Winning Space Race with Data Science

Raymond
2021 Dec

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection
  - Data Wrangling
  - Exploratory Data Analysis (Data Visualisation and SQL)
  - Interactive Map (Folium)
  - Dashboard (Ploty Dash)
  - Classification (Predictive Analysis)
- Summary of all 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 one can accurately predict the likelihood of the first stage rocket landing successfully, the cost of a launch can be determined.
- Problems you want to find answers
  - What factors influences the success rate of a rocket landing?
  - How does each factor affect the landing success rate for a certain rocket type?
  - What conditions are required to achieve the highest landing success rate?

Section 1

Methodology

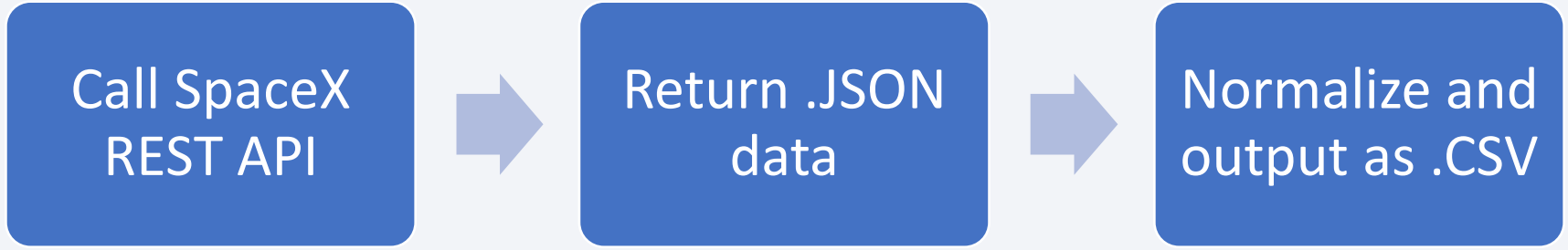
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web Scraping (Falcon 9 historical launch records from a Wikipedia page)
- Perform data wrangling
  - Transform data collected based on landing outcomes
  - Drop irrelevant columns
- 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

1. SpaceX REST API
   - A series of GET request was performing to return information from API in json format
   - Data is subsequently normalized using Pandas to output a CSV file



2. Wikipedia Falcon 9 Launch data Web Scraping
   - Extract Data using beautiful soup
   - Normalize and output as CSV file

Data Collection – SpaceX API

1. Get response from SpaceX API:

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
print(response.content)
```

2. Convert response as a dataframe:

```
data1 = response.json()
data= pd.json_normalize(data1)
data
```

3. Use Helper function to extract relevant :

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

4. Construct 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. Clean up unwanted data:

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data[data.BoosterVersion == 'Falcon 9']
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
```

6. Address missing values

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

7. Output flat file (.CSV)

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

https://github.com/raymondlyz/Applied-Data-Science-Capstone-Project/blob/master/1%20Data%20Collection%20API%20Lab.ipynb

Data Collection - Scraping

1. Define HTML Source

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

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

2. Create Beautiful Soup Object

```
soup = BeautifulSoup(response, 'html.parser')
```

3. Find tables

```
html_tables = soup.find_all("table")
print(html_tables)
```

4. Extract Column Header

```
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`) int
o a list called column_names
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

5. Create dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelevant 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. Append data to Dictionary keys

```
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 dictionary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict["Flight No."].append(Flight_number)
            #print(Flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            #print(date)
            launch_dict["Date"].append(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            #print(time)
            launch_dict["Time"].append(time)

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            #print(bv)
            launch_dict["Version Booster"].append(bv)

            # Launch Site
            # TODO: Append the bv into launch_dict with key `Launch Site`
            launch_site = row[2].a.string
            #print(launch_site)
            launch_dict["Launch site"].append(launch_site)

            # Payload
            # TODO: Append the payload into launch_dict with key `Payload`
            payload = row[3].a.string
            #print(payload)
            launch_dict["Payload"].append(payload)

            # Payload Mass
            # TODO: Append the payload_mass into launch_dict with key `Payload mass`
            payload_mass = get_mass(row[4])
            #print(payload)
            launch_dict["Payload mass"].append(payload_mass)

            # Orbit
            # TODO: Append the orbit into launch_dict with key `Orbit`
            orbit = row[5].a.string
            #print(orbit)
            launch_dict["Orbit"].append(orbit)

            # Customer
            # TODO: Append the customer into launch_dict with key `Customer`
            #customer = row[6].a.string
            #https://www.coursera.org/learn/applied-data-science-capstone/discussions/weeks/1/threads/PS-KAvVLEeuofxI2H0ASIw/re
plies/04136gMdEeyxWBK7UBgFow
            customer = row[6].text.strip()
            #print(customer)
            launch_dict["Customer"].append(customer)

            # Launch outcome
            # TODO: Append the launch_outcome into launch_dict with key `Launch outcome`
            launch_outcome = list(row[7].strings)[0]
            #print(launch_outcome)
            launch_dict["Launch outcome"].append(launch_outcome)

