data frame to Classes 1 and 0 respectively and assigned the classes Green and Red markers on the map to Marker Clusters grouped by Launch Site
- Used lines and points to measure (via Haversine's Distance Formula) and label the minimum distances of the launch sites to:
  - Cities
  - Highways
  - Coastlines
  - Railways

Github URL to Notebook :
Folium Map Notebook by Denis O'Byrne

Answered the following Questions:

- Are launch sites in close proximity to railways? Yes
- Are launch sites in close proximity to highways? Yes
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? About 50 km

Build a Dashboard with Plotly Dash

- Built Plotly Dashboard to make an interactive web app to visualize launch data
  - Includes Pie Charts to visualize launch landing success broken down by Launch Site
    - If all sites are selected we get the proportion successful launch landings each site accounts for
    - If we select an individual site we see the proportion of all launches at that site which landed successfully
  - Includes Scatter Plot of Payload Mass (kg) vs Landing Success Rating (0 for Failure, 1 for Success) color coded by booster version
    - Selecting a single site removes points from other sites, Selecting All includes all data
    - Plot Payload Mass Range can be selected by a slider for Min and Max values

Github URL to Notebook that Runs the App :

Plotly Dashboard App for SpaceX launches by Denis O'Byrne Final

Predictive Analysis (Classification)

Build

Building Model:

- Transform data to Scale the columns
- Split Data into Testing and Training Sets
- Selected machine learning algorithms to use for classification (KNN, Decision Tree, SVM, Logistic Regression)
- Use Grid Search and Cross Validation to find best tuning parameters for each model fitting on training sets

Evaluate

Evaluating Model:

- Check accuracy of each model on training and testing sets
- Plot Confusion Matrix

Improve

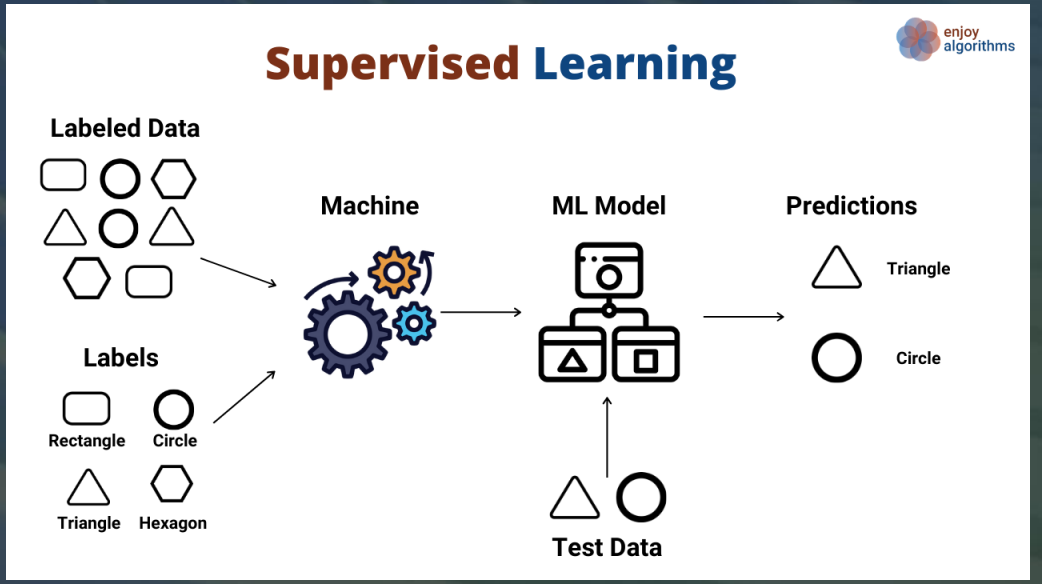
Improving Model:

- Feature Engineering
- Algorithm tuning

Select

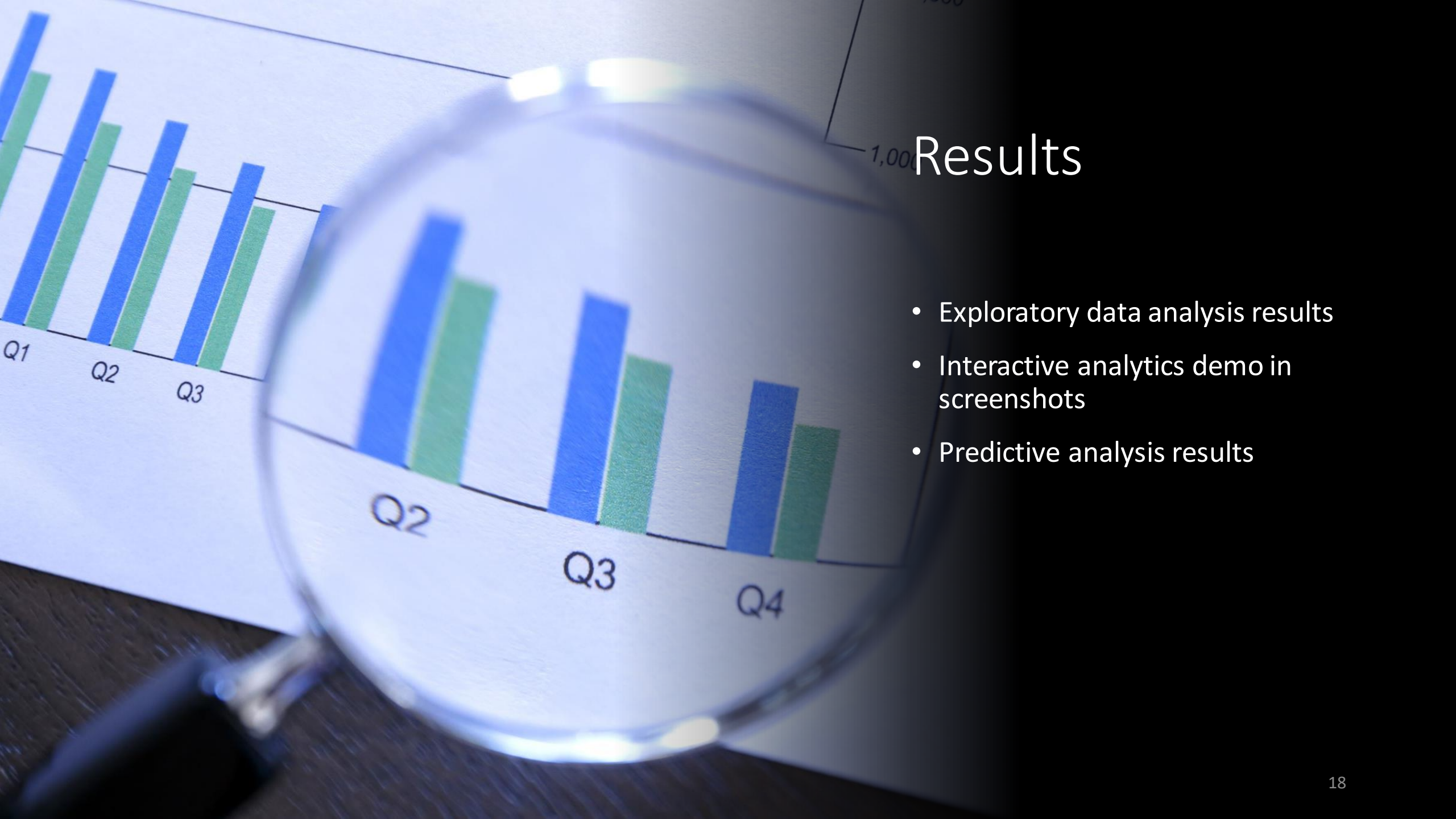
Selecting the best performing classifier:

- Model with the best accuracy score on test set is the best model. If there is a tie, check accuracy on training set as well



Github URL to Notebook :

