IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Arjuman Ara Mimi
13th October, 2022

Outline

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

Executive Summary

- Summary of methodologies
  - ➢ Data Collection API with Web Scrapping
  - ➢ Data Wrangling
  - ➢ Exploratory Data Analysis with SQL
  - ➢ Exploratory Data Analysis with Data Visualization
  - ➢ Data Visualization with Folium
  - ➢ Building and Interactive Visual Analytics with Dashboard
  - ➢ Predictive Analysis(Classification)
- Summary of all results
  - ➢ Exploratory Data Analysis results
  - ➢ Interactive Analytics and Dashboard
  - ➢ Predictive Analytics Results

Introduction

- Project background and context
  - ➢ The commercial space age is here, and companies are making space travel affordable for everyone. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards 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 space X for a rocket launch.

- Problems you want to find answers
  - ➢ The task is to create a machine learning pipeline to predict if the first stage of SpaceX Falcon 9 will land successfully

Section 1

Methodology

Methodology

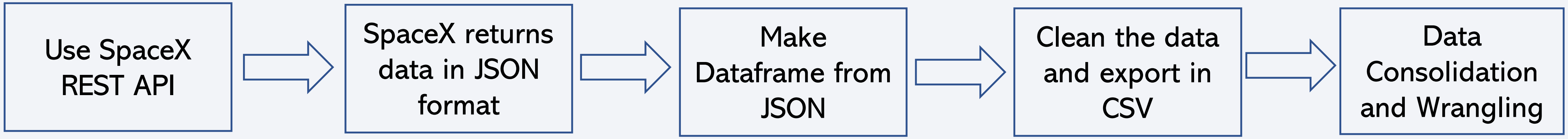
Executive Summary

- Data collection methodology:
  - ➢ SpaceX Rest API
  - ➢ Web Scrapping from Wikipedia
- Perform data wrangling
  - ➢ Dropping unnecessary columns and data cleaning of null values
  - ➢ One hot encoding for Classification Models
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - ➢ LR, KNN, SVM and DT models have been built for predictive analysis

Data Collection

- Data sets were collected using SpaceX REST API and web scrapping of Wikipedia.

➢ Using SpaceX REST API, launch data including launch site, payload delivered, launch specifications, landing specifications, landing outcome, etc information on Falcon 9 was gathered.



Use SpaceX REST API

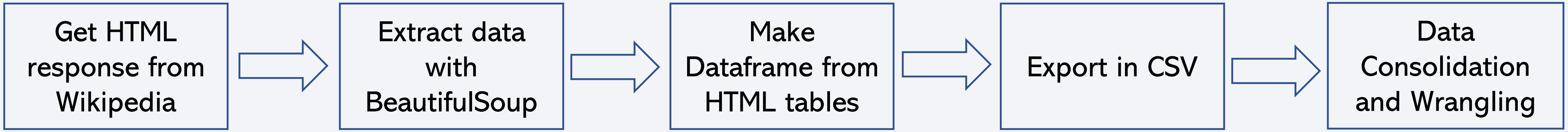
SpaceX returns data in JSON format

Make Dataframe from JSON

Clean the data and export in CSV

Data Consolidation and Wrangling

➢ Another popular data source for obtaining Falcon 9 launch data is web scrapping Wikipedia using BeautifulSoup.



