IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Rattapong Pojpatinya
March 11, 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection with API and Web Scraping
  - Exploratory Data Analysis with Visualization
  - Exploratory Data Analysis with SQL
  - Interactive Map with Folium
  - Dashboard with Plotly Dash
  - Predictive Analysis
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive Maps and Dashboard
  - Predictive results

Introduction

- Project background and context
  - The objective of this project is to predict the successful landing of the Falcon 9’s first stage. According to SpaceX's website, the Falcon 9 launch carries a price tag of 62 million dollars, which is significantly lower than other providers' cost of over 165 million dollars per launch. This cost difference is because SpaceX could reuse the initial stage. By predicting the landing outcome of the first stage, we can ascertain the cost of a launch, a valuable piece of information for companies seeking to compete with SpaceX in the rocket launch market.
- Problems you want to find answers
  - What are the characteristics of the successful and failed landing ?
  - What are the effects of each attribute to the successful and failed landing ?

Section 1

Methodology

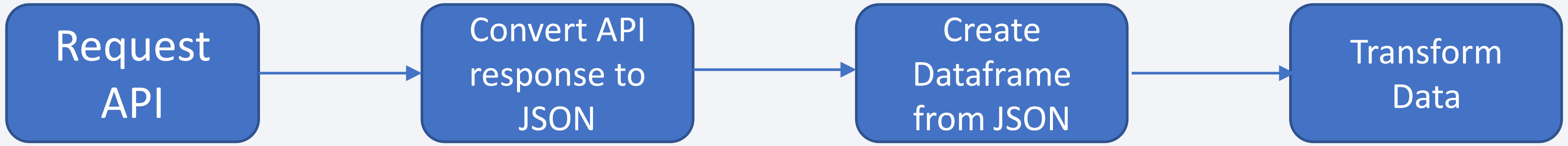
Methodology

Executive Summary

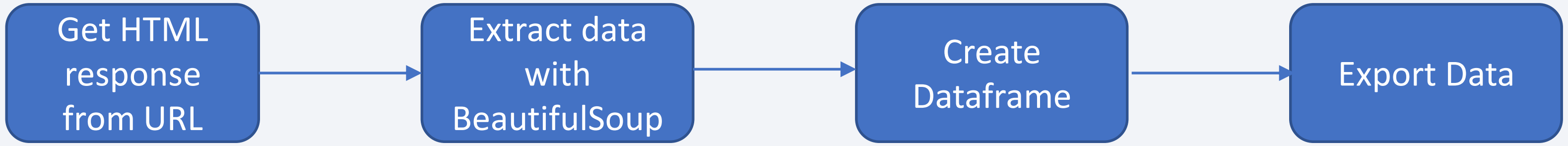
- Data collection methodology:
  - SpaceX REST API
  - Web Scraping from Wikipedia
- Perform data wrangling
  - Drop unnecessary columns
  - One Hot Encoding for categorical variables for classification
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Datasets were collected by using:
  - SpaceX REST API



  - Web Scraping from Wikipedia



Data Collection – SpaceX API

1. Request response

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

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

2. Convert to JSON and Normalize

```
# Use json_normalize meethod to convert the
data = pd.json_normalize(response.json())
```

3. Transform Data

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

the list has now been update

```
BoosterVersion[0:5]
```

```
['Falcon 1', 'Falcon 1', '
```

we can apply the rest of the fu

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

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

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

4. Create Dictionary and convert to Dataframe

```
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}
```

Then, we need to create a Pandas data frame from the dictionary launc

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

5. Filter Dataframe

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']
```


Data Collection API Lab.ipynb

Data Collection - Scraping

1. Request response

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

2. Extract data with BeautifulSoup as an object

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

3. Find required table in BeautifulSoup object

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

Starting from the third table is our target tab

```
# Let's print the third table and chec
first_launch_table = html_tables[2]
print(first_launch_table)
```

4. Set columns' name

```
column_names = []
for elem in first_launch_table.find_all('th'):
    if extract_column_from_header(elem) is not None and len(extract_column_from_header(elem)) > 0:
        column_names.append(extract_column_from_header(elem))
# 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 colu
```

5. Create 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
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. Add 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:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
```

7. Convert dictionary to dataframe

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

8. Export data to csv file

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


[Data Collection with Web Scraping lab.ipynb]

Data Wrangling

- There are multiple cases of landing outcome which success or failed:
  - Success:
    - True ASDS
    - True RTLS
    - True Ocean
  - Failed:
    - None None
    - False ASDS
    - False Ocean
    - None ASDS
    - False RTLS
- We categorized these cases into 2 categorical variables as 1 and 0 and assign to Class Column

Data Wrangling

1. Calculate launches number for each site

```
# Apply value_counts() on column
df['LaunchSite'].value_counts()
```

```
CCAFS SLC 40     55
KSC LC 39A       22
VAFB SLC 4E      13
```

2. Calculate the number and occurrence of each orbit

```
# Apply value_counts on Orb
df['Orbit'].value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
```

3. Calculate number of each landing outcome

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
```

3. Categorize landing outcome and assign to new column

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

This variable will represent the classificatio did not land successfully; one means the fi

```
df['Class']=landing_class
df[['Class']].head(8)
```

5. Calculate success rate

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