Machine Learning Prediction Model Notebook by Denis O'Byrne

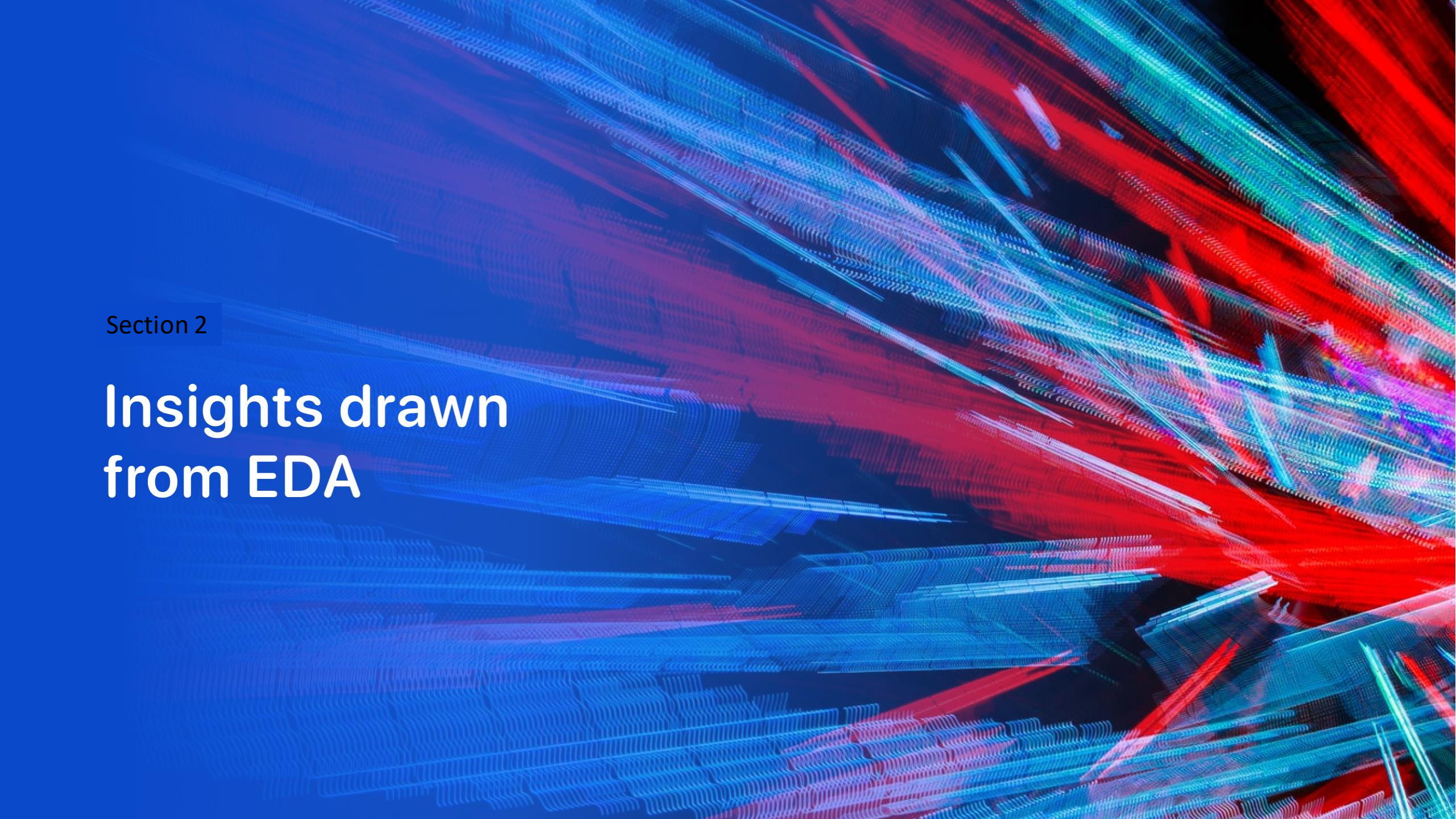


Results

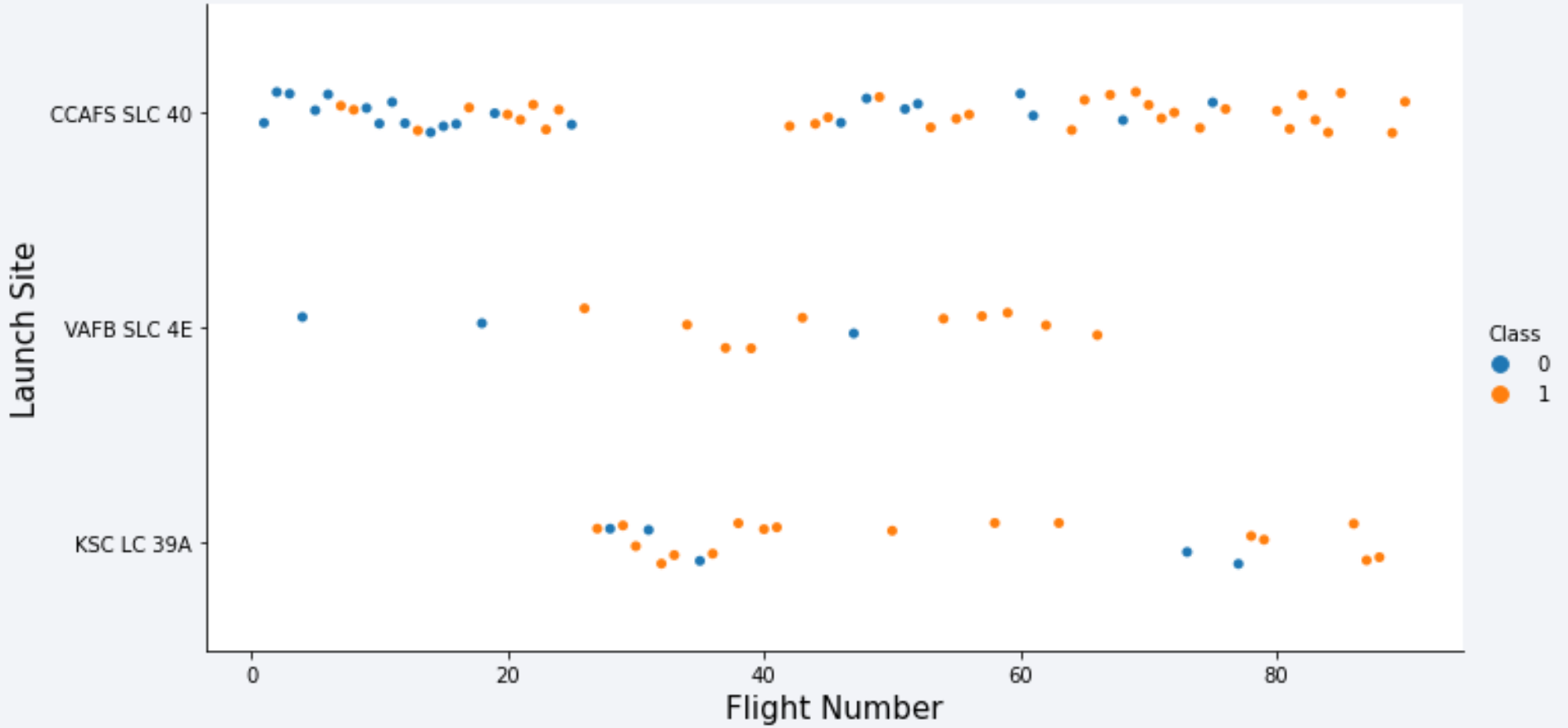
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

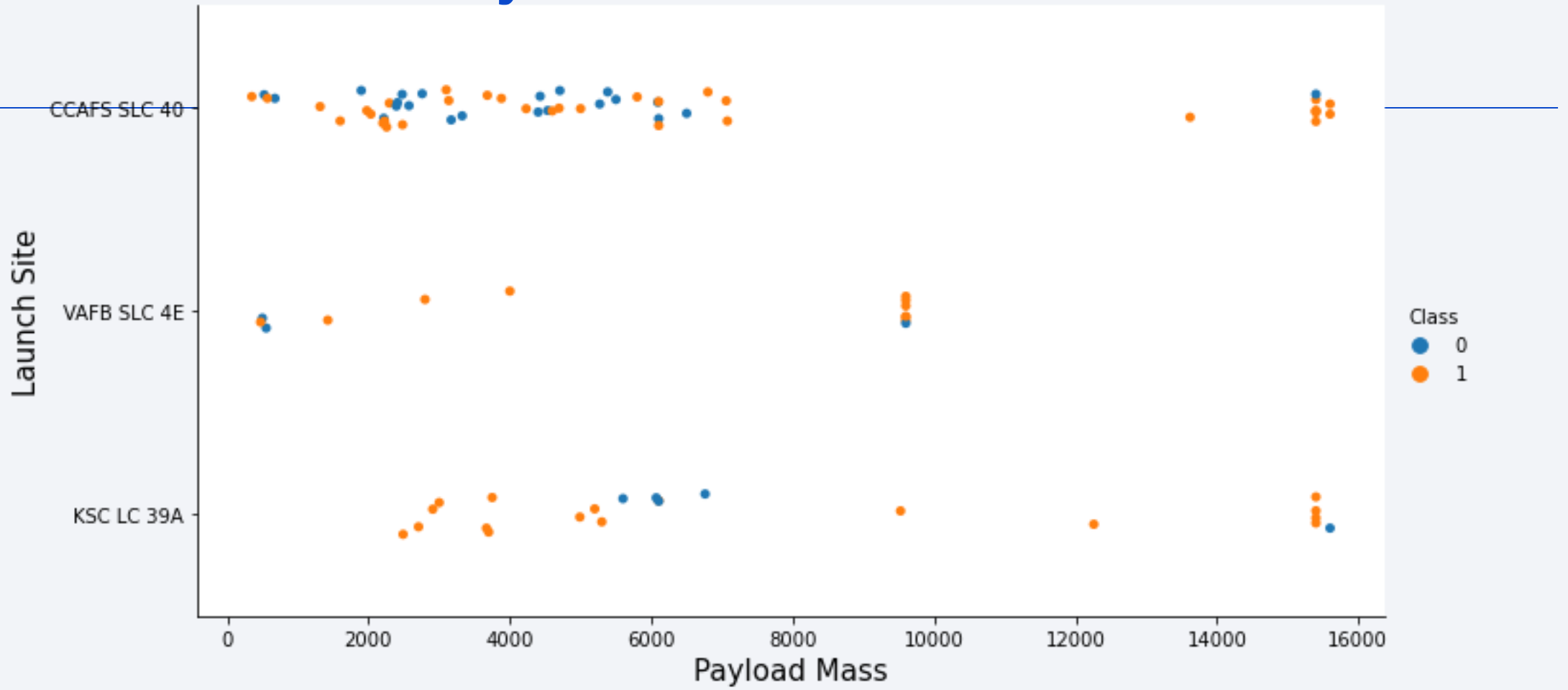


Flight Number vs. Launch Site



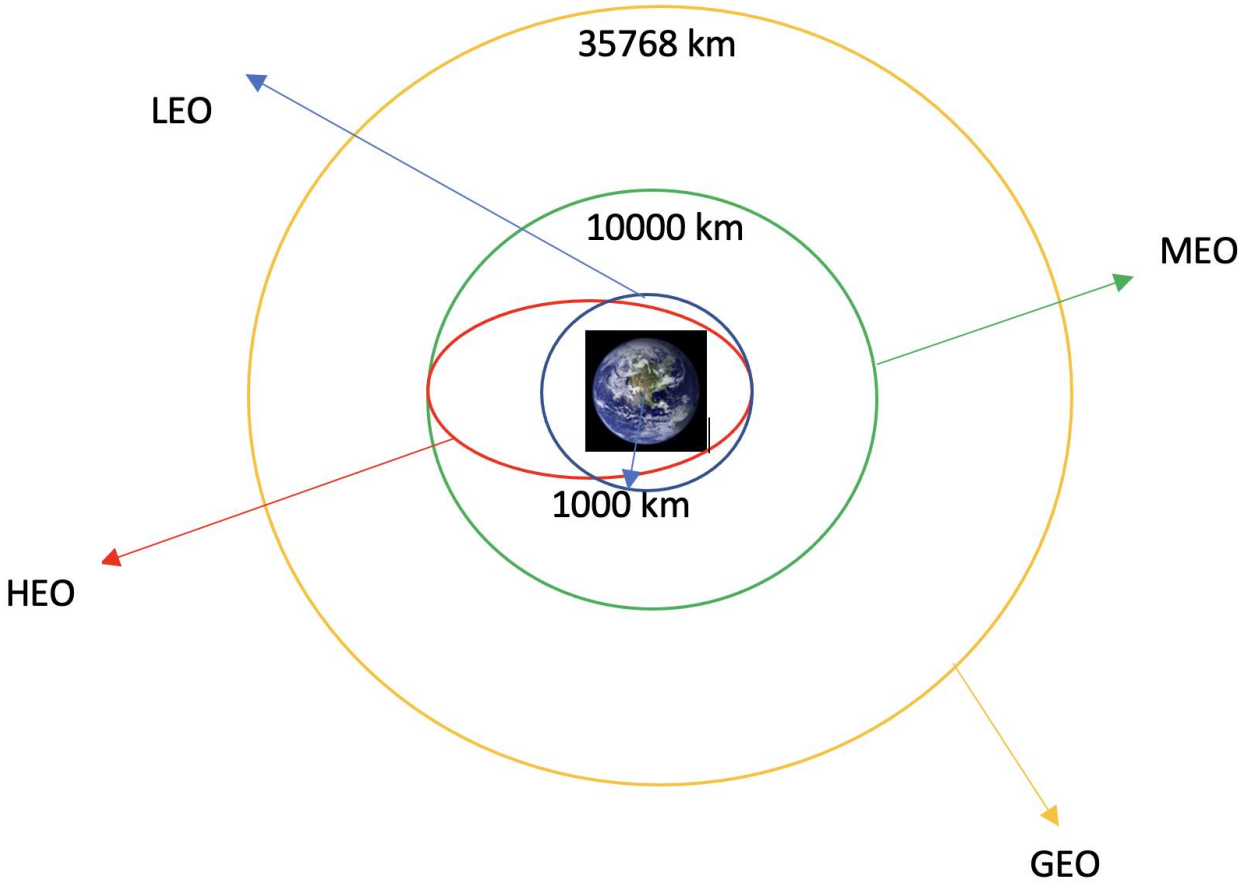
We can see that as the flight number increases the number of successful landings increases
We also see that as more flights occur at a given launch site, the number of successful landings increases

Payload vs. Launch Site



- We can see that as payload mass increases for Site CCAFS SLC 40, the probability of a successful landing increases
- There is not a clear correlation between payload mass and launch success for the other two sites
- Lastly we find that middle mass launches (~9,000kgs) are usually performed at VAFB SLC 4E but the other two sites launch heavy masses (>10,000kgs) while VAFB SLC 4E does not