Get HTML response from Wikipedia

Extract data with BeautifulSoup

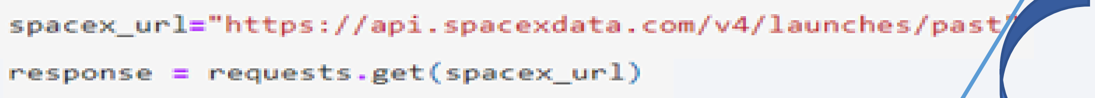
Make Dataframe from HTML tables

Export in CSV

Data Consolidation and Wrangling

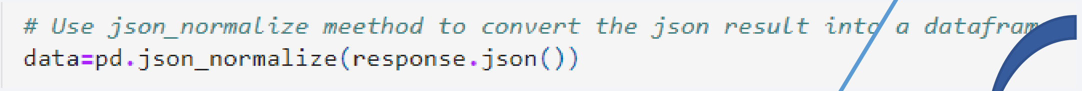
Data Collection – SpaceX API

1. Requesting and parsing the SpaceX launch data



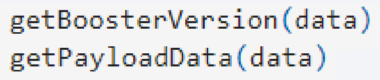
```
spacex_url="https://api.spacexdata.com/v4/launches/past"

response = requests.get(spacex_url)
```

2. Converting Json results into Dataframe

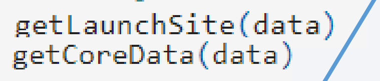


```
# Use json_normalize meethod to convert the json result into a datafram
data=pd.json_normalize(response.json())
```

3. Apply Custom functions to clean data

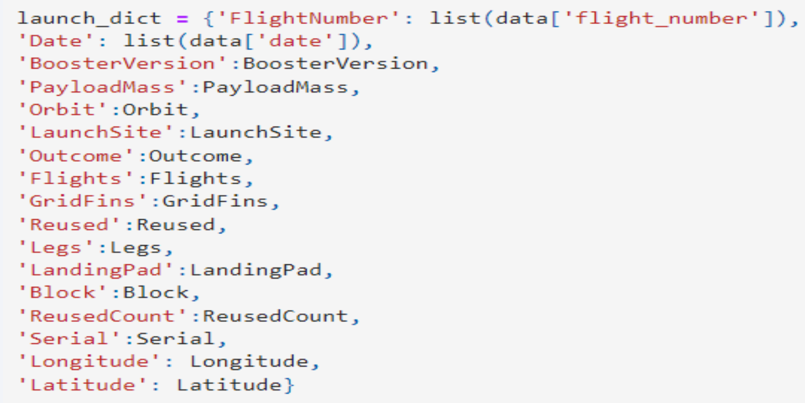


```
getBoosterVersion(data)
getPayloadData(data)
```



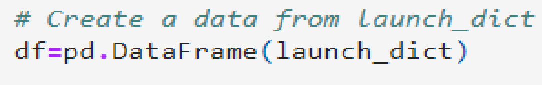
```
getLaunchSite(data)
getCoreData(data)
```

4. Create Dictionary with data



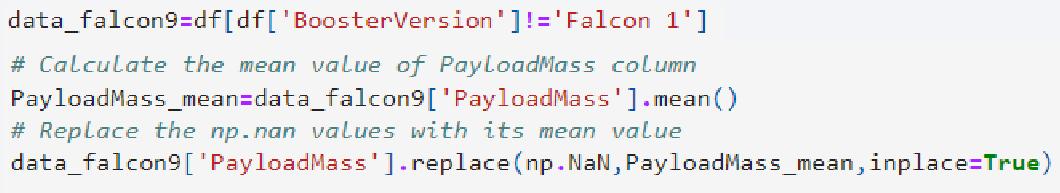
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

5. Create dataframe from dictionary



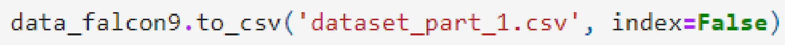
```
# Create a data from launch_dict
df=pd.DataFrame(launch_dict)
```

6. Filter and clean Dataframe



```
data_falcon9=df[df['BoosterVersion']!='Falcon 1']

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

7. Export to CSV file

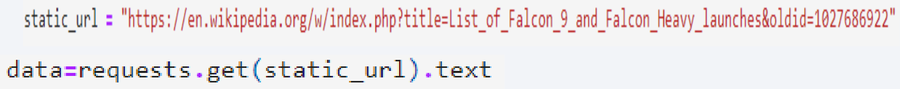