```
0.6666666666666666
```


Data Wrangling

EDA with Data Visualization

- Scatter Plot – to show the relationship between variables
  - Flight Number vs. Payload Mass
  - Flight Number vs. Launch Site
  - Payload vs. Launch Site
  - Orbit vs. Flight Number
  - Payload vs. Orbit Type
  - Orbit vs. Payload Mass
- Bar Graph – to show the relationship between the numerical and categorical variables
  - Success rate vs. Orbit
- Line Graph – to show the trend of variable which respects to time
  - Success rate vs. Year

[EDA with Visualization lab.ipynb]()

EDA with SQL

- We performed SQL queries to gather and understand data from dataset:
  - 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 v.1.1
  - 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 records which will display the month names, failure landing outcomes in drone ship, booster versions, launch site for the months in year 2015
  - Rank the count of successful landing outcomes between date 2010-06-04 and 2017-03-20 in descending order

EDA with SQL lab.ipynb

Build an Interactive Map with Folium

- Folium map object is a map centered on NASA Johnson Space Center at Houson, Texas
    - Red circle at NASA Johnson Space Center's coordinate with label showing its name
        - folium.Circle, folium.map.Marker
    - Red circles at each launch site coordinates with label showing launch site name
        - folium.Circle, folium.map.Marker, folium.plugins.MarkerCluster
    - Grouping of points in a cluster to show different information of the same coordinates
        - folium.plugins.MarkerCluster
    - Markers to show success and failed landing
        - folium.map.Marker, folium.icon
    - Markers to show distance between launch site to key locations and plot a line graph
        - folium.map.Marker, folium.PolyLine, folium.features.DivIcon
- These objects are crated to see the characteristic of attributes by visualization in interactive map

[Interactive Visual Analytics with Folium.ipynb]

Build a Dashboard with Plotly Dash

- Dashboard contains filtered dropdown and rangeslider with pie chart and scatter plot