Orbit Types and Radius Visualization



- **LEO**: Low Earth orbit (LEO)is an Earth-centred orbit with an altitude of 2,000 km (1,200 mi) or less (approximately one-third of the radius of Earth),[1] or with at least 11.25 periods per day (an orbital period of 128 minutes or less) and an eccentricity less than 0.25.[2] Most of the manmade objects in outer space are in LEO \[1].
- **VLEO**: Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 450 km. Operating in these orbits can provide a number of benefits to Earth observation spacecraft as the spacecraft operates closer to the observation\[2].
- **GTO** A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. Located at 22,236 miles (35,786 kilometers) above Earth's equator, this position is a valuable spot for monitoring weather, communications and surveillance. Because the satellite orbits at the same speed that the Earth is turning, the satellite seems to stay in place over a single longitude, though it may drift north to south," NASA wrote on its Earth Observatory website \[3] .
- **SSO (or SO)**: It is a Sun-synchronous orbit also called a heliosynchronous orbit is a nearly polar orbit around a planet, in which the satellite passes over any given point of the planet's surface at the same local mean solar time [4] . (800 – 1000 km)
- **ES-L1** :At the Lagrange points the gravitational forces of the two large bodies cancel out in such a way that a small object placed in orbit there is in equilibrium relative to the center of mass of the large bodies. L1 is one such point between the sun and the earth \[5] .
- **HEO** A highly elliptical orbit, is an elliptic orbit with high eccentricity, usually referring to one around Earth \[6].
- **ISS** A modular space station (habitable artificial satellite) in low Earth orbit. It is a multinational collaborative project between five participating space agencies: NASA (United States), Roscosmos (Russia), JAXA (Japan), ESA (Europe), and CSA (Canada) \[7]
- **MEO** Geocentric orbits ranging in altitude from 2,000 km (1,200 mi) to just below geosynchronous orbit at 35,786 kilometers (22,236 mi). Also known as an intermediate circular orbit. These are "most commonly at 20,200 kilometers (12,600 mi), or 20,650 kilometers (12,830 mi), with an orbital period of 12 hours \[8]
- **HEO** Geocentric orbits above the altitude of geosynchronous orbit (35,786 km or 22,236 mi) \[9]
- **GEO** It is a circular geosynchronous orbit 35,786 kilometres (22,236 miles) above Earth's equator and following the direction of Earth's rotation \[10]
- **PO** It is one type of satellites in which a satellite passes above or nearly above both poles of the body being orbited (usually a planet such as the Earth

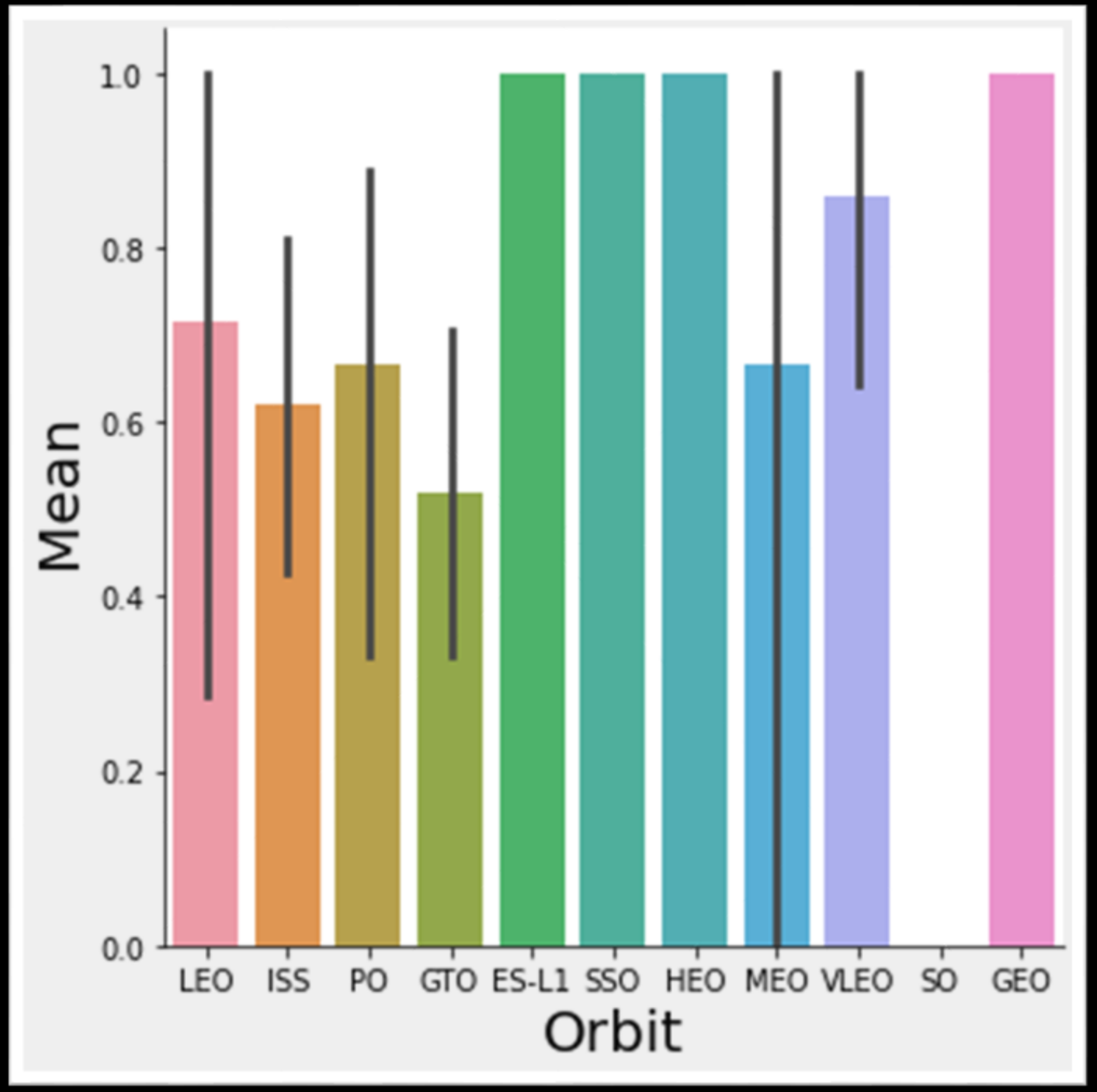
Success Rate vs. Orbit Type

ES-L1, SSO, HEO, and GEO orbits have perfect landing scores

From the previous slide we find that orbits within 450-10000 kilometers have the highest rate of success

Launches beyond 10000 km have lower success ratings

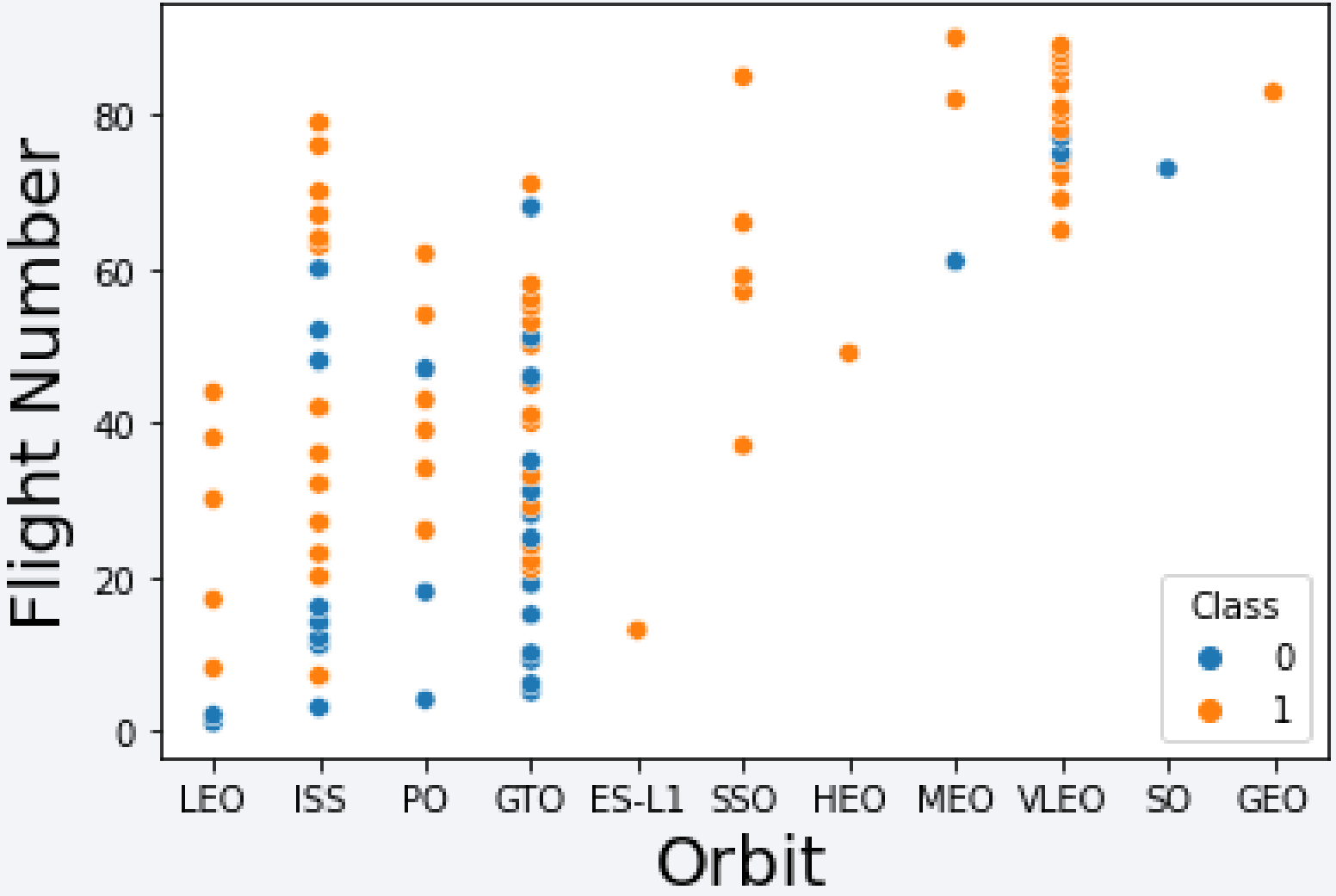
Launches between 1000-2000 km have less success than launches between 1000-450 km (lower altitude) and launches between 2000-10000km (higher altitude)