```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

GitHub URL

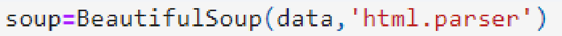
Data Collection - Scraping

1. Getting response from HTML



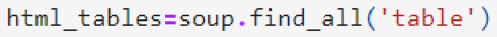
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

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

2. Creating BeautifulSoup Object



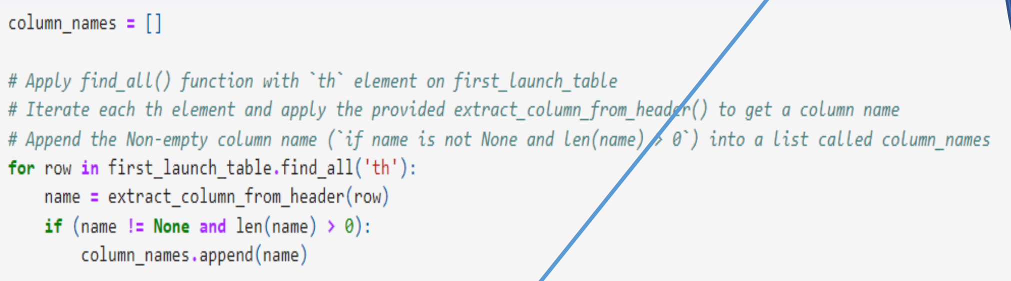
```
soup=BeautifulSoup(data,'html.parser')
```

3. Finding Tables



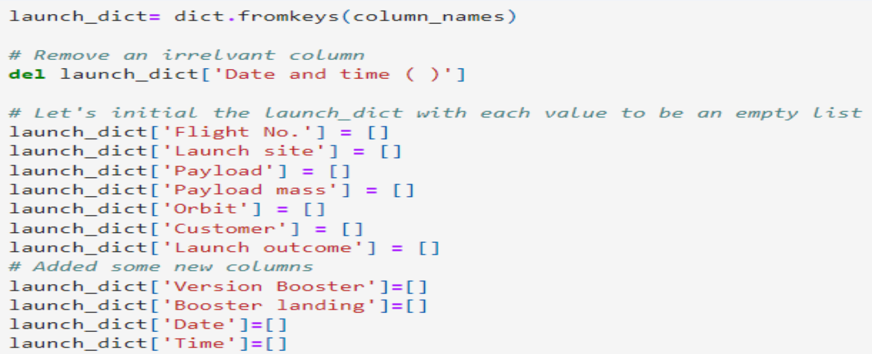
```
html_tables=soup.find_all('table')
```

4. Getting column names



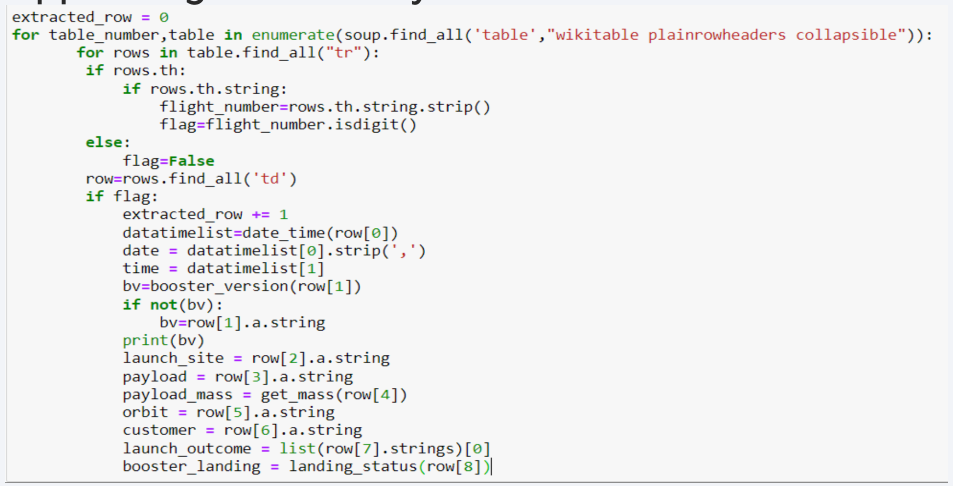
```
column_names = []

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

5. Creation of Dictionaries



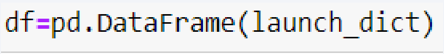
```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Appending Data to keys