  - Dropdown filter – to select launch site
    - dash_core_components.Dropdown
  - Rangeslider – to select range of payload mass
    - dash_core_components.RangeSlider
  - Pie chart – to show the number and ratio of success and failed landing for chosen launch sites
    - plotly.express.pie
  - Scatter plot – to show the relationship between success and payload mass
    - plotly.express.scatter

[spacex_dash_py.py]

Predictive Analysis (Classification)

- Data preparation
  - Load dataset
  - Normalize data
  - Split data into training and test datasets
- Model preparation
  - Selection Machine Learning Algorithms
  - Set param_grid of each algorithm for GridSearchCV
- Model evaluation
  - Get the best parameter for each model
  - Calculate accuracy of test dataset for each model
  - Plot confusion matrix for each model
- Model comparison
  - Compare the test accuracy of each model
  - Compare the confusion matrix of each model

Machine Learning Prediction.ipynb

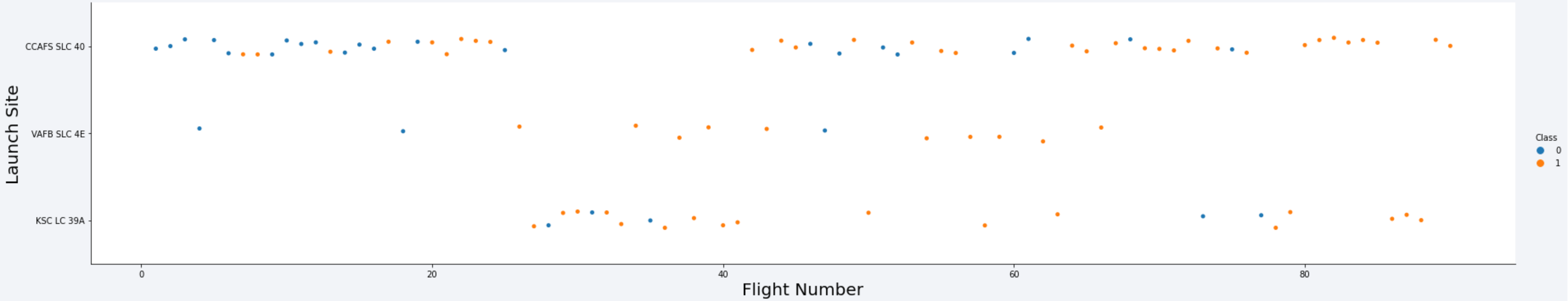
Results

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

Section 2

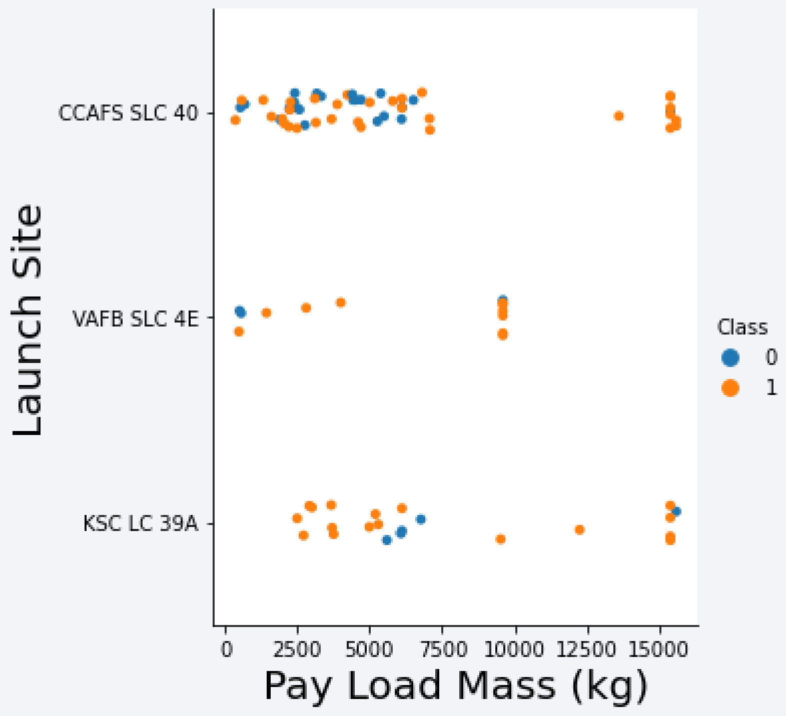
Insights drawn from EDA

Flight Number vs. Launch Site



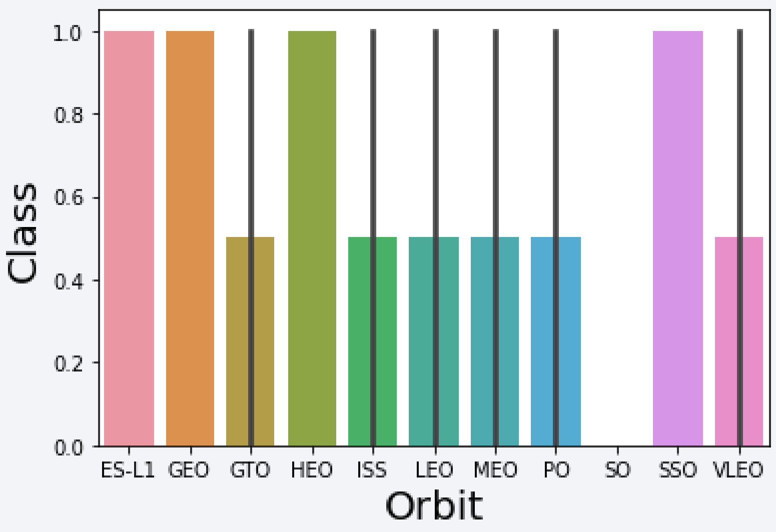
- For each site, the later flight number is, the more success rate increase

Payload vs. Launch Site



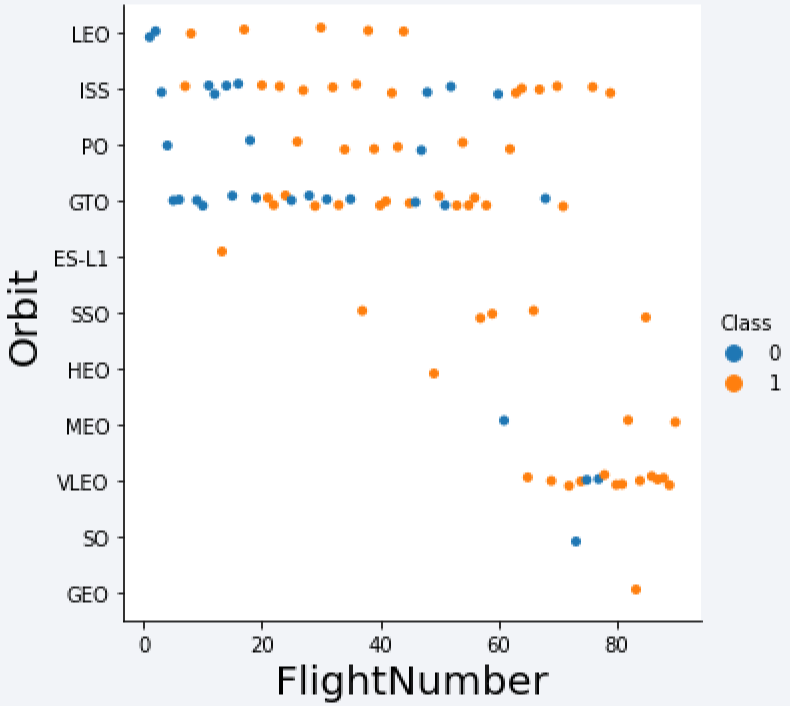
- More payload seems to have more success rate

Success Rate vs. Orbit Type



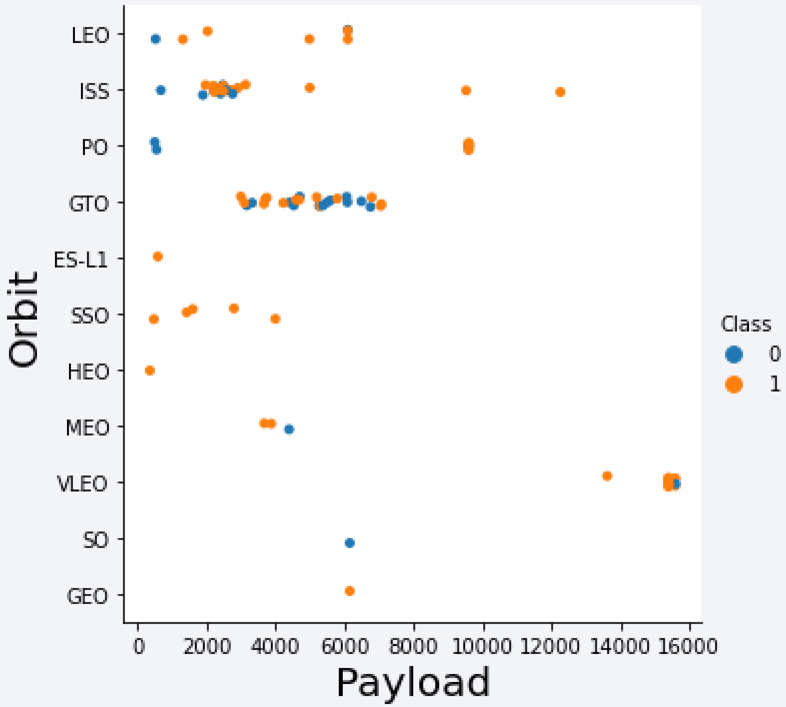
- The bar chart show the performance of each orbit.
- ES-L1, GEO, HEO, and SSO has 100% success rate

Flight Number vs. Orbit Type



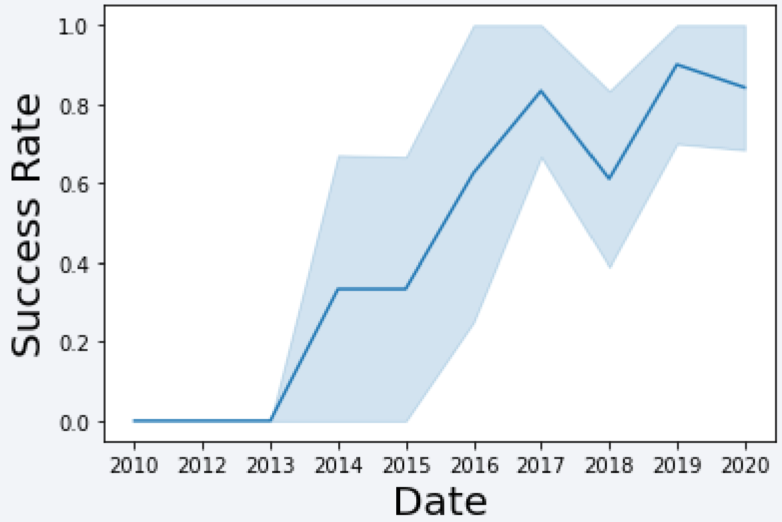
- For each orbit, the earlier flight seems to have lower success rate
- Only GTO orbit still has low success rate compared to others

Payload vs. Orbit Type



- Except GRO orbit, heavier payload seems to have more success rate than lighter weight

Launch Success Yearly Trend



- The first success landing started in 2013.
- Success rate kept increasing until 2017

All Launch Site Names