Flight Number vs. Orbit Type

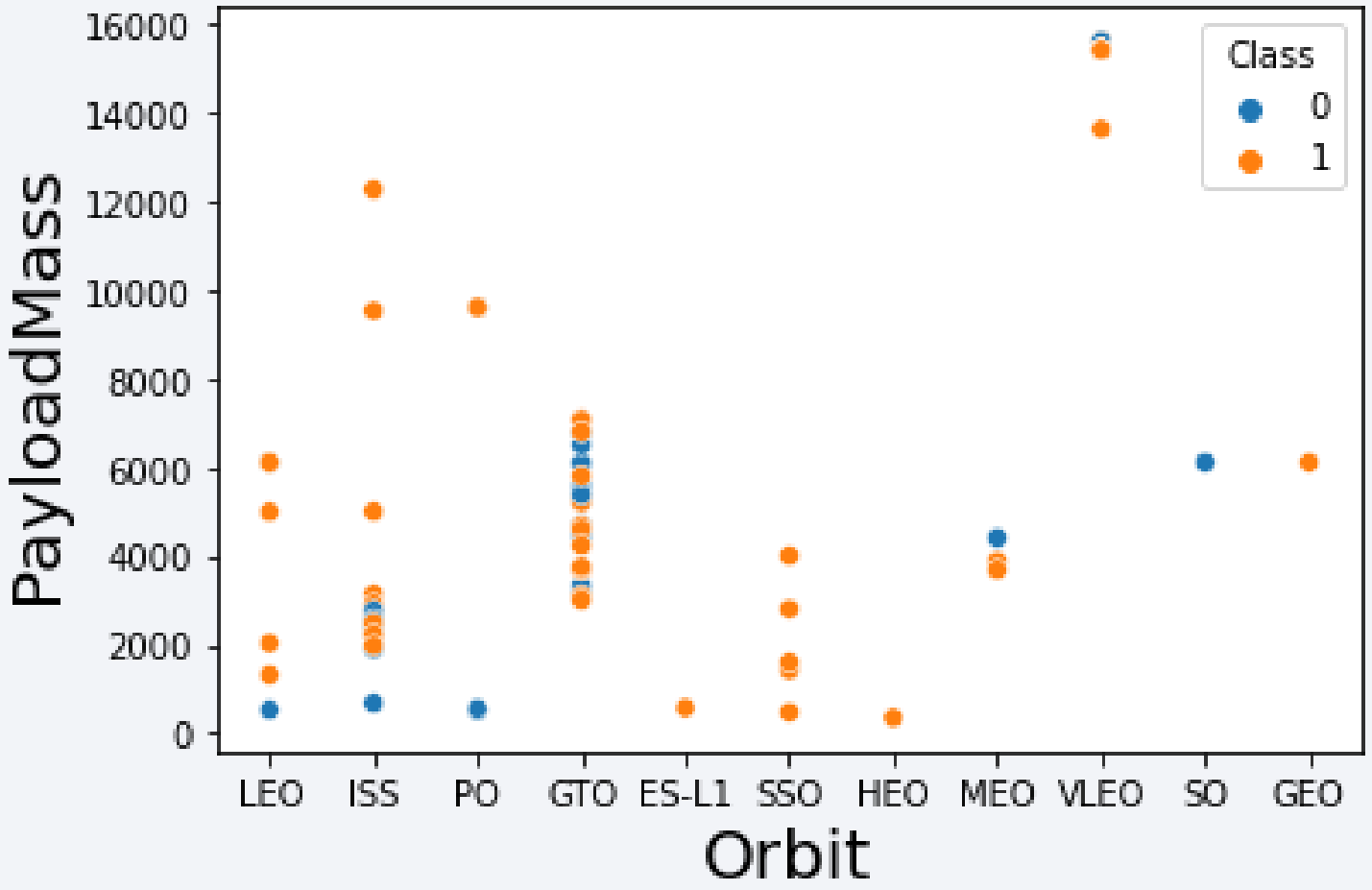
For LEO and PO orbits the Success appears related to the number of flights

There seems to be no relationship between flight number and success when in GTO or ISS orbit



Payload vs. Orbit Type

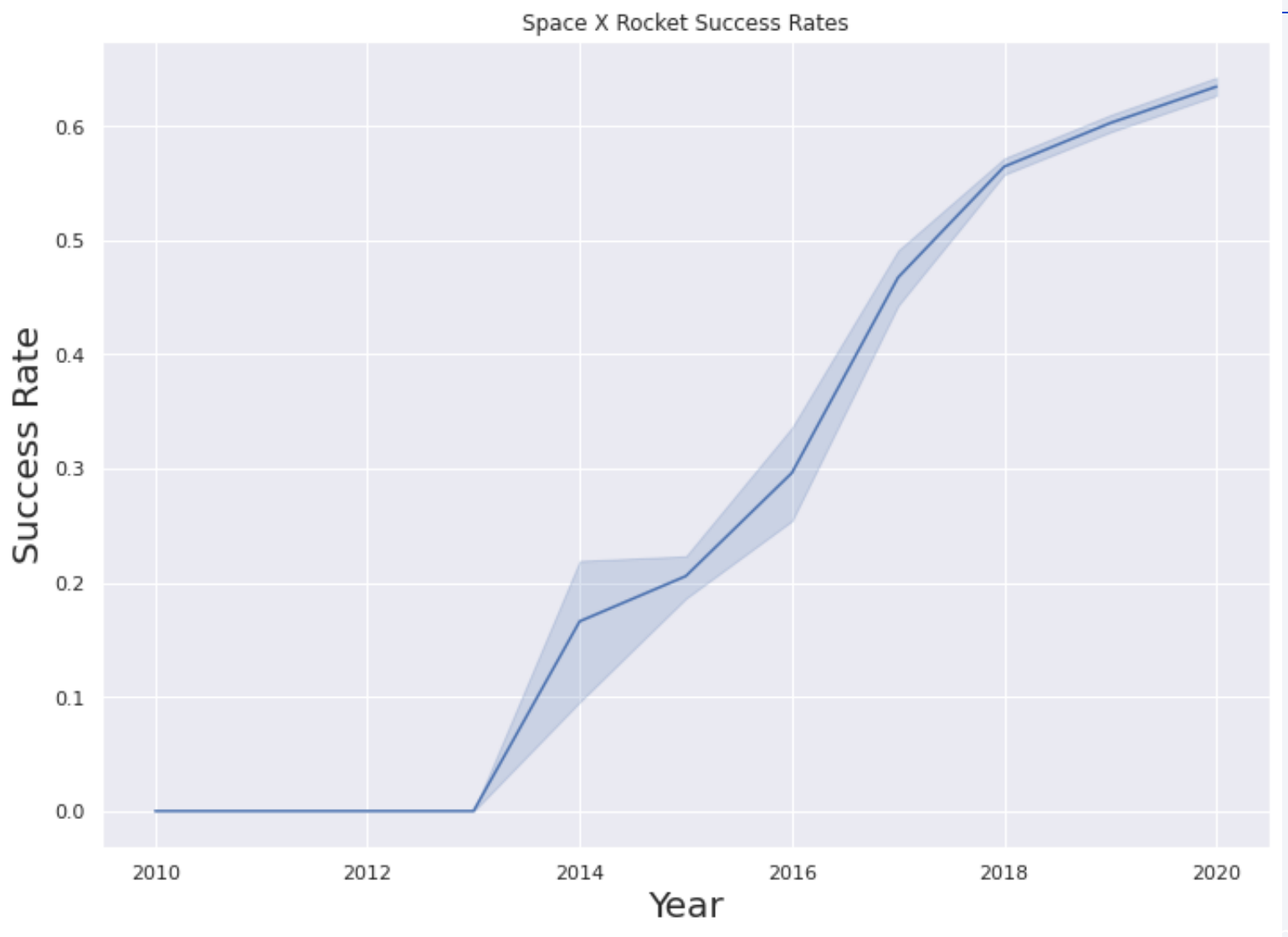
- PO, LEO and ISS Orbits have higher probability for successful landings as payload mass increases
- GTO orbits have no correlation between payload mass and landing success
- All other orbits do not have enough data to indicate a relationship between payload mass and orbit type



Launch Success Yearly Trend

- We can see that there were no successful landings prior to 2013

- We also see that since 2013 the probability of successful landings has increased every year however in recent years (2018-2020) the yearly rate of increase has declined with a maximum probability of success being about 0.65 in 2020



EDA with SQL

All Launch Site Names

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

Using SQL Magic we can query the database as follows

%sql Select Unique Launch_Site from SpaceX

By specifying Unique the database only returns distinct entries from the column launch_site in the table SpaceX

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

Using multi-line sql magic we can query the database as follows:

```
%%sql Select * from SpaceX
where Launch_Site like 'CCA%'
limit 5
```

By selecting * we request all columns of the data frame

The where clause requires results to have a Launch_Site beginning with "CCA"

The limit clause restricts the database to only return the first 5 rows in the matching the request

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

Calculate the total payload mass carried by boosters from NASA (CRS)

Using multi-line sql magic we can query the database as follows:

```
%%sql Select customer, sum(payload_mass_kg) as "Total Payload Mass" from
        (Select customer, payload_mass_kg from SpaceX
        where customer LIKE 'NASA (CRS)')
        GROUP BY CUSTOMER
```

The sum call caluclates the total

The as clause renames the returned column to Total Payload mass

We select from a subquery that queries the list of customers and their payload masses but limits the search to rows where the customer is Nasa (CRS)

The group by clause makes sure our sum combines the payloads of all NASA (CRS) rows

customer	Total Payload Mass
NASA (CRS)	45596

Average Payload Mass by F9 v1.1

Using multi-line sql magic we can query the database as follows:

```
%%sql Select BOOSTER_VERSION, AVG(payload_mass_kg) as "AVERAGE Payload Mass" from

    (Select BOOSTER_VERSION, payload_mass_kg from SpaceX

    where BOOSTER_VERSION LIKE 'F9 v1.1')

    GROUP BY BOOSTER_VERSION
```

The AVG call calculates the average payload mass

The as clause renames the returned column to AVERAGE Payload mass

We select from a subquery that queries the list of boster_versions and their payload masses but limits the search to rows where the booster_version is F9 v1.1

The group by clause makes sure our average is calculated over all entries with the Booster_version F9 v1.1

booster_version	AVERAGE Payload Mass
F9 v1.1	2928

First Successful Ground Landing Date on Ground Pad

Using multi-line sql magic we can query the database as follows:

```
%%sql Select MIN(DATE) as "FIRST SUCCESS" FROM
      (SELECT DATE FROM SPACEX
      WHERE LANDING_OUTCOME LIKE 'Success (Ground Pad)')
```

The MIN call selects the earliest date meeting the criteria

The as clause renames the returned column to FIRST SUCCESS

We select from a subquery that queries the list of DATES from the SpaceX table where the LANDING_OUTCOME is Success (Ground Pad)

FIRST SUCCESS
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000 kg

Using multi-line sql magic we can query the database as follows:

```
%%sql SELECT BOOSTER_VERSION,
payload_mass_kg,
Landing_Outcome  FROM SPACEX

where 4000 < payload_mass_kg and
payload_mass_kg < 6000 and
Landing_Outcome = 'Success (drone ship)'
```

The where clause restricts results to those with a payload mass between 4000 and 6000 kgs and a Landing_Outcome of Success (drone ship)

The database returns the Booster Version, Payload mass, and Landing outcome for these data

booster_version	payload_mass_kg	landing_outcome
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	Success (drone ship)

Total Number of Success and Failure Mission Outcomes

Using multi-line sql magic we can query the database as follows:

```
%%sql SELECT Mission_Outcome,
count(Mission_Outcome) as "Total" FROM
SPACEX

Group by Mission_Outcome
```

Select Mission Outcomes and count of each type of Mission outcome as the Total

Group by clause ensures we count the success and failures as separate groups

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

Boosters Which Have Carried the Maximum Payload

Using multi-line sql magic we can query the database as follows:

```
%%sql SELECT Unique
Booster_version, payload_mass_kg
FROM SPACEX

where payload_mass_kg = (Select
max(payload_mass_kg) from SPACEX)
```

Select Unique Booster Versions to ensure we get distinct results

Returned Payload Mass to see what the Max Payload is

Where clause checks that the payload mass equals the max payload which is found via a sub query

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
F9 B5 B1060.2	15600
F9 B5 B1060.3	15600

2015 Launch Records that Failed in Drone Ship

Using multi-line sql magic we can query the database as follows:

```
%%sql Select landing_outcome, booster_version, launch_site, DATE from SPACEX

where landing_outcome = 'Failure (drone ship)' and Year(DATE) = 2015
```

Select Landing_Outcome, Booster Versions, Launch_Site, and Date as requested

Where clause restricts results to have a landing outcome of Failure (Drone Ship) and a launch date in the year 2015

landing_outcome	booster_version	launch_site	DATE
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015-01-10
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015-04-14

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

Using multi-line sql magic we can query the database as follows:

```
%%sql select landing_outcome,
count(landing_outcome) as "Total" from Spacex

where DATE between '2010-06-04' and '2017-03-20'

group by landing_outcome

order by "Total" desc
```

landing_outcome	Total
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

Select Landing_Outcome and the count of each landing outcome listed as the Total

Where clause restricts results to have a launch date between 2010-06-04 and 2017-03-20