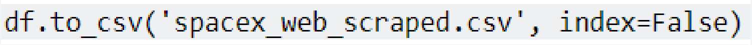
```
extracted_row = 0
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
       for rows in table.find_all("tr"):
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        row=rows.find_all('td')
        if flag:
            extracted_row += 1
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            time = datatimelist[1]
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            print(bv)
            launch_site = row[2].a.string
            payload = row[3].a.string
            payload_mass = get_mass(row[4])
            orbit = row[5].a.string
            customer = row[6].a.string
            launch_outcome = list(row[7].strings)[0]
            booster_landing = landing_status(row[8])
```

7. Converting Dictionary to Datafrmae



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

8. Dataframe to CSV file

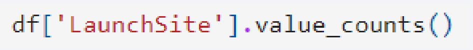


```
df.to_csv('spacex_web_scraped.csv', index=False)
```

GitHub URL

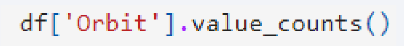
Data Wrangling

1. Calculate launch number for each site



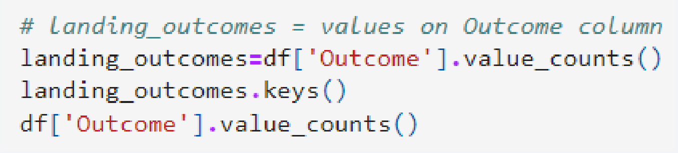
```
df['LaunchSite'].value_counts()
```

2. Calculate the number and occurrence of each orbit



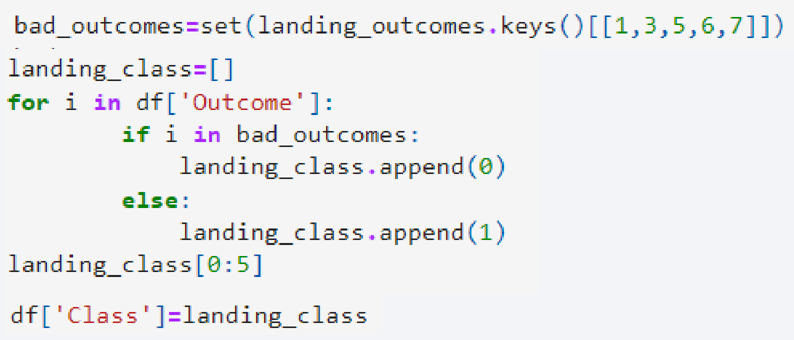
```
df['Orbit'].value_counts()
```

3. Calculate number and occurrence of mission outcome per orbit type



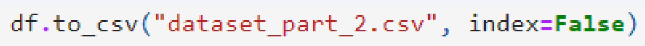
```
# landing_outcomes = values on Outcome column
landing_outcomes=df['Outcome'].value_counts()
landing_outcomes.keys()
df['Outcome'].value_counts()
```

4. Create landing outcome label from Outcome Column



```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
landing_class=[]
for i in df['Outcome']:
        if i in bad_outcomes:
            landing_class.append(0)
        else:
            landing_class.append(1)
landing_class[0:5]
df['Class']=landing_class
```

5. Export to CSV file



```
df.to_csv("dataset_part_2.csv", index=False)
```

GitHub URL

EDA with Data Visualization

- Charts Plotted:
  - ❑ Scatter Graphs:
    - ➢ Flight Number vs Payload Mass
    - ➢ Flight Number vs Launch Site
    - ➢ Payload Mass vs Launch Site
    - ➢ Orbit type vs Flight Number
    - ➢ Payload Mass vs Orbit type
  - ❑ Bar Graph of Success rate vs Orbit type
  - ❑ Line Graph of Success rate vs year