            # Booster landing
            # TODO: Append the launch_outcome into launch_dict with key `Booster landing`
            booster_landing = landing_status(row[8])
            #print(booster_landing)
            launch_dict["Booster landing"].append(booster_landing)
```

7. Convert Dictionary into Dataframe

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

headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

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

8. Output flat file (.csv)

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

https://github.com/raymondlyz/Applied-Data-Science-Capstone-Project/blob/master/2%20Data%20Collection%20with%20Web%20Scraping.ipynb

Data Wrangling

1. Perform exploratory Data Analysis and determine Training Labels
   - Exploratory Data Analysis
   - Determine Training Labels
2. Calculate the number of launches on each site
3. Calculate the number and occurrences of each orbit
4. Calculate the number and occurrence of mission outcome per orbit type

https://github.com/raymondlyz/Applied-Data-Science-Capstone-Project/blob/master/3%20EDA%20lab.ipynb

EDA with Data Visualization

Scatter Plot of:

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

Bar Graph of:

- Success rate of each Orbit type

Line Graph of:

- Launch Success Yearly trend

https://github.com/raymondlyz/Applied-Data-Science-Capstone-Project/blob/master/5%20EDA%20with%20Visualization.ipynb

EDA with SQL

- Insights gathered via SQL:
  - 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. Use a subquery
  - List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  - Rank the count of successful landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order

https://github.com/raymondlyz/Applied-Data-Science-Capstone-Project/blob/master/4%20EDA%20with%20SQL%20lab.ipynb

Build an Interactive Map with Folium

- Using Launch_Site coordinates, launch data were marked on an interactive map to aid visualization