```
%%sql
SELECT DISTINCT
    LAUNCH_SITE
FROM
    SPACE_X
```

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

- SELECT DISTINCT allows to remove duplication of the field LAUNCH_SITE

Launch Site Names Begin with 'CCA'

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

- WHERE clause contains the LAUNCH_SITE field with the condition LIKE which contains CCA at the beginning of the value

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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)
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-12	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170	GTO	SES	Success	No attempt

Total Payload Mass

```
%%sql
SELECT
    SUM(payload_mass__kg_)
FROM
    SPACE_X
WHERE
    payload LIKE '%CRS%'
```

1
56479

- SUM the value in field payload_mass__kg_ which has the condition in WHERE clause: payload contains CRS in the value

Average Payload Mass by F9 v1.1

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

1
3226

- Use AVG to calculate the average value of the field payload_mass__kg_
- Use WHERE clause condition:
  - Booster_version contains value F9 v1.1

First Successful Ground Landing Date

```
%%sql
SELECT
    MIN(DATE)
FROM
    SPACE_X
WHERE
    landing__outcome LIKE '%Success%'
```

1
2016-06-05

- Find the dates of the first successful landing outcome on ground pad
- Present your query result with a short explanation here

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql
SELECT
    booster_version
FROM
    SPACE_X
WHERE
    landing__outcome = 'Success (drone ship)'
    AND payload_mass__kg_ > 4000
    AND payload_mass__kg_ < 6000
```

booster_version
F9 FT B1022
F9 FT B1031.2

- Select booster version with WHERE clause condition with landing outcome and payload mass

Total Number of Successful and Failure Mission Outcomes

```
%%sql
SELECT
    COUNT(CASE WHEN UPPER(mission_outcome) LIKE '%SUCCESS%' THEN mission_outcome END) AS total_success,
    COUNT(CASE WHEN UPPER(mission_outcome) LIKE '%FAIL%' THEN mission_outcome END) AS total_failure
FROM
    SPACE_X
```

total_success	total_failure
45	0

- Select count mission outcome with condition in the count statement

Boosters Carried Maximum Payload

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

booster_version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1058.3
F9 B5 B1060.2

- Select distinct booster version with the condition that these booster versions' payload mass is the max value

2015 Launch Records

```
%%sql
SELECT DISTINCT
    date,
    landing__outcome,
    booster_version,
    launch_site
FROM
    SPACE_X
WHERE
    UPPER(landing__outcome) LIKE '%FAIL%'
    AND YEAR(DATE(date)) = 2015
```