Group by clause ensures we count landing outcomes separately

Order By desc orders the results in descending order as desired

Section 3

Launch Sites Proximities Analysis

Global Map of SpaceX Launch Sites

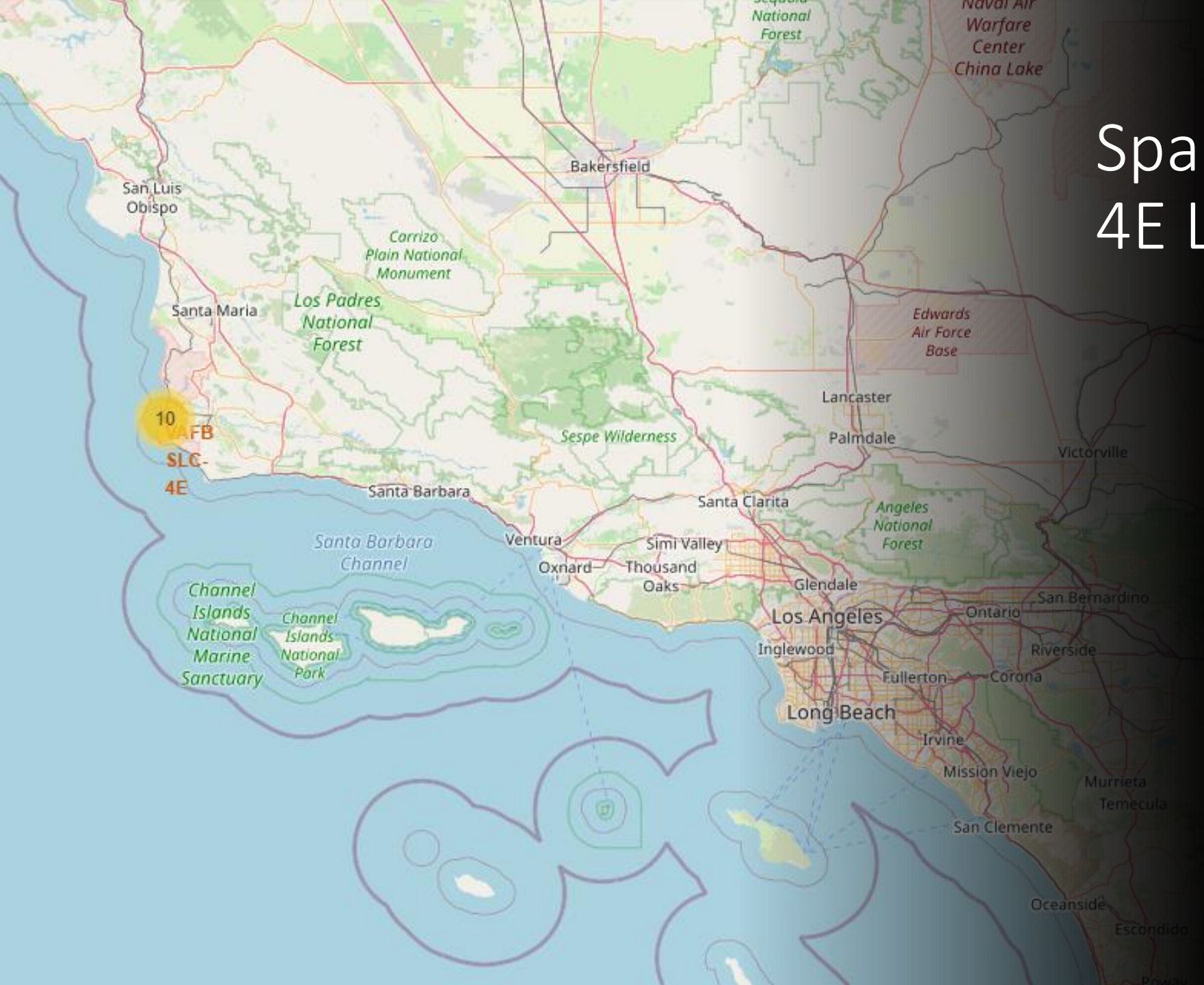
Space X has Launch Sites Exclusively in the United States

There are 4 total Launch Sites

1 is Located on the West Coast in California

3 are located on the East Coast in Florida

SpaceX VAFB SLC-4E Launch Site



- Space X has a single Launch Site Stationed off the Coast of Santa Maria in California
- Our Cluster Marker indicates 10 Falcon 9 launches have taken place at this site

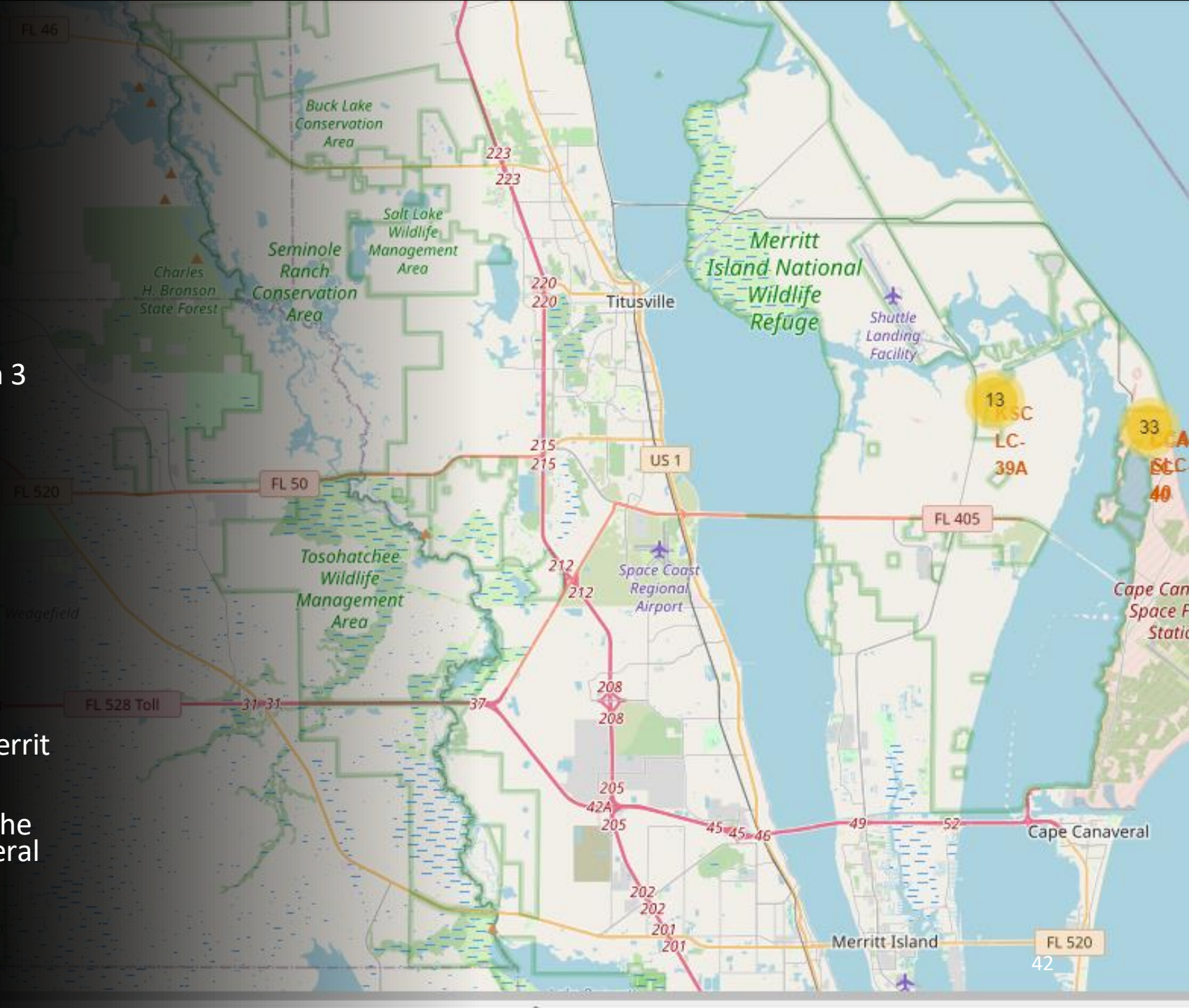
SpaceX VAFB SLC-4E Launch Site

- Zooming in we can see our markers color coded to indicate how many launches landed successfully
- 4 Rockets Landed
- 6 Rockets Failed to Land

SpaceX Florida Launch Sites

- Space X has 2 separate bases with 3 total launch sites located on:
  - ❑ Merritt Island:
    - A. KSC LC-39A
  - ❑ Cape Canaveral:
    - A. CCAFS LC40
    - B. CCAFS SLC-40

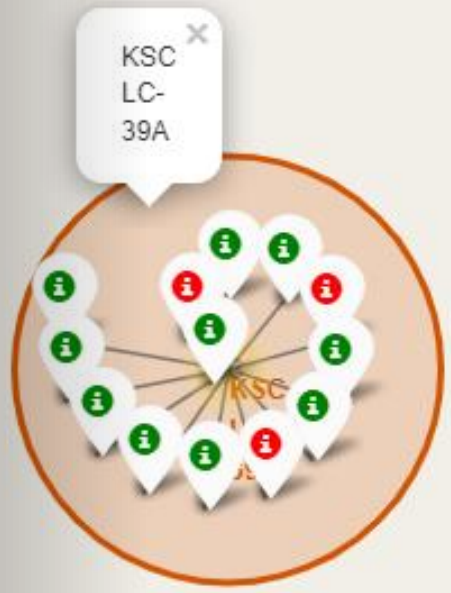
- Our Cluster Marker indicates 13 Launches have taken place at the Merrit Island Base
- 33 Launches have taken place at the two launch sites on the Cape Canaveral Base



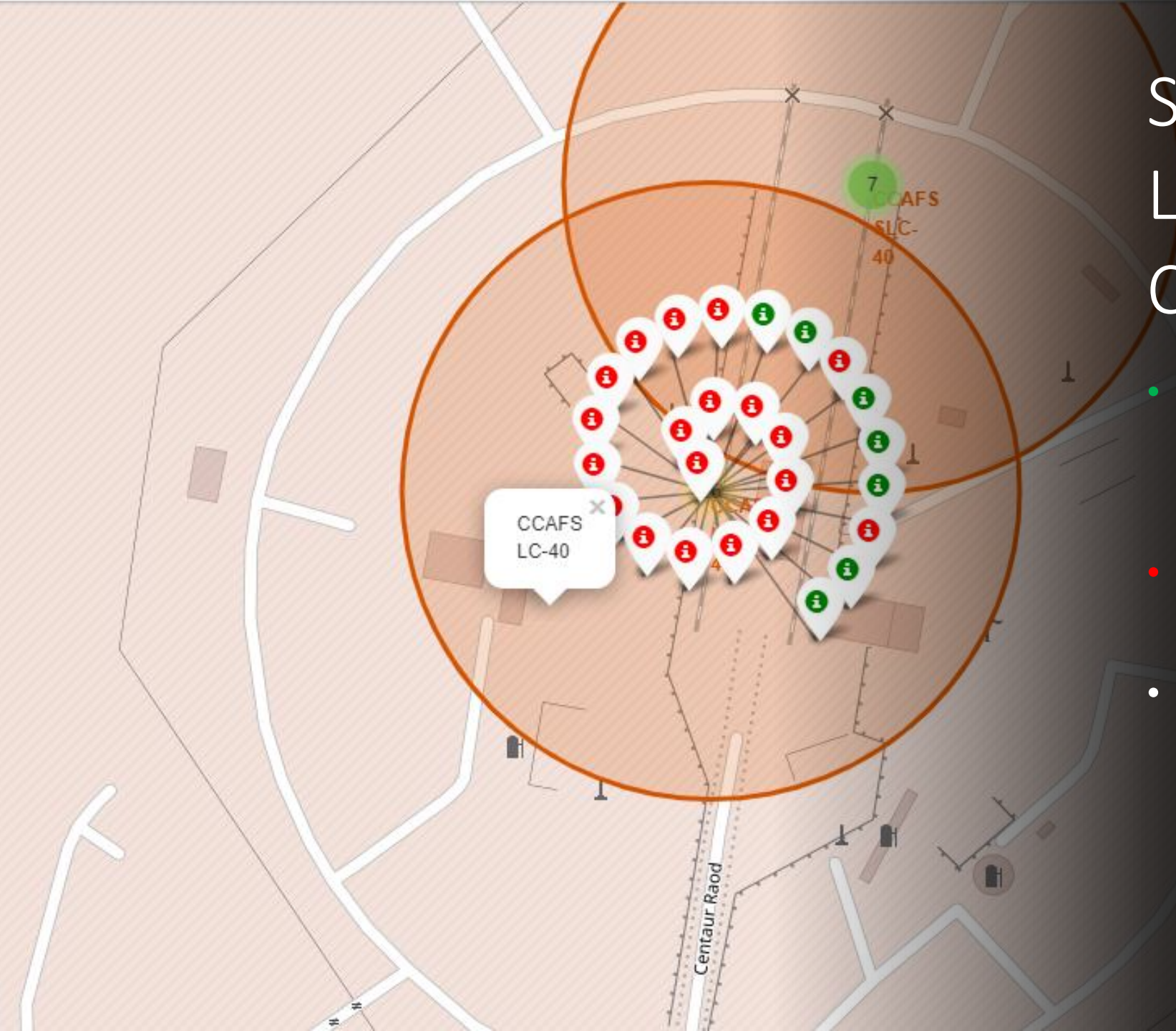
SpaceX KSC LC-39A Launch Site Merritt Island