- Launch_Outcome can also be visualized based on the classes:
  - Successful launch (Class 1) in Green
  - Failed launch (Class 0) in Red
- Distances between an identified launch site (CCAFS SLC-40) to its proximities were determined to answer some trends related to launch site location:
  - Are launch sites in close proximity to railways?
  - Are launch sites in close proximity to highways?
  - Are launch sites in close proximity to coastline?
  - Do launch sites keep certain distance away from cities?

https://github.com/raymondlyz/Applied-Data-Science-Capstone-Project/blob/master/6.2%20lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- A Dashboard provide live trending of proportion of successful launches by varying launch site when SpaceX conducts new launches which will help to improve prediction accuracy overtime
- A Scatterplot of Launch Outcome vs Payload Mass (kg) for different booster versions provide an easily observable relationship between the variables

https://raw.githubusercontent.com/raymondlyz/Applied-Data-Science-Capstone-Project/master/SpaceXPloty_Ray2.py

Predictive Analysis (Classification)

- Building the classification model
  - Load dataset into NumPy and Pandas
  - Transformation: Standardize data and reassign to variable
  - Split data to Train and Test set
  - Determine test sample count
  - Perform logistic regression object and create GridSearchCV
  - Fit dataset into GridSearchCV and train with dataset
- Evaluating and Improving the classification model
  - Check accuracy of each model
  - Get best parameter (Tuned Hyperparameters)
  - Plot Confusion Matrix
- Finding the best performing classification model
  - Determining the model with the best accuracy score

https://github.com/raymondlyz/Applied-Data-Science-Capstone-Project/blob/master/7%20Machine%20Learning%20Prediction.ipynb

Results

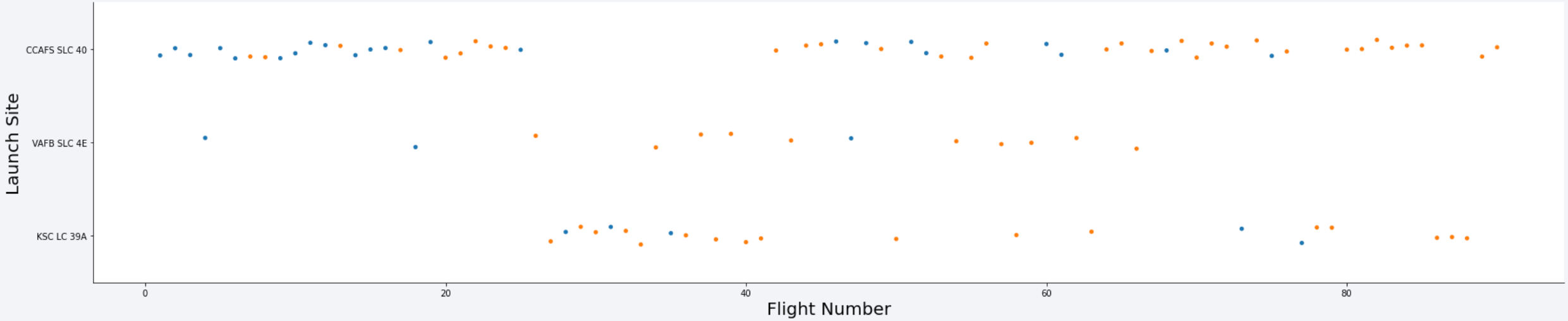