- Charts were plotted to:
  - ❑ Find correlation between variables
  - ❑ Show the relation between numeric and categoric variables
  - ❑ Show data variables and their trend

GitHub URL

EDA with SQL

- SQL Queries performed include:
  - ➢ Displaying the names of the unique launch sites in the space mission
  - ➢ Displaying 5 records where launch sites begin with the string 'CCA'
  - ➢ Displaying the total payload mass carried by boosters launched by NASA (CRS)
  - ➢ Displaying average payload mass carried by booster version F9 v1.1
  - ➢ Listing the date when the first successful landing outcome in the ground pad was achieved
  - ➢ Listing the names of the boosters which have success in drone ships and have payload mass greater than 4000 but less than 6000
  - ➢ Listing the total number of successful and failed mission outcomes
  - ➢ Listing the names of the booster versions which have carried the maximum payload mass. Use a subquery
  - ➢ Listing the records which will display the month names, failure landing outcomes in drone ship, booster versions, and launch site for the months in the year 2015
  - ➢ Rank the count of successful landing outcomes between the dates 2010-06-04 and 2017-03-20 in descending order.

GitHub URL

Build an Interactive Map with Folium

- Markers, Circles, Lines, and marker clusters were used with Folium Maps
  - ➢ Markers indicate points like Launch Sites
  - ➢ Circles indicate highlighted areas around specific coordinates like NASA Johnson Space Center
  - ➢ Marker Clusters indicate groups of events in each coordinate, like launches in a launch site and successful and unsuccessful landings. Green for Successful landing and red for unsuccessful landing.
  - ➢ Lines are used to indicate distances between the launch site to key locations like railway, highway, city, and coast way.

Github URL

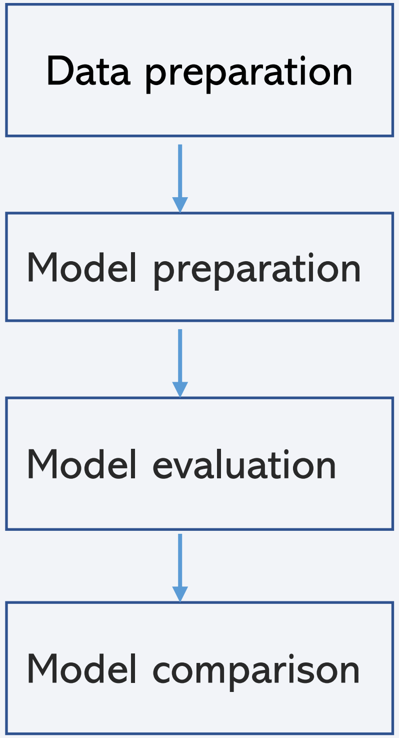
Build a Dashboard with Plotly Dash

- Dashboard has a dropdown, pie chart, range slider and scatter plot components
  - ➢ Dropdown allows a user to choose a launch site or all the launch sites.
  - ➢ Pie chart shows the total success and total failure for the launch site chosen with the dropdown component
  - ➢ Rangeslider allows a user to select a payload mass in a fixed range
  - ➢ Scatter chart shows the relationship between two variables, in particular, Success vs Payload Mass

GitHub URL

Predictive Analysis (Classification)

- Data preparation
  - ➢ Load dataset
  - ➢ Normalize data
  - ➢ Split data into training and test sets.
- Model preparation
  - ➢ Selection of machine learning algorithms
  - ➢ Set parameters for each algorithm to GridSearchCV
  - ➢ Training GridSearchModel models with the training dataset
- Model evaluation
  - ➢ Get the best hyperparameters for each type of model
  - ➢ Compute accuracy for each model with the test dataset
  - ➢ Plot Confusion Matrix
- Model comparison
  - ➢ Comparison of models according to their accuracy
  - ➢ The model with the best accuracy will be chosen