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

- Select the landing outcome information with the condition fail and year 2015

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

```
%%sql
SELECT DISTINCT
    landing__outcome,
    COUNT(landing__outcome) AS total_outcome
FROM
    SPACE_X
WHERE
    DATE(date) BETWEEN DATE('2010-06-04') AND DATE('2017-03-20')
GROUP BY landing__outcome
ORDER BY COUNT(landing__outcome) DESC
```

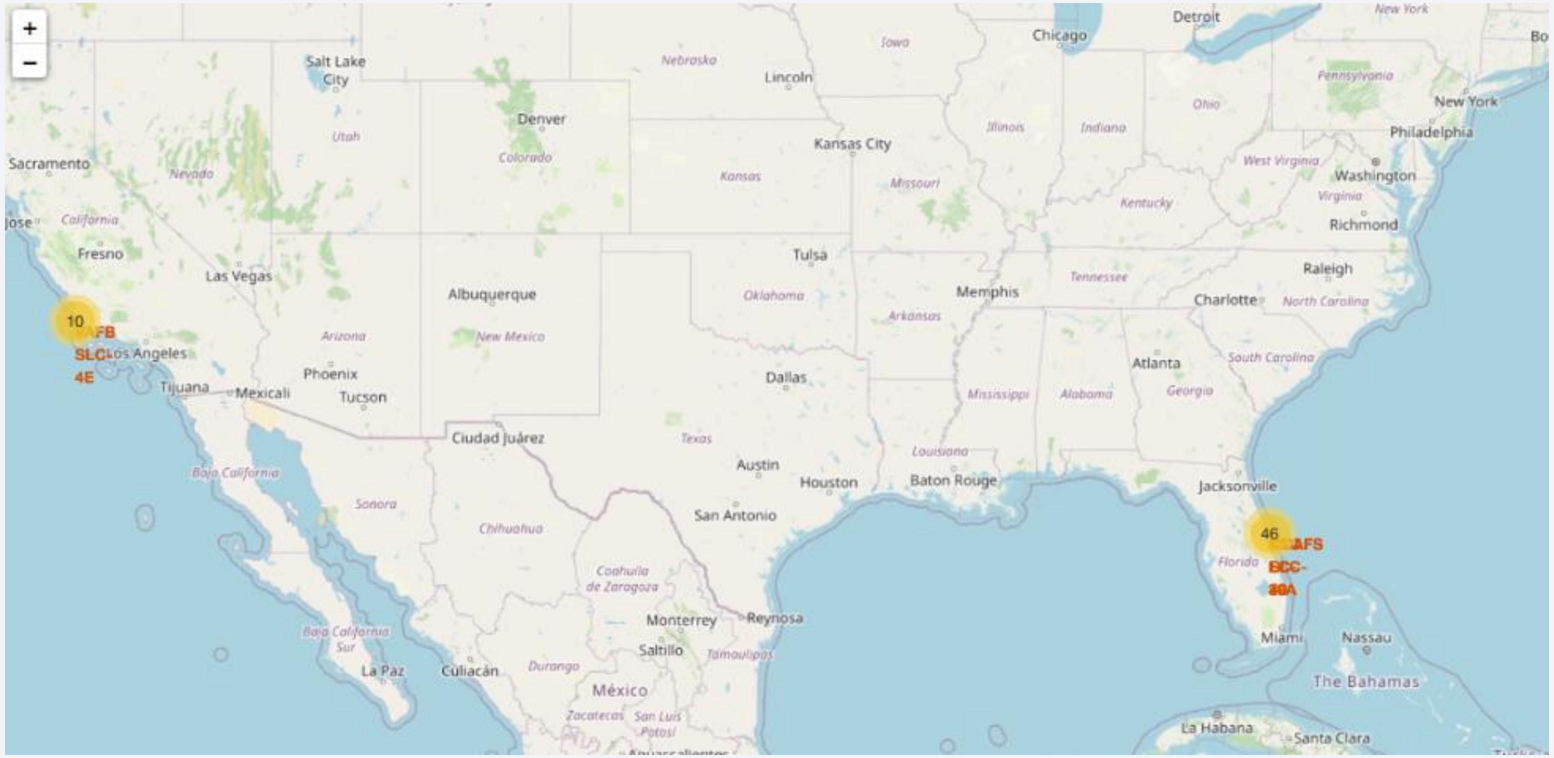
landing__outcome	total_outcome
No attempt	7
Failure (drone ship)	2
Success (drone ship)	2
Success (ground pad)	2
Controlled (ocean)	1
Failure (parachute)	1

- Select landing outcome and count each of their record with condition date range and sort with the counting of each landing outcome descending