- 10 Rockets Landed

- 3 Rockets Failed to Land

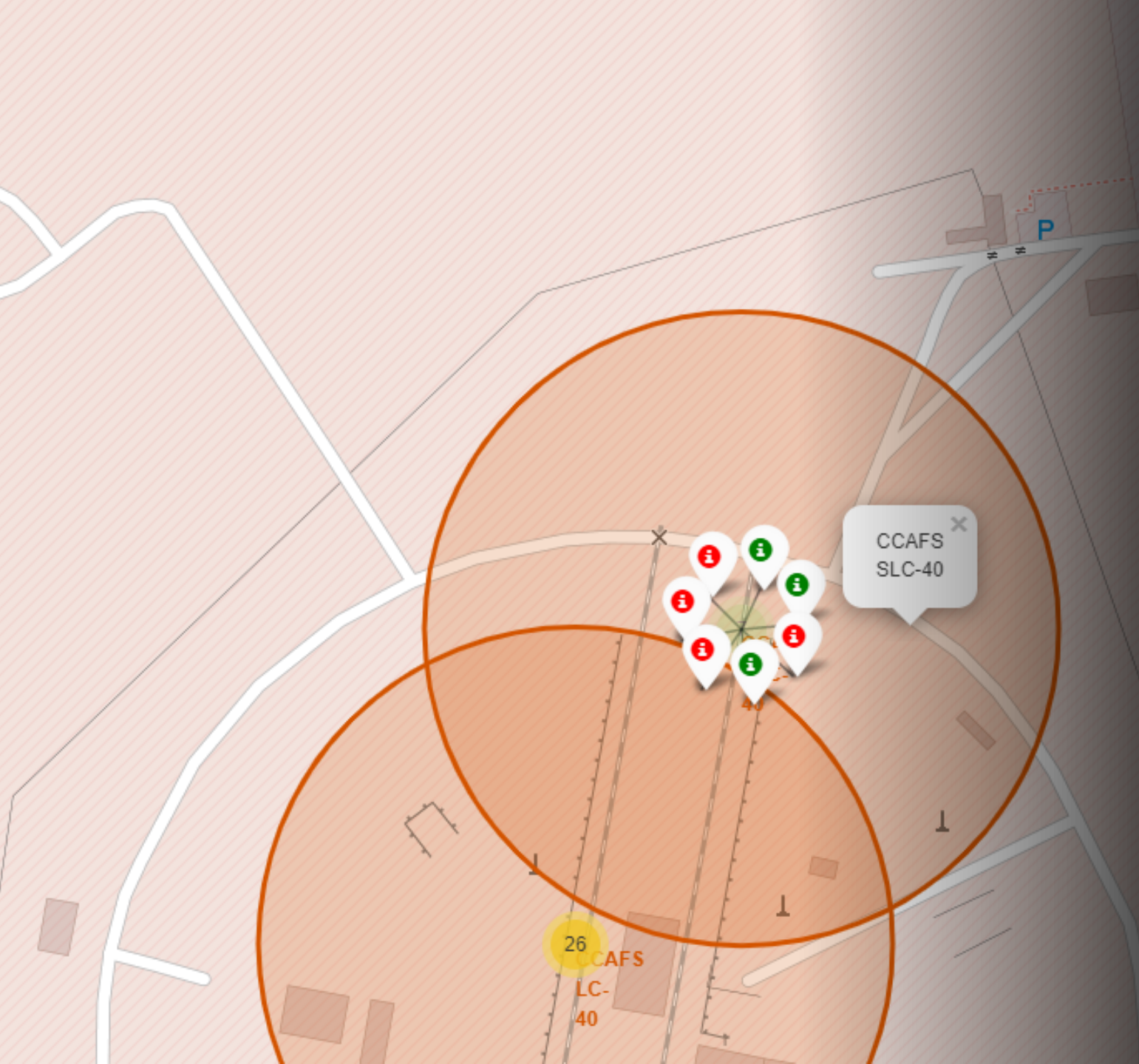


SpaceX CCAFS LC-40 Launch Site Cape Canaveral



- 7 Rockets Landed
- 19 Rockets Failed to Land
- 26 Total Launches at this Site

SpaceX CCAFS SLC-40 Launch Site Cape Canaveral



- 3 Rockets Landed
- 4 Rockets Failed to Land
- 7 Total Launches at this Site

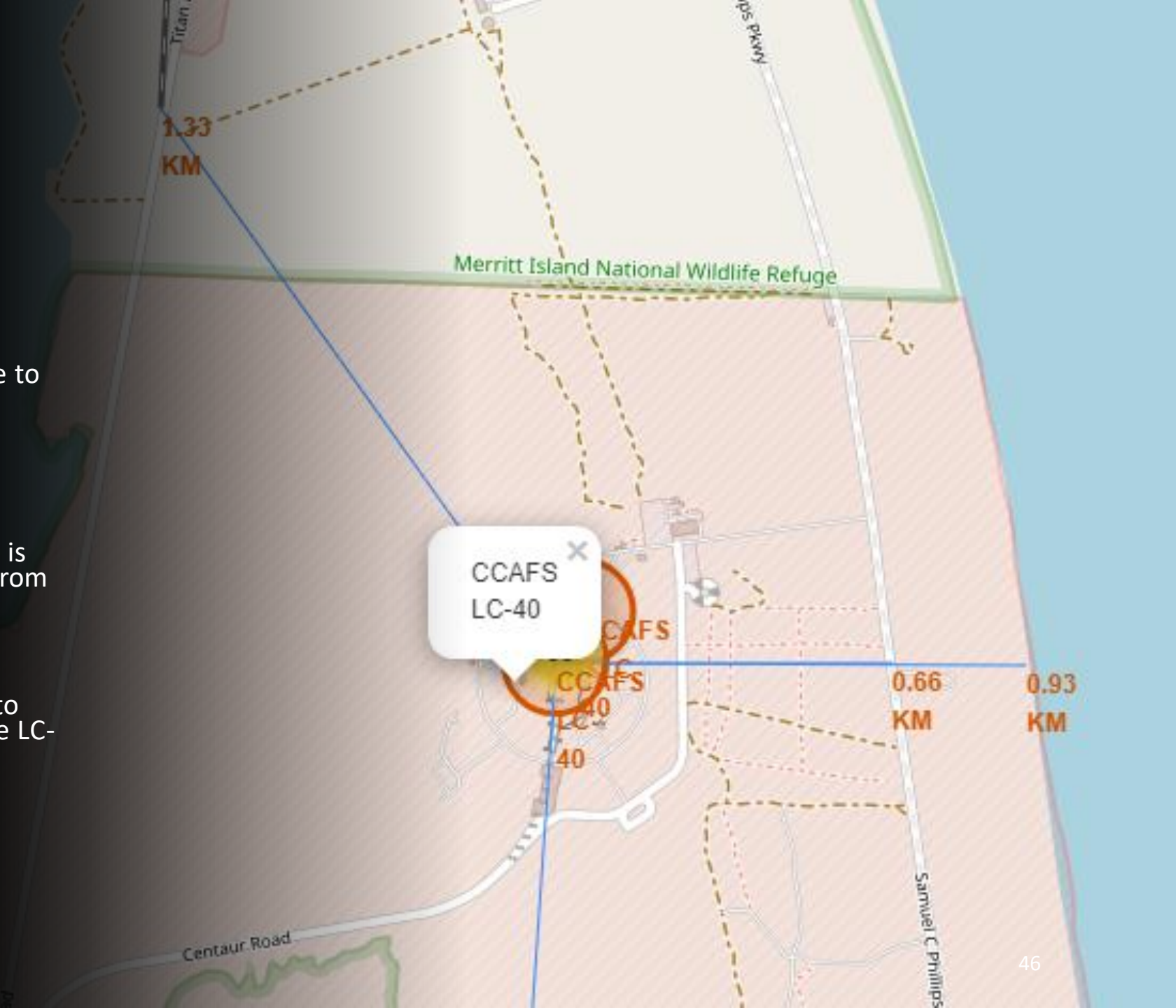
Infrastructure near Cape Canaveral Sites

- We can see that the nearest coastline to the Cape Canaveral sites is within 1 kilometer and exactly measured to 0.93km for the LC-40 site

- We can see that the nearest highway is within 0.7 km and is exactly 0.66km from the LC-40 site