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

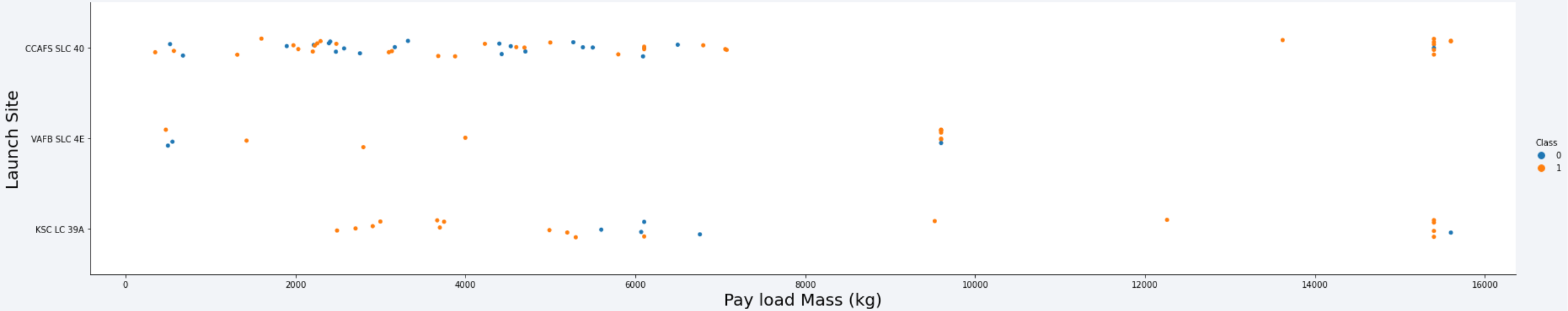
- Scatter plot of Flight Number vs. Launch Site



- General trend of higher success rate with higher flight number

Payload vs. Launch Site

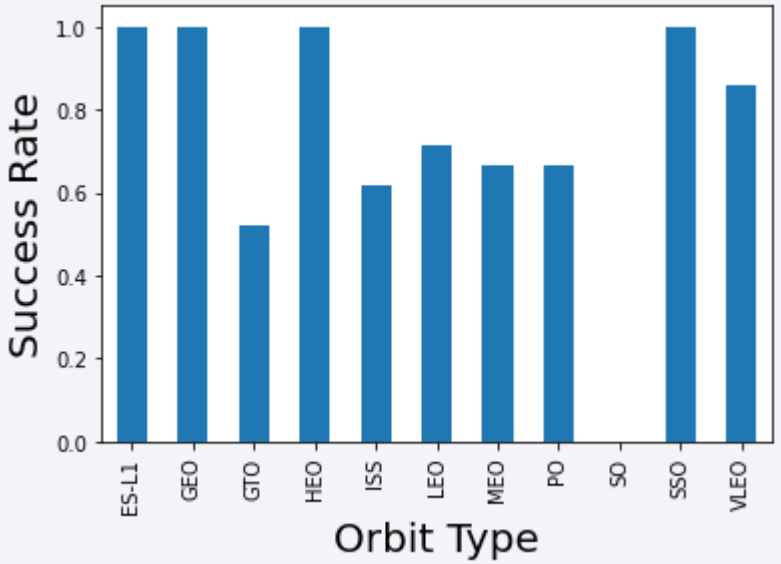
- Scatter plot of Payload vs. Launch Site

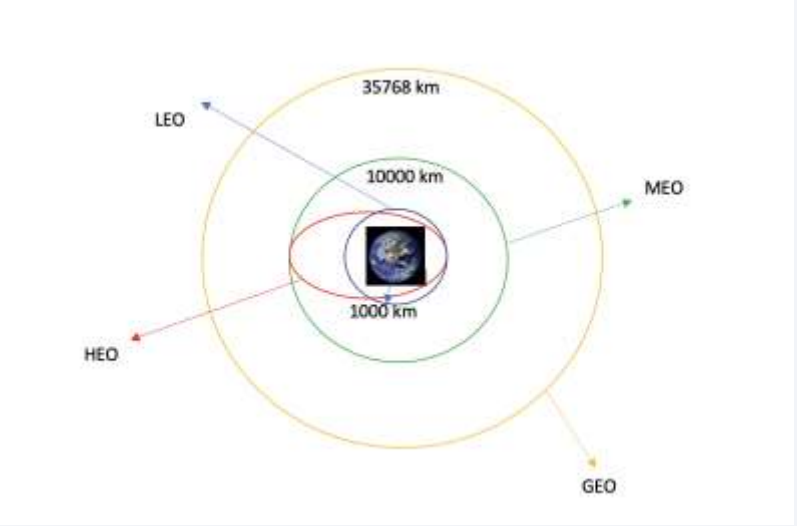


- CCAFS SLC 40: Higher success rate at greater payload mass
- No observable trend if launch site is dependent on payload mass for successful launch

Success Rate vs. Orbit Type

- Bar chart for the success rate of each orbit type

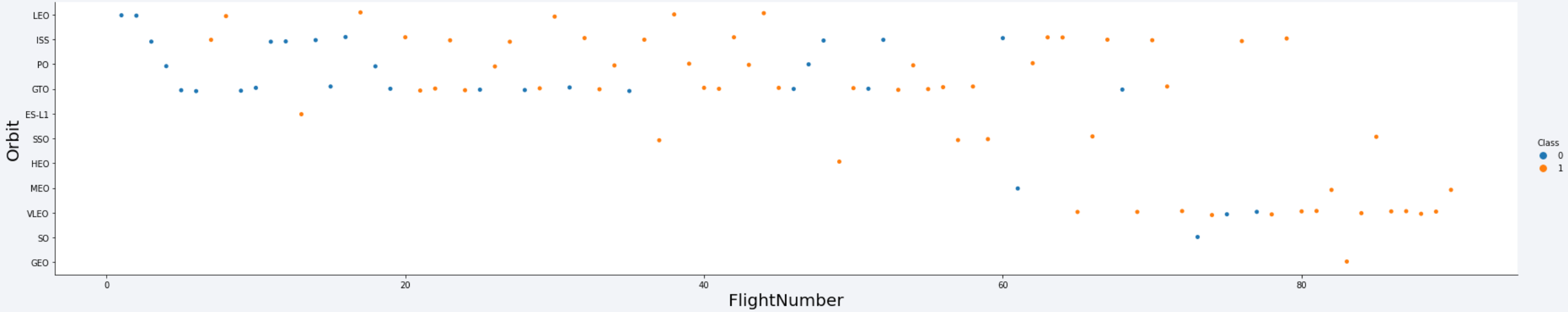




- Orbit type with highest success rate:
  - ES-L1
  - GEO
  - HEO
  - SSO

Flight Number vs. Orbit Type

- Scatter point of Flight number vs. Orbit type



- LEO success rate is related to the number of flights
- No observable relationship between GEO orbit type and the number of flights

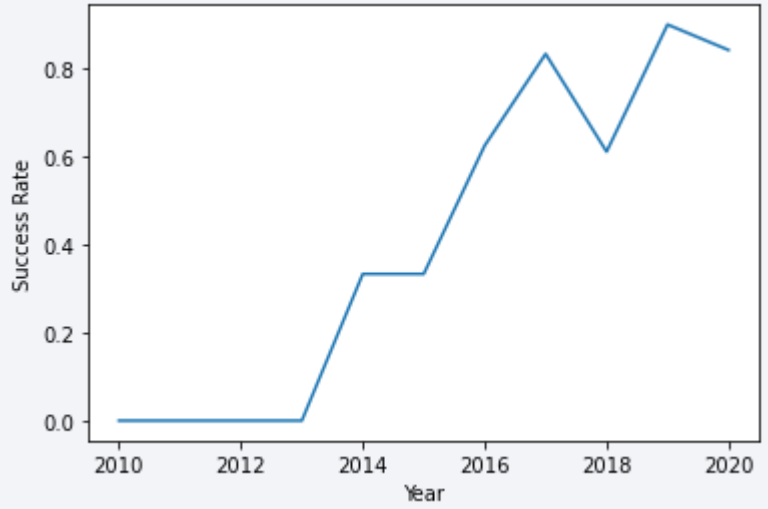
Payload vs. Orbit Type

- Scatter point of payload vs. orbit type

- Heavier payload has a general negative influence on GTO orbit
- Heavier payload has a general positive influence on LEO, ISS, PO orbit

Launch Success Yearly Trend

- Line chart of yearly average success rate



- General upward trend for success rates over the years (Unknown dip in 2018)

All Launch Site Names

- Select distinct(LAUNCH_SITE) from SPACEXTBL
  - Extract the list of unique launch site from the Table [SPACEXTBL] using SQL native “Select DISTINCT”

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

Launch Site Names Begin with 'CCA'

- Select * from SPACEXTBL where LAUNCH_SITE like 'CCA%' limit 5
  - Like ‘CCA%’ will result in returning rows with [LAUNCH_SITE] beginning with ‘CCA’
  - To restrict the number of rows returned, there can be 2 methods:
    1. Select TOP 5 *
    2. Limit 5

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

Total Payload Mass