Section 3

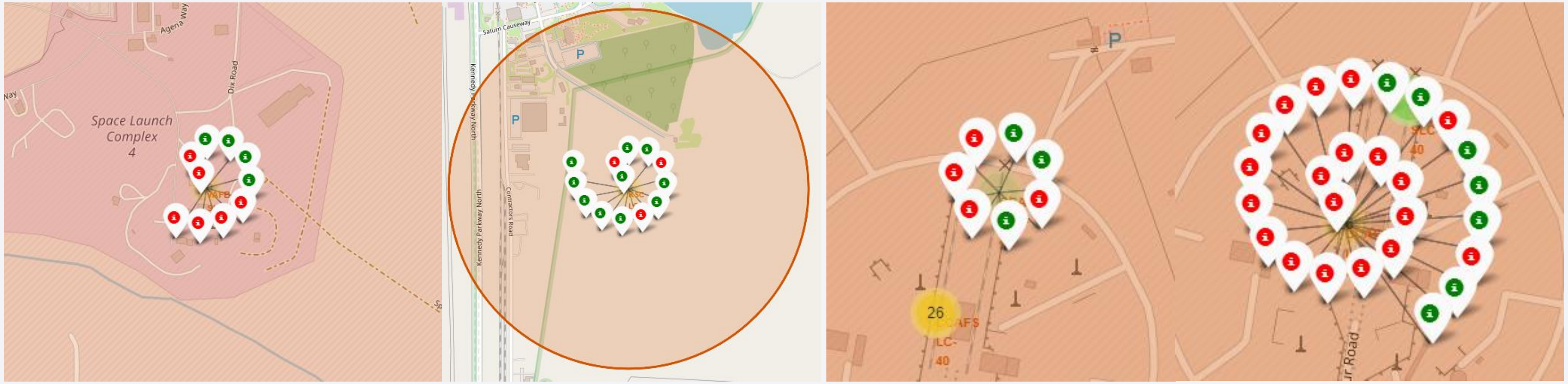
Launch Sites Proximities Analysis

Folium Map – Ground stations



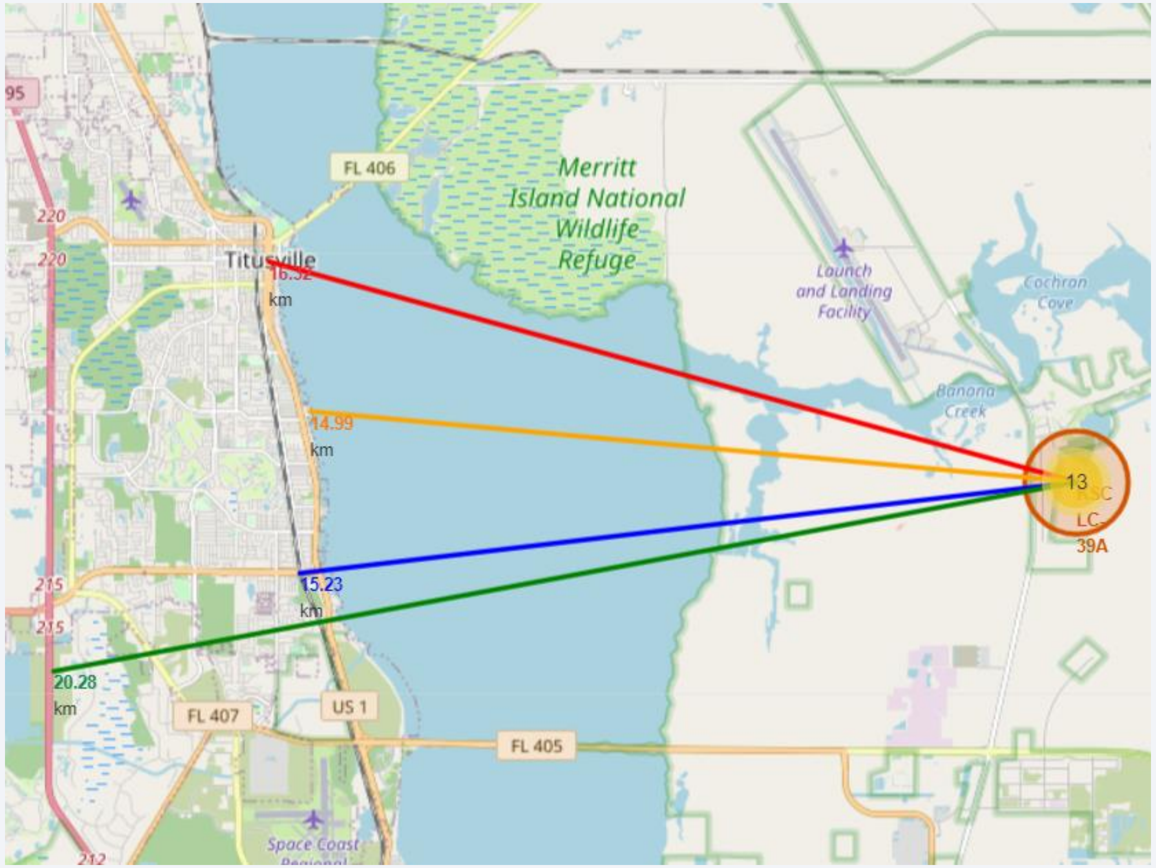
- We see that SpaceX's launch sites are located on the coast of United States

Folium Map – Color Labeled Markers



- Each of the markers represents the launches
  - Green marker: Successful launch
  - Red marker: failed launch
- The CCAFS SLC-40, which 2 locations on the right, has low success rate