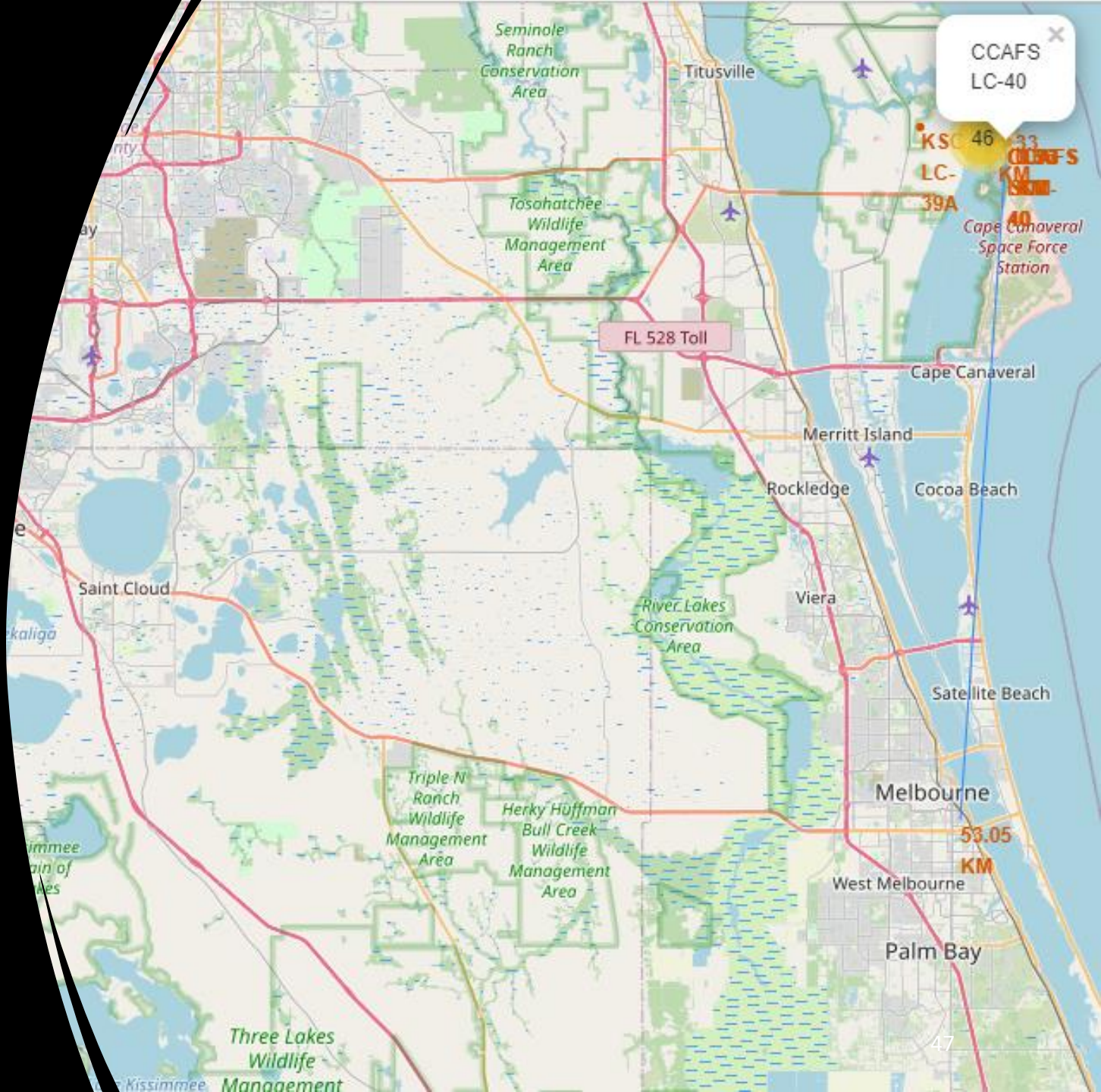
- The nearest railway is within 1.5 km to the base and exactly 1.33km from the LC-40 site

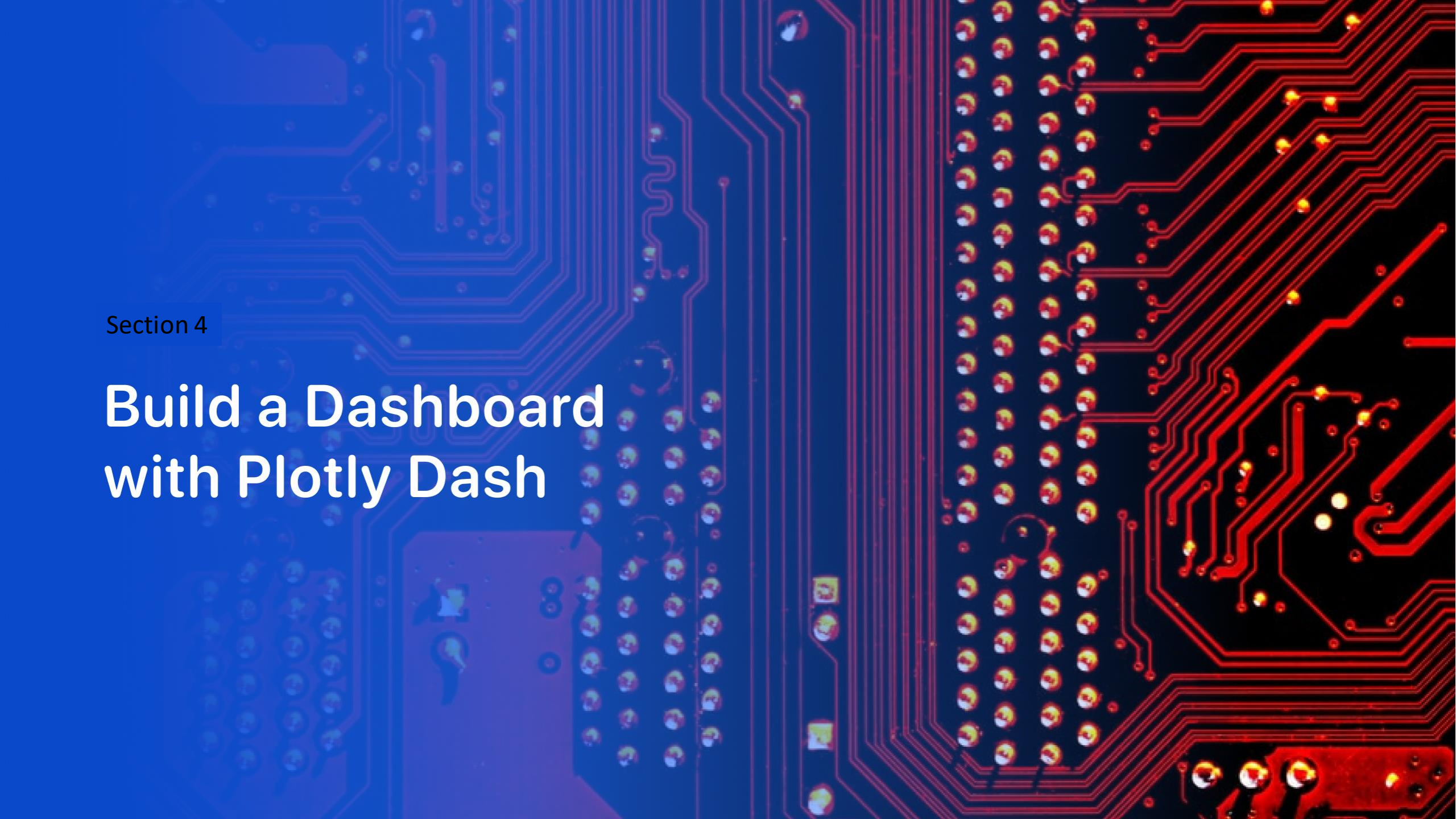
- From this we recognize that infrastructure is in close proximity to launch sites for easy access to manufactured parts



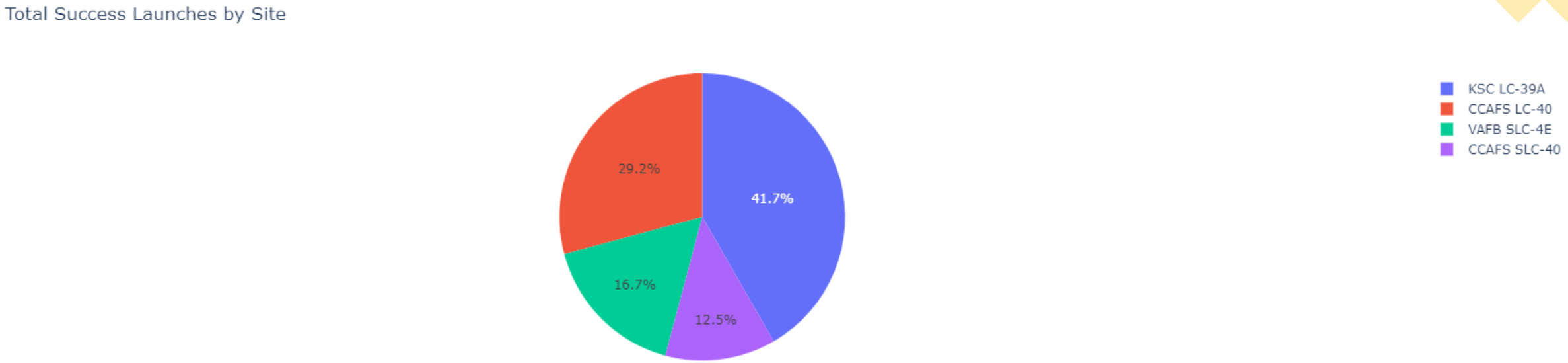
City distance to Cape Canaveral Sites

- The nearest city to the launch sites in Florida is Melbourne located 53.05Km from the Cape Canaveral LC-40 Site
- Proximity to the city is not a priority for SpaceX as we previously saw infrastructure is built near the launch sites meaning travel time for parts and employees is minimal



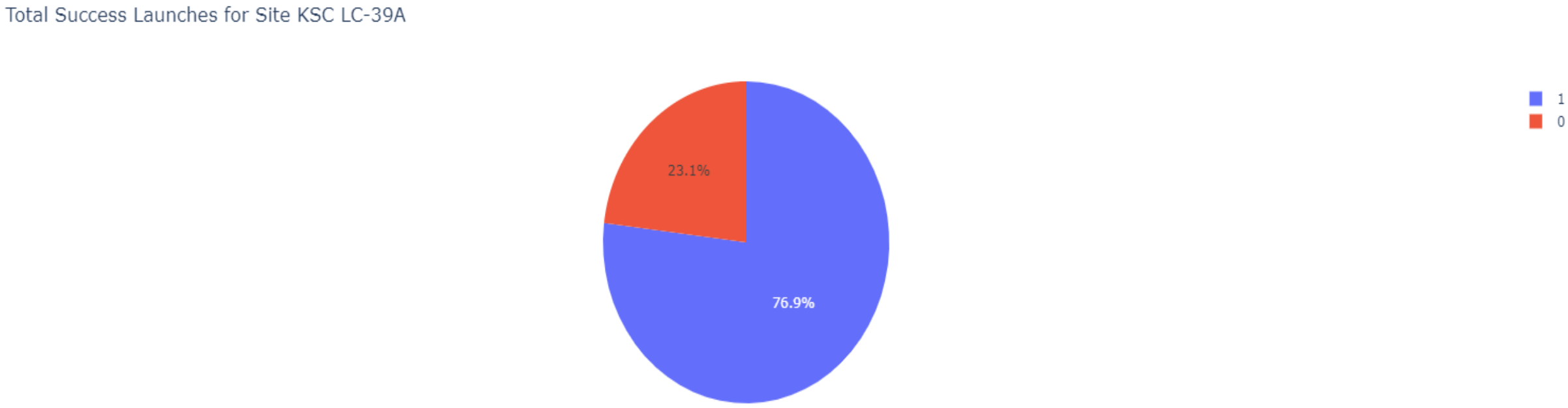

Section 4
Build a Dashboard with Plotly Dash

Probability of launch site given Successful Landing



- We see that the KSC LC-39A Launch site accounts for the largest percentage of the total number of successful landings at 41.7%