- Select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where CUSTOMER = 'NASA (CRS)'
  - To find total Payload Mass, the sum() function is used
  - A 'where CUSTOMER = 'NASA (CRS)' is added to filter what is the total payload mass for NASA (CRS) only.

Total Payload Mass
45596

Average Payload Mass by F9 v1.1

- Select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where BOOSTER_VERSION = 'F9 v1.1'
  - To find average Payload Mass, the avg() function is used
  - A 'where BOOSTER_VERSION = 'F9 v1.1' is added to calculate average for the specified booster version only.

Average Payload Mass by F9 v1.1
2928

First Successful Ground Landing Date

- Select min(DATE) from SPACEXTBL where Landing__Outcome = 'Success (ground pad)'
  - To return only successful landing, a filter of where Landing__Outcome = 'Success (ground pad)' is applied
  - In order to return only the earliest successful landing date, a min() function is applied.

First Successful Ground Landing Date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Sql select BOOSTER_VERSION from SPACEXTBL where Landing__Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000
  - To filter the BOOSTER_VERSION returned in the results, 2 filter was applied

    1) Successful Drone Ship Landing only: Landing__Outcome = 'Success (drone ship)' **AND**

    2) Predefined payload range: PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000

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

Total Number of Successful and Failure Mission Outcomes

- Sql select count(MISSION_OUTCOME) from SPACEXTBL where MISSION_OUTCOME = 'Success' or MISSION_OUTCOME = 'Failure (in flight)'
  - Since a total count of mission is required regardless of outcome, a count is done regardless if MISSION_OUTCOME = 'Success' **or** MISSION_OUTCOME = 'Failure (in flight)'

Total Number of Successful and Failure Mission Outcomes
100

Boosters Carried Maximum Payload

- Sql select BOOSTER_VERSION from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)