Folium Map – Distance between CCAFS SLC-40 and its proximities

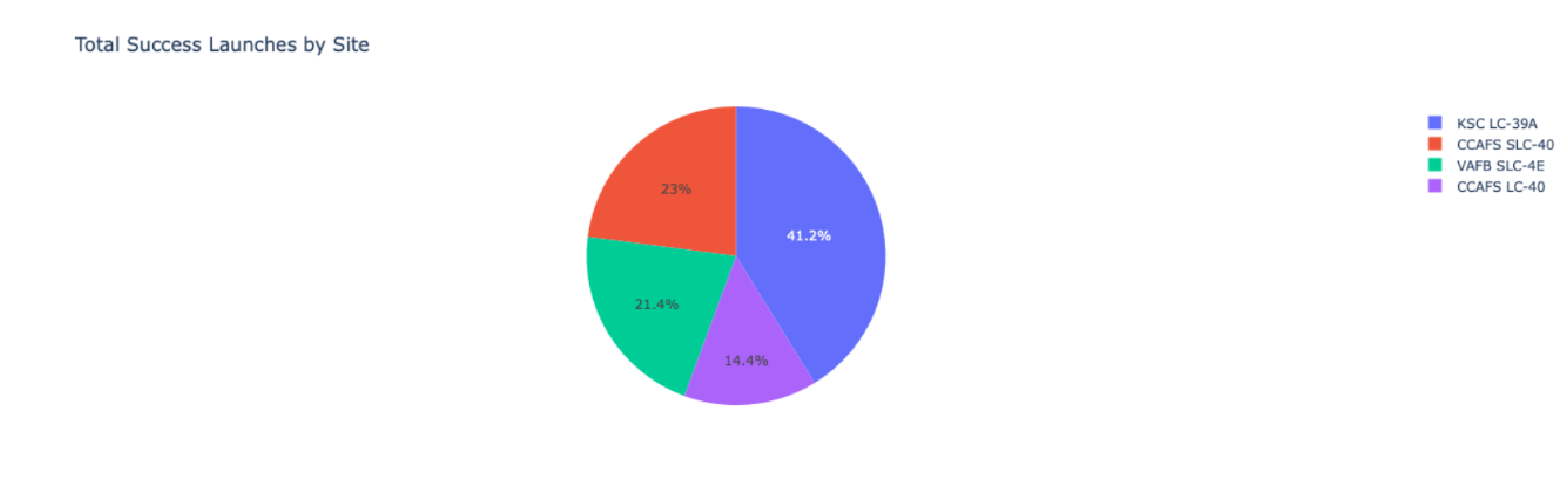


- From the map, launch site KSC LC-39A
  - Close to railway
  - Close to highway
  - Close to coastline
  - Close to city (Titusville)