Launch Site with the Highest Probability of Success

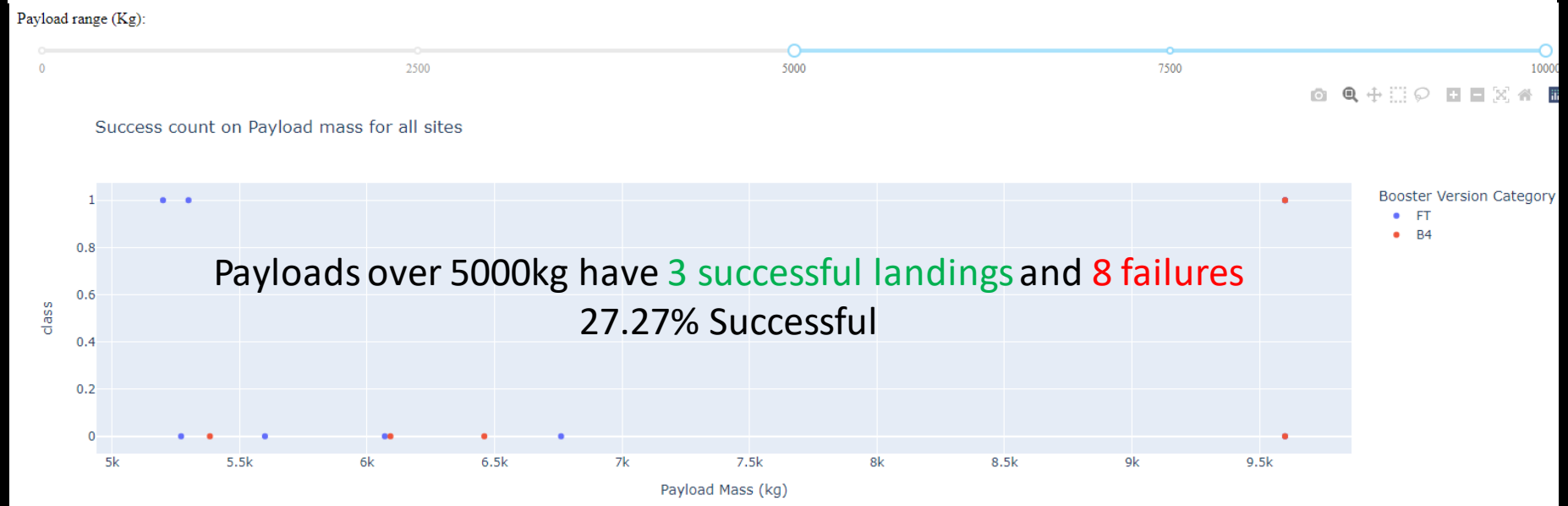


- The KSC LC-39A Launch Site also has the highest probability of success per launch
- 76.9% of all launches at the KSC LC-39A Site Land Successfully
- 23.1% of all launches at the KSC LC-39A Site Fail to Land

Payload Weight vs Success Labeled by Booster Version

Heavy Vs Light Weight

- Payloads under 5000kgs have a higher probability of success than payloads over 5000kgs
- Booster Versions FT and B4 are the only boosters to carry payloads over 5000kgs
- All boosters carry payloads below 5000 kgs



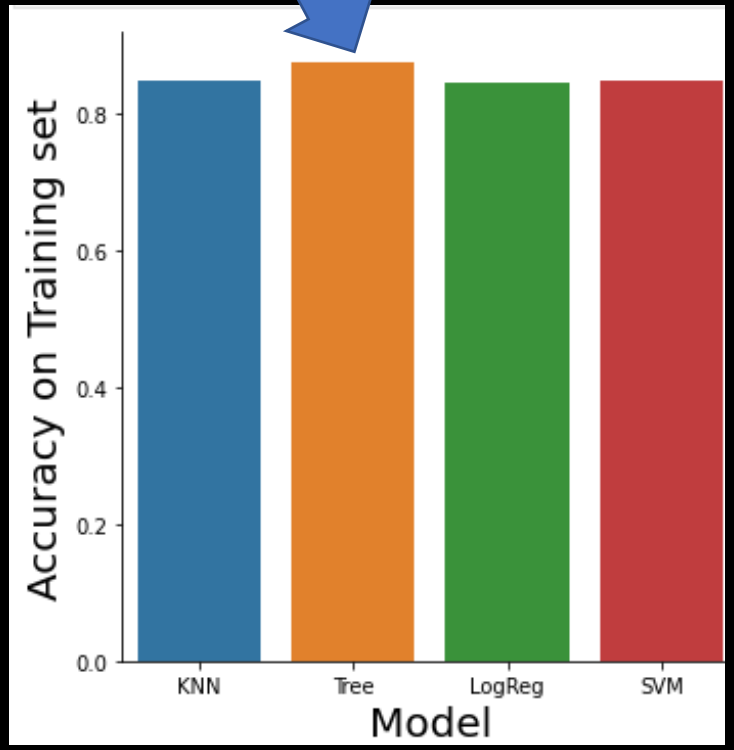
Section 5

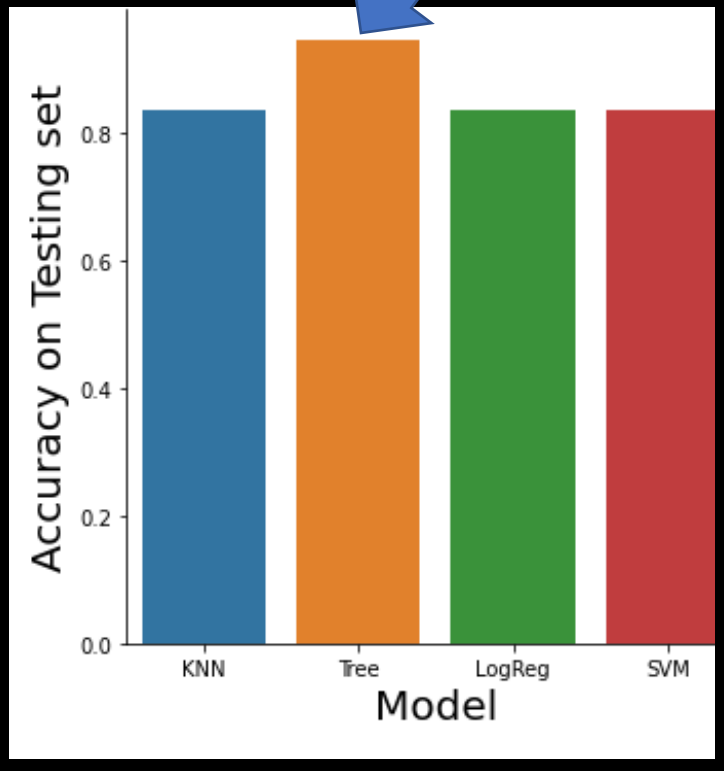
Predictive Analysis (Classification)

Classification Accuracy

Decision tree highest training accuracy = 0.875

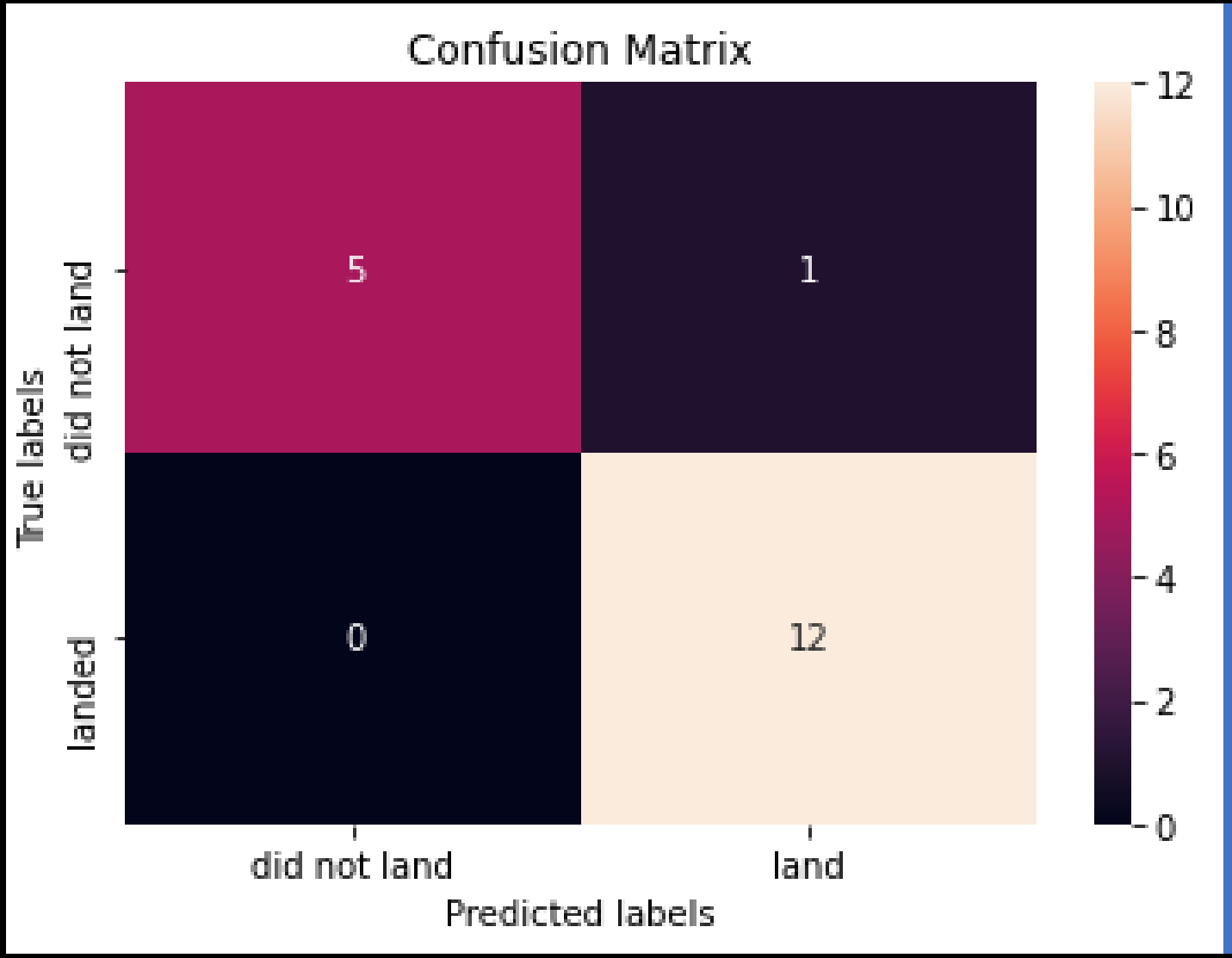
Decision tree highest testing accuracy = 0.9444





- The data is split into a testing and training set for developing the model
- Model Tuning is performed using Grid Search and 10 fold cross validation on training set
- Bar Graphs Show Training and Testing accuracy of all 4 models
- Decision Tree Model has the Highest accuracy on testing and training data
- All models are resilient to new data and retain their accuracy well on the testing set
- Decision Tree is clearly the best performing model

```
Best Model is Tree with an accuracy of 0.875
Best Params is : {'criterion': 'gini', 'max_depth': 14, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter': 'random'}
```



Decision Tree Confusion Matrix Testing Set

- We see that our model is able to correctly predict that 5 of the 6 testing points that fail to land
- Sensitivity = 5/6 = 83.33%
- Our model is able to correctly predict that all 12 rockets that land will land
- Specificity = 12/12 = 100%
- Accuracy = (5+12)/(5+1+12+0) = 94.44%

Conclusions

- We were able to build a decision tree model that can predict the probability of Falcon 9 Rocket Stage 1 Landing Successfully with an 94.44% accuracy on our out of sample data and 87.5% accuracy on in sample data
- We found that low weight payloads are more likely to land successfully than heavy payloads
- We found that SpaceX engineers have been improving the probability of success every year since 2013 but progress has begun to slow down reaching a maximum yearly success percentage of about 63% in 2020 meaning our model will have to be refined as time passes to keep up to date
- The KSC LC-39A Launch Site also has the highest probability of success per launch
- The type of orbit required for the launch has an impact on the landing success, the ES-L1, SSO, HEO, and GEO orbits have the highest rate of success for landing

Appendix

- IBM DB2:
  - The data from this project for the SQL section was stored in an IBM DB2 cloud storage environment

Appendix

- Haversine's Formula used to calculate distances on Folium Maps:

You can calculate the distance between two points on the map based on their `Lat` and `Long` values using the following method:

```
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
```

Thank you!