  1) Determine what is the maximum Payload carried:

  select max(PAYLOAD_MASS__KG_) from SPACEXTBL

  2) Using the Maximum payload value to return a list of result of the BOOSTER_VERSION using a where statement

  where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)

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

2015 Launch Records

- Sql select DATE, booster_version, launch_site from SPACEXTBL where landing__outcome like 'Fail%' and DATE like '2015%'
  - Only Date, Booster_Version and Launch_site is required

1) Limit the search by the year 2015:
   - DATE like '2015%'
2) Limit the search based on any failed landing outcome:
   - landing__outcome like 'Fail%'

DATE	booster_version	launch_site
2015-01-10	F9 v1.1 B1012	CCAFS LC-40
2015-04-14	F9 v1.1 B1015	CCAFS LC-40

Rank Landing Outcomes [Success (ground pad)] Between 2010-06-04 and 2017-03-20

- Sql select * from SPACEXTBL where Landing__Outcome like 'Success%' and (DATE between '2010-06-04' and '2017-03-20') order by date desc
  - Zooming in on ranking for Success only:
    - Landing_Outcome like 'Success%'
  - Limit date range:
    - DATE between '2010-06-04' and '2017-03-20'
  - Rank based on date (latest on top)
    - order by date desc

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-01-14	17:54:00	F9 FT B1029.1	VAFB SLC-4E	Iridium NEXT 1	9600	Polar LEO	Iridium Communications	Success	Success (drone ship)
2016-08-14	05:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-07-18	04:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2016-05-27	21:39:00	F9 FT B1023.1	CCAFS LC-40	Thaicom 8	3100	GTO	Thaicom	Success	Success (drone ship)
2016-05-06	05:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-04-08	20:43:00	F9 FT B1021.1	CCAFS LC-40	SpaceX CRS-8	3136	LEO (ISS)	NASA (CRS)	Success	Success (drone ship)
2015-12-22	01:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)

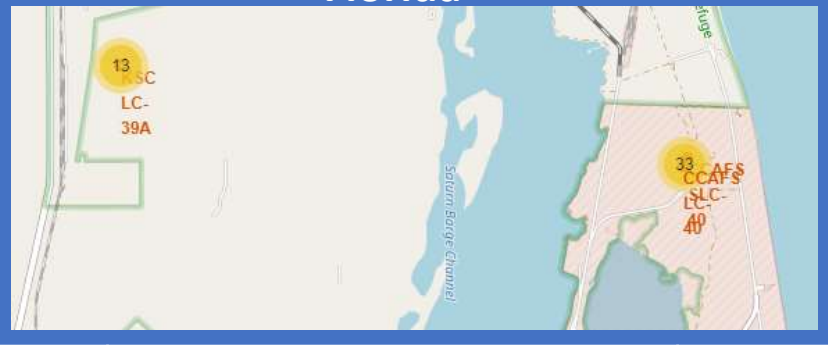