Section 4

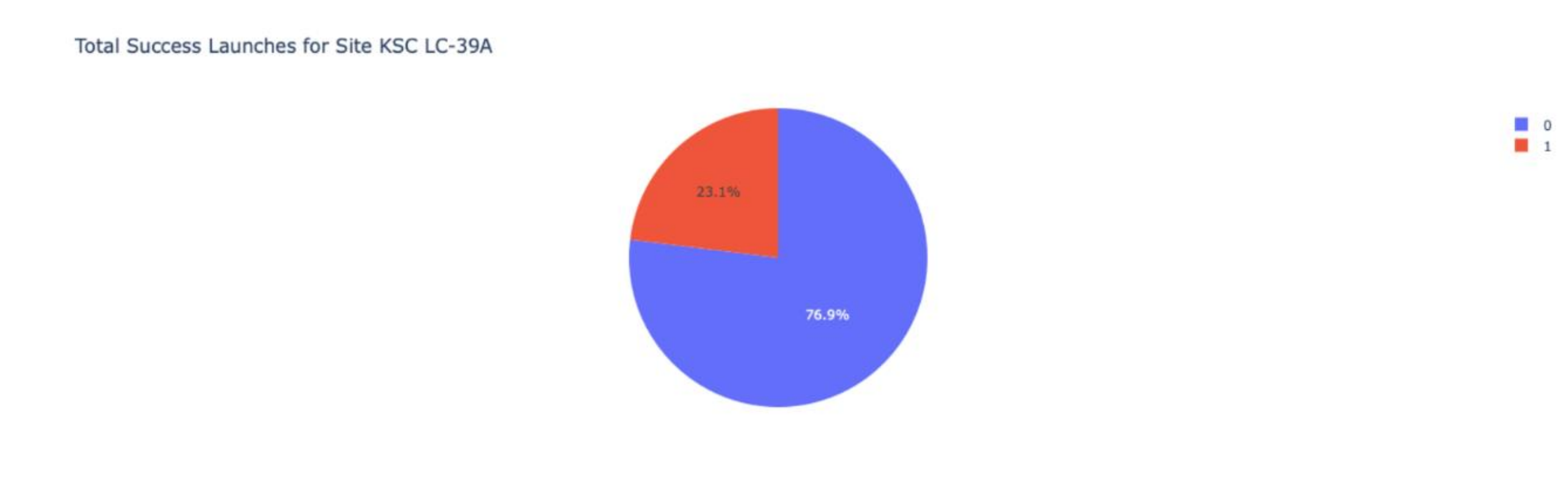
Build a Dashboard with Plotly Dash

Launch success count for all sites



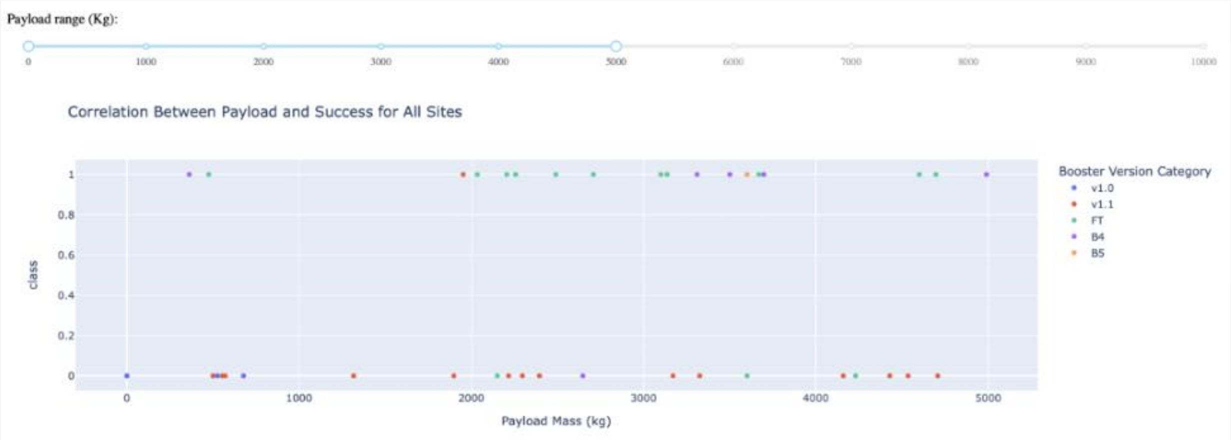
- Among 4 launch sites, KSC LC-39A has the most success launches with 41.2% of total success launches

Launch site with highest launch success ratio



- KSC LC-39A has the highest success ratio with 76.9% (10 success launches, 3 failed launches)

Payload Mass vs. Launch Outcome for all sites



- The highest success rate is on the range of payload mass between 2000 and 5500 kg.

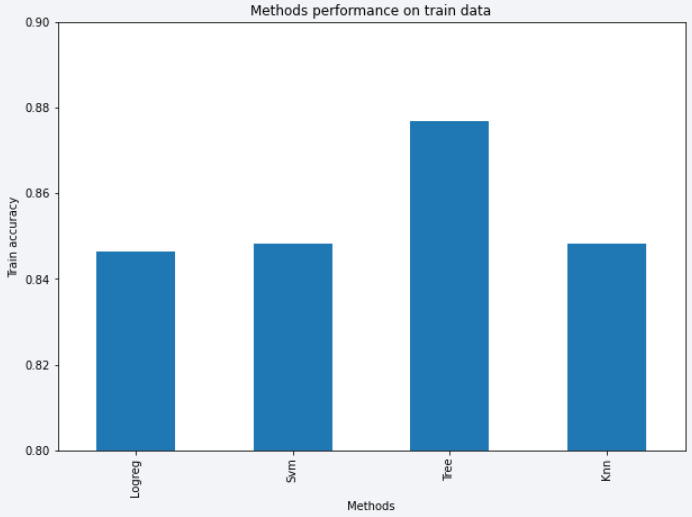
Section 5

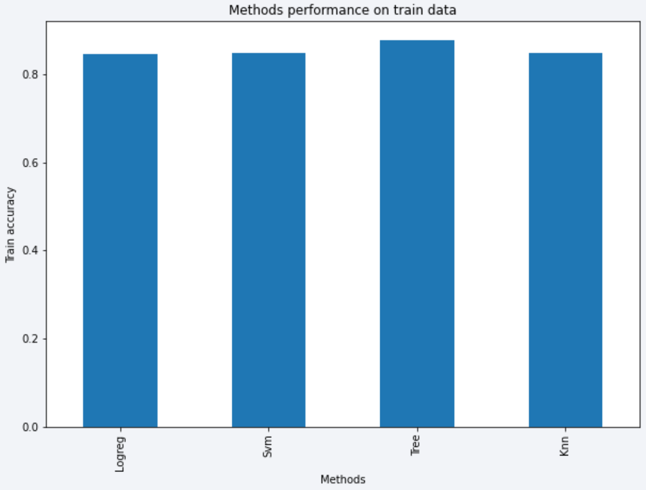
Predictive Analysis (Classification)

Classification Accuracy

- For test dataset, all the 4 models has the same accuracy which is 0.833
- To choose the best model, Decision Tree would be the selected one as it has the highest accuracy for training dataset

	Accuracy Train	Accuracy Test
Logreg	0.846429	0.833333
Svm	0.848214	0.833333
Tree	0.876786	0.833333
Knn	0.848214	0.833333





Confusion Matrix

- As in previous slide, every models has the same accuracy. All of them also has the same output for confusion matrix
- The accuracy 0.833 has some error from the FP which is the case that the model predicts as land for the did not land case

Conclusions

- Most of launch sites are in proximity to the Equator line and in very close proximity to the coast
- The orbits that has the best success rates are GEO, HEO, SSO, and ES-L1
- Success rate is increasing every year
- Payload mass has an influence for success landing. For most of the orbits, the lighter weight perform better than the heavy weight.
- The best model is Decision Tree for this dataset.

Thank you!