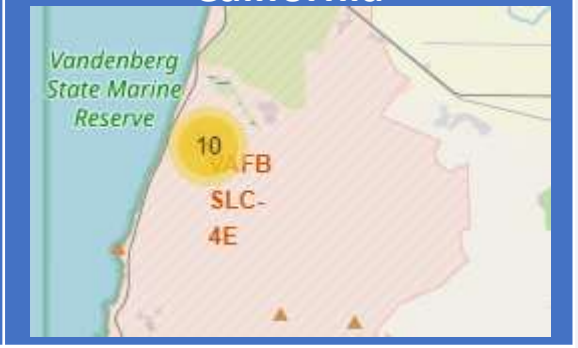
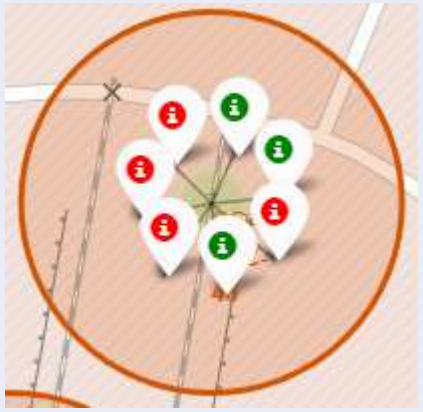
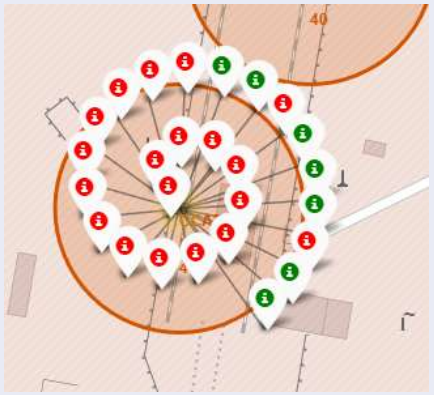
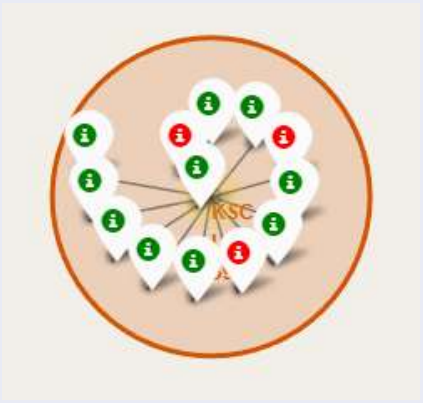
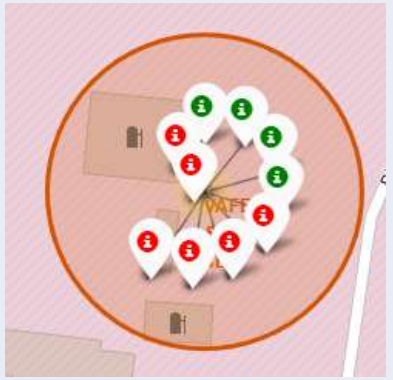
Section 4

Launch Sites Proximities Analysis

Global Map reflecting all launch sites

SpaceX launch site are in USA (Florida and California)

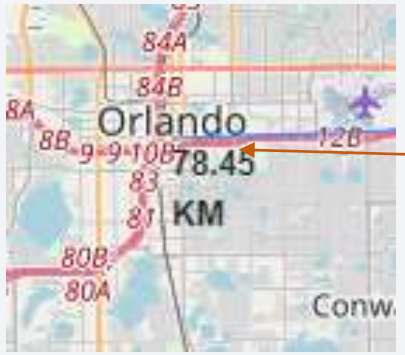
Launch Site with respective launch outcomes

<table>
<tr><th colspan="3">Florida</th><th>California</th></tr>
<tr><td colspan="3"> 
</td><td> 
</td></tr>
<tr><td>CCAFS SLC-40</td><td>CCAFS LC-40</td><td>KSC LC-39A</td><td>VAFB SLC-4E</td></tr>
<tr><td>  </td><td>  </td><td>  </td><td>  </td></tr>
</table>

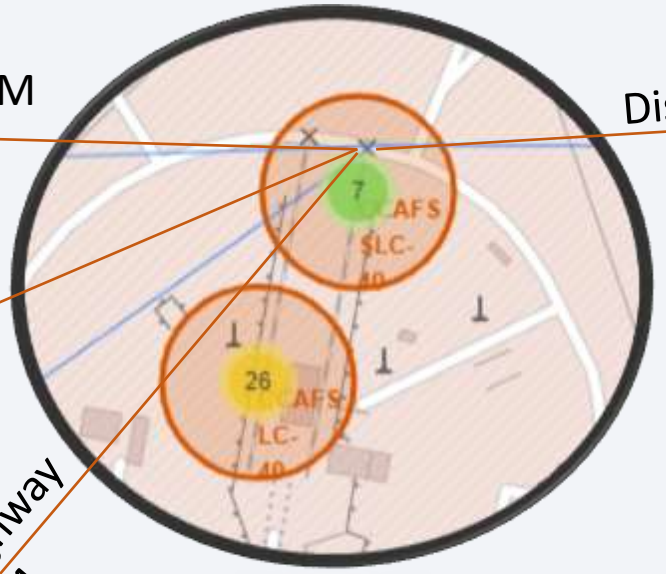
Green Marker: Successful Launches

Red Marker: Failed Launches

Explore and analyze the proximities of launch sites

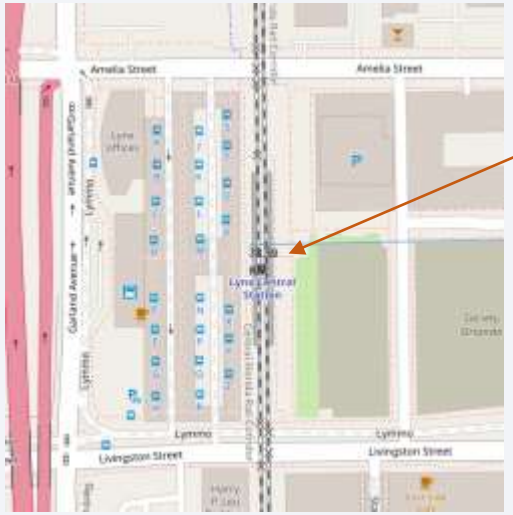


Distance to City 78.45 KM



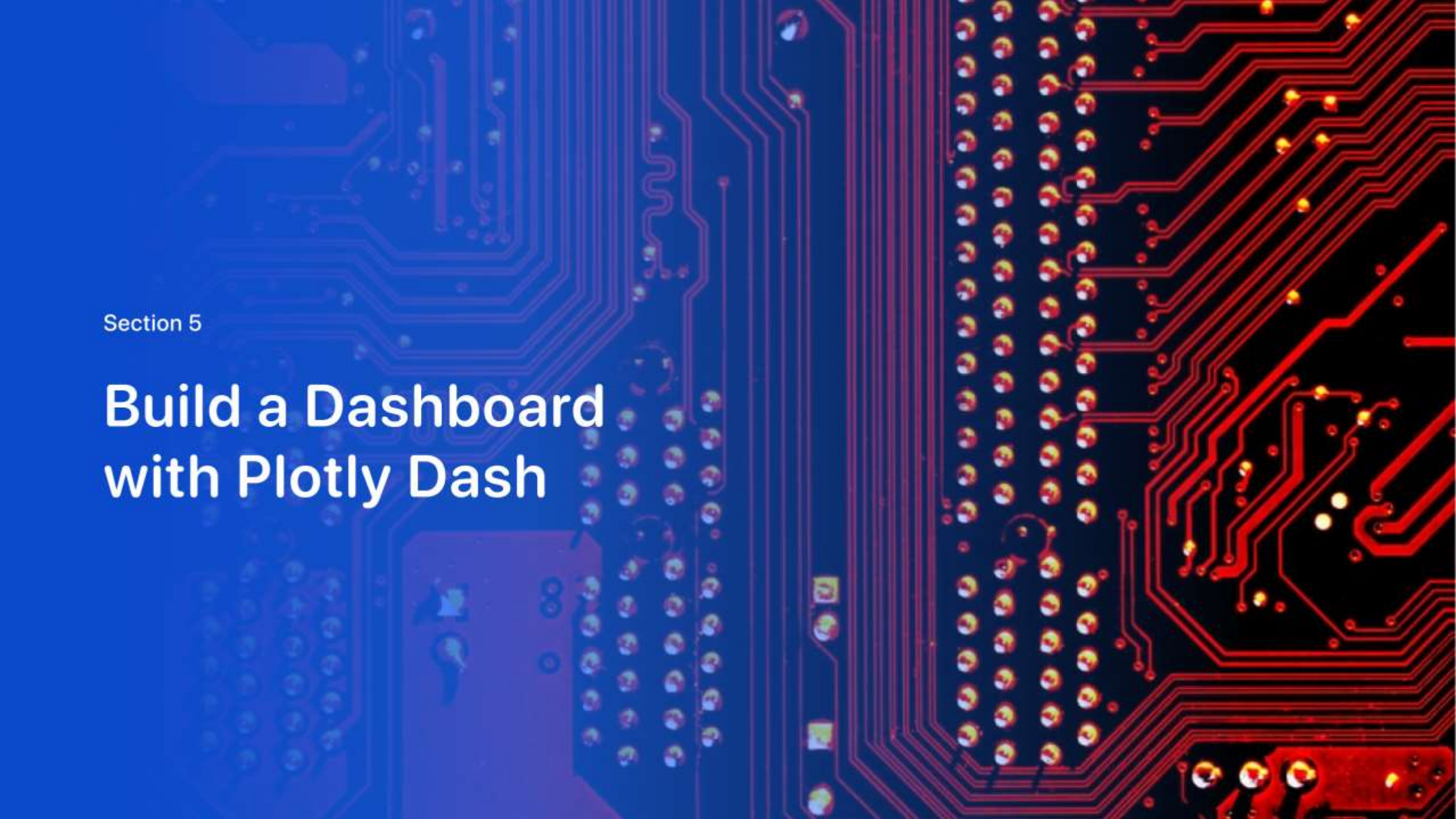
Distance to Coastline
0.90 KM

Distance to Railways 78.59 KM

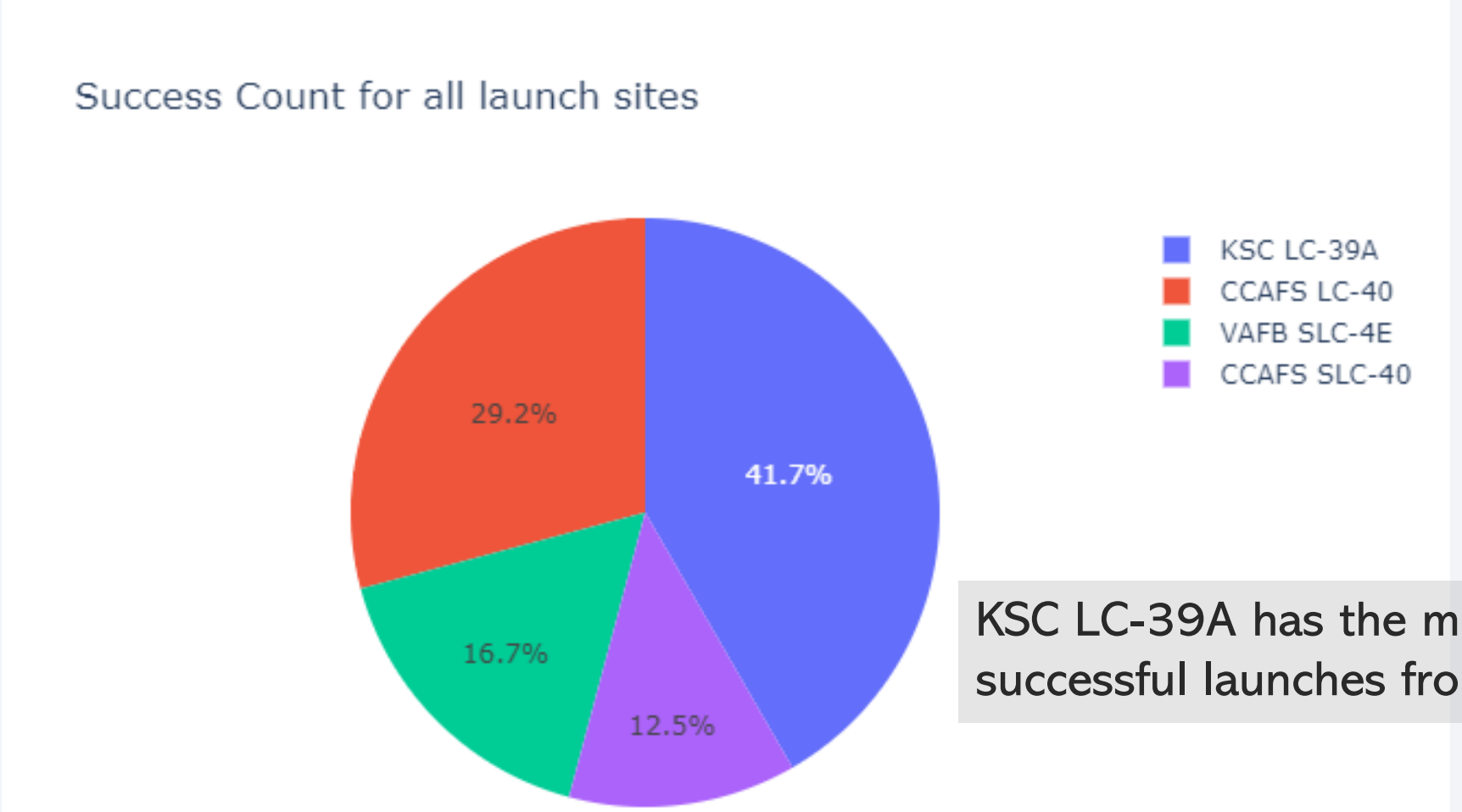


Distance to Highway
29.21 KM

- 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

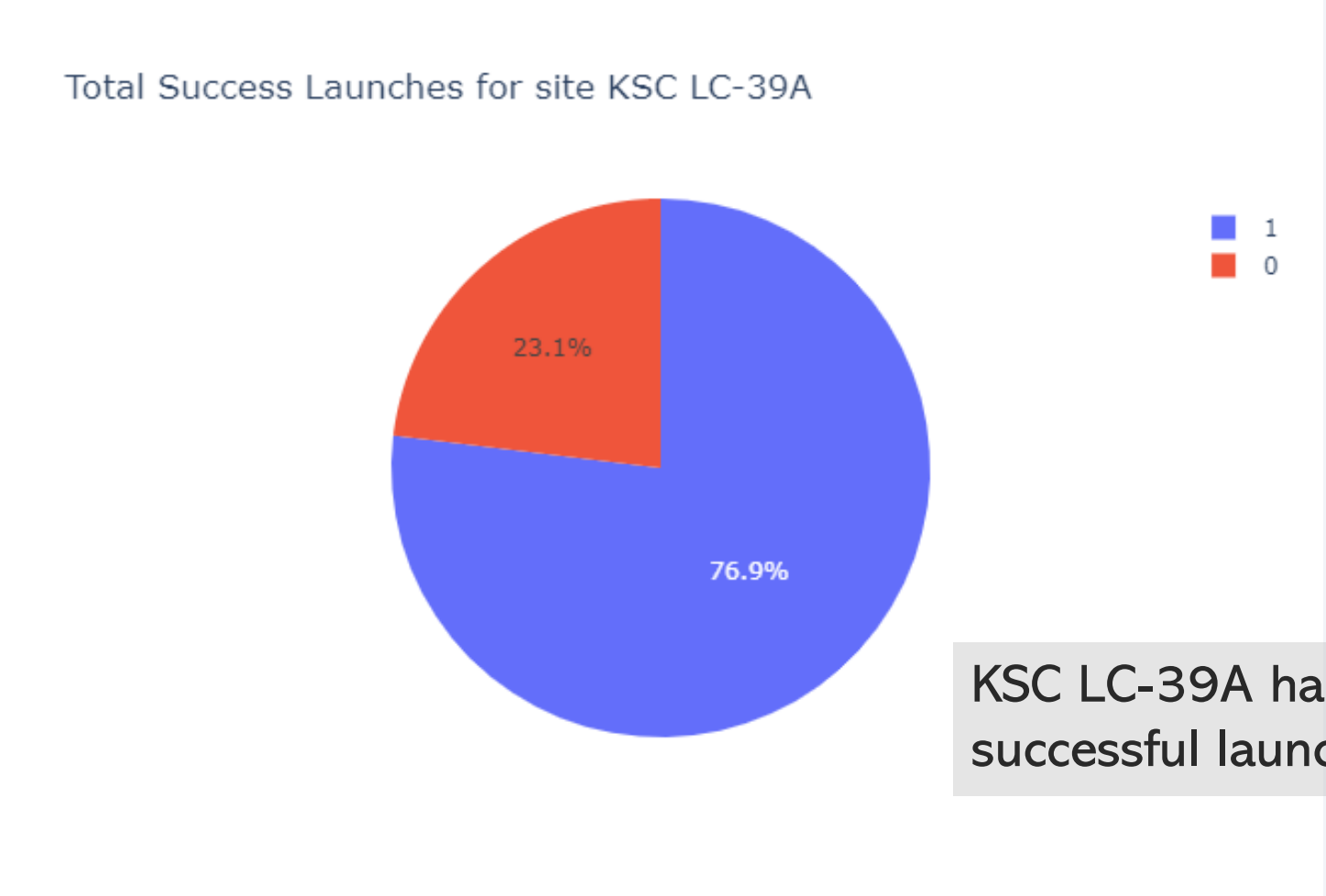

Section 5
Build a Dashboard with Plotly Dash

Dashboard: Success Percentage by Launch Sites



KSC LC-39A has the most count of successful launches from all the sites

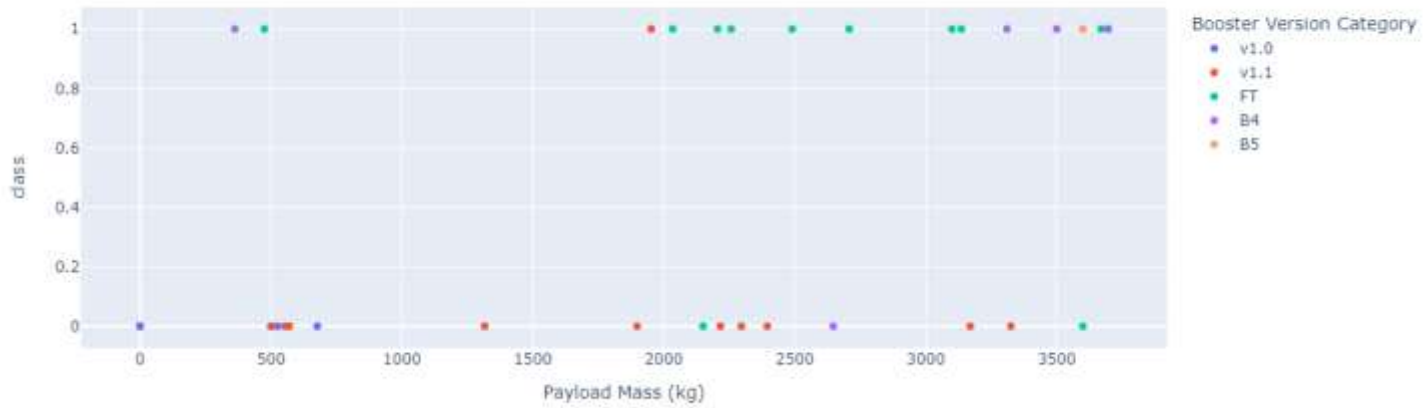
Dashboard: Break down for site with highest success count



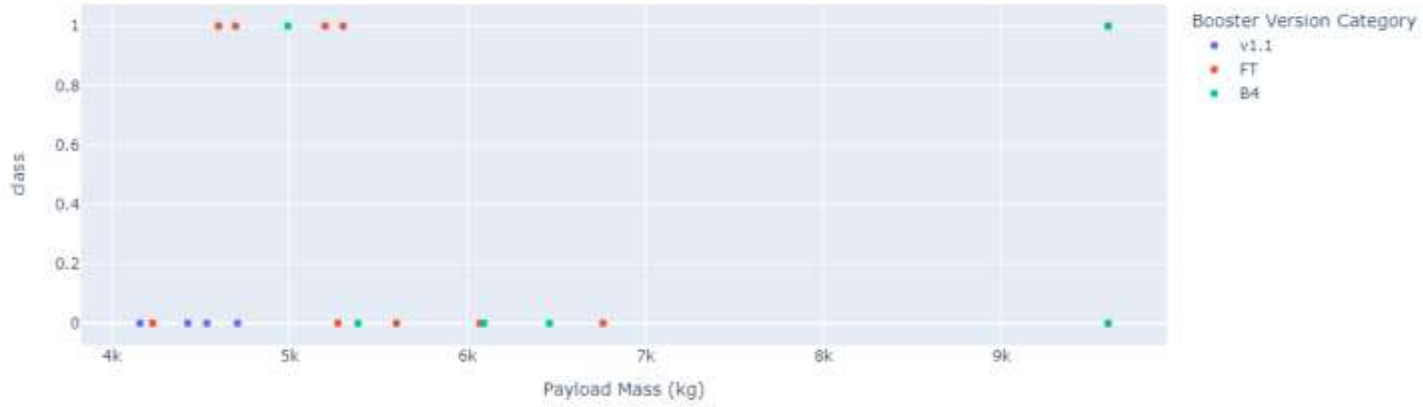
KSC LC-39A had a 76.9% successful launch rate

Dashboard: Payload vs Launch Outcome

Success count on Payload mass (0 to 4000kg) for all sites



Success count on Payload mass (4000kg and above) for all sites



1) Higher success rate at lower Payload mass
2) FT Booster Version seems to have a higher success rate

Section 6

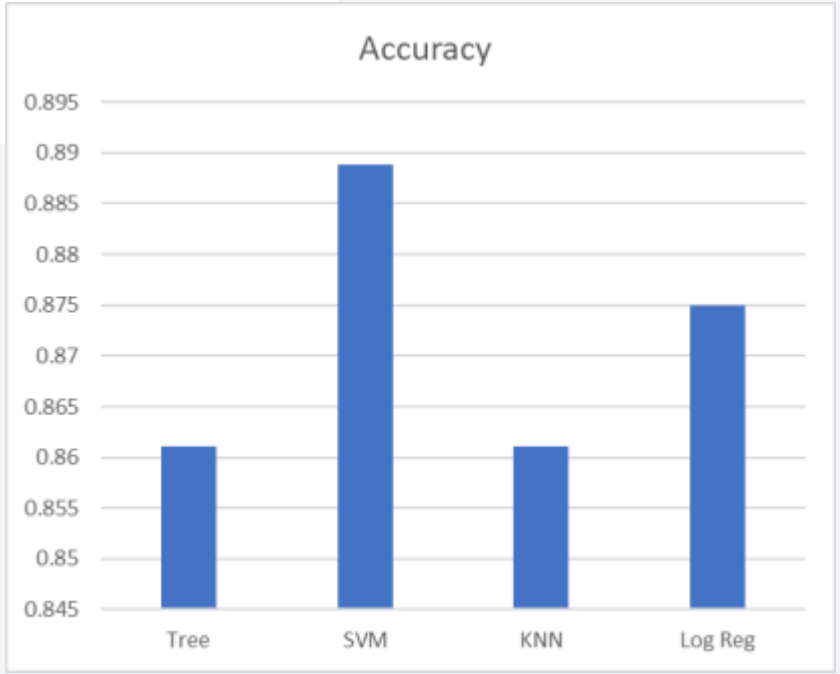
Predictive Analysis (Classification)

Classification Accuracy

```
print ("Log Reg: Best Score:" + str(logreg_cv.best_score_) + "  Accuracy " + str(logreg_cv.score(X_train, Y_train)))
print ("SVM: Best Score:" + str(svm_cv.best_score_) + "  Accuracy " + str(svm_cv.score(X_train, Y_train)))
print ("Tree: Best Score:" + str(tree_cv.best_score_) + "  Accuracy " + str(tree_cv.score(X_train, Y_train)))
print ("KNN: Best Score:" + str(knn_cv.best_score_) + "  Accuracy " + str(knn_cv.score(X_train, Y_train)))
```

```
Log Reg: Best Score:0.8464285714285713  Accuracy 0.875
SVM: Best Score:0.8482142857142856  Accuracy 0.8888888888888888
Tree: Best Score:0.8892857142857142  Accuracy 0.8611111111111112
KNN: Best Score:0.8482142857142858  Accuracy 0.8611111111111112
```

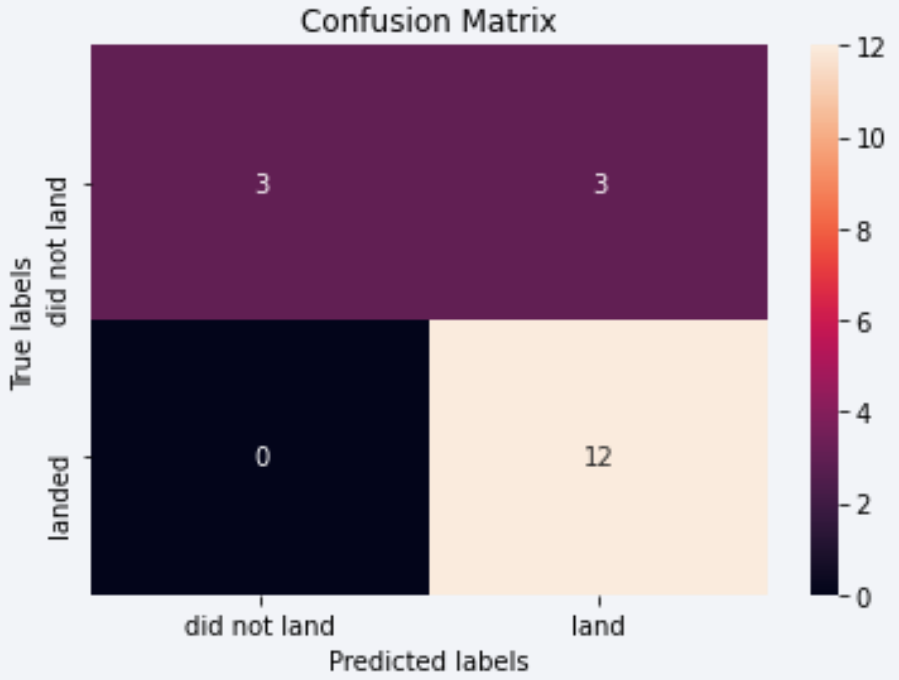
Algorithm	Best Score	Accuracy
Tree	0.889285714	0.861111111
SVM	0.848214286	0.888888889
KNN	0.848214286	0.861111111
Log Reg	0.846428571	0.875



- While all Algorithm have close accuracy range, SVM has the highest accuracy amongst all at 0.8889

Confusion Matrix for SVM

- The Matrix shows that SVM was able to predict accurately true positive of successful landing, but it still have several false positive for failure to land



Conclusions

- Orbit ES-L1, GEO, HEO and SSO has the best Success Rates
- There is a direct proportion of success launch rates as compared to the number of years
- KSC LC-39A had the most successful launches from all launch sites
- Lower weighted payload performed better than heavier ones
- SVM Classifier Algorithm is best suited for Machine Learning of this data set

Thank you!