Data preparation

Model preparation

Model evaluation

Model comparison

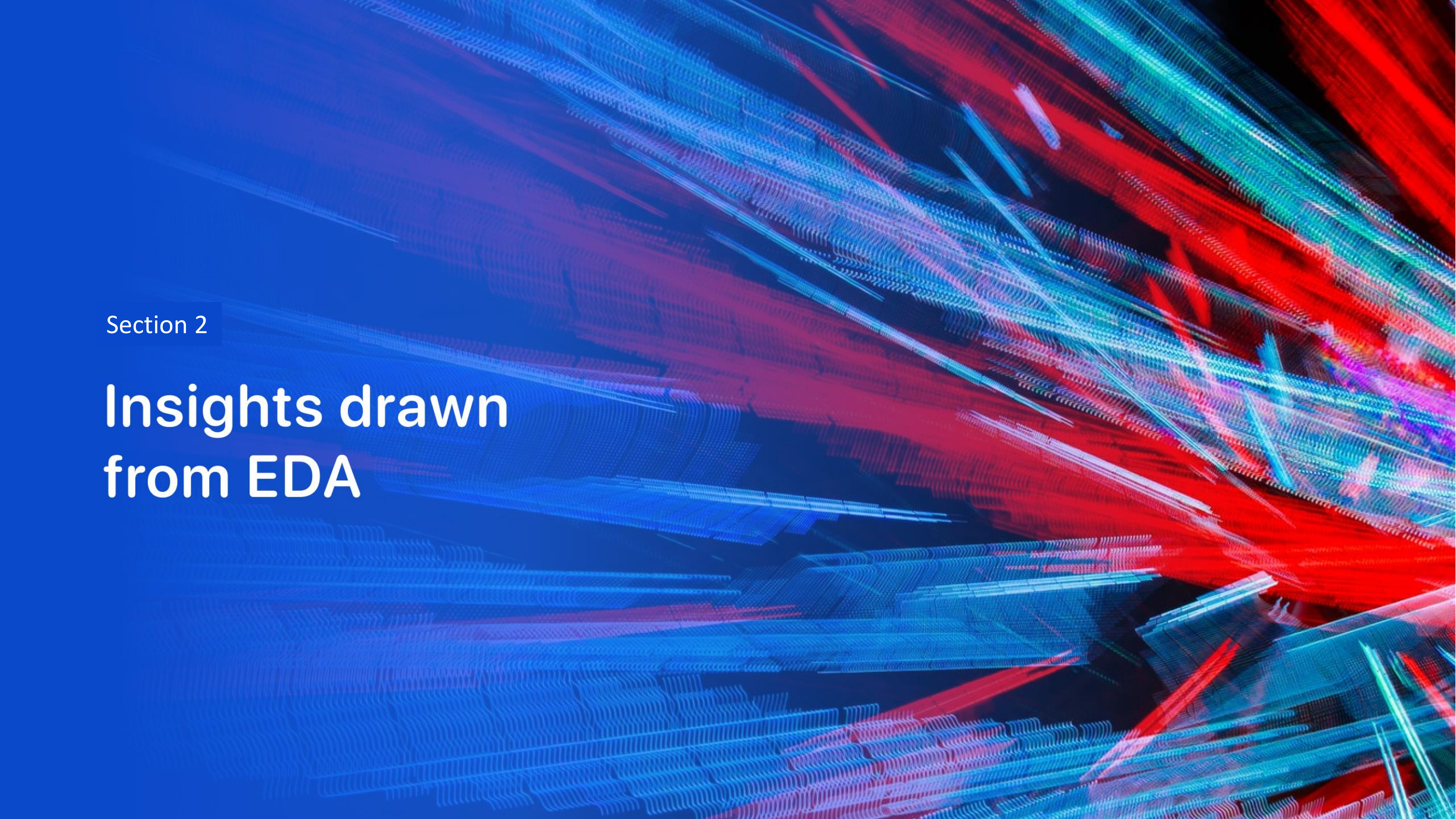
[GitHub URL]

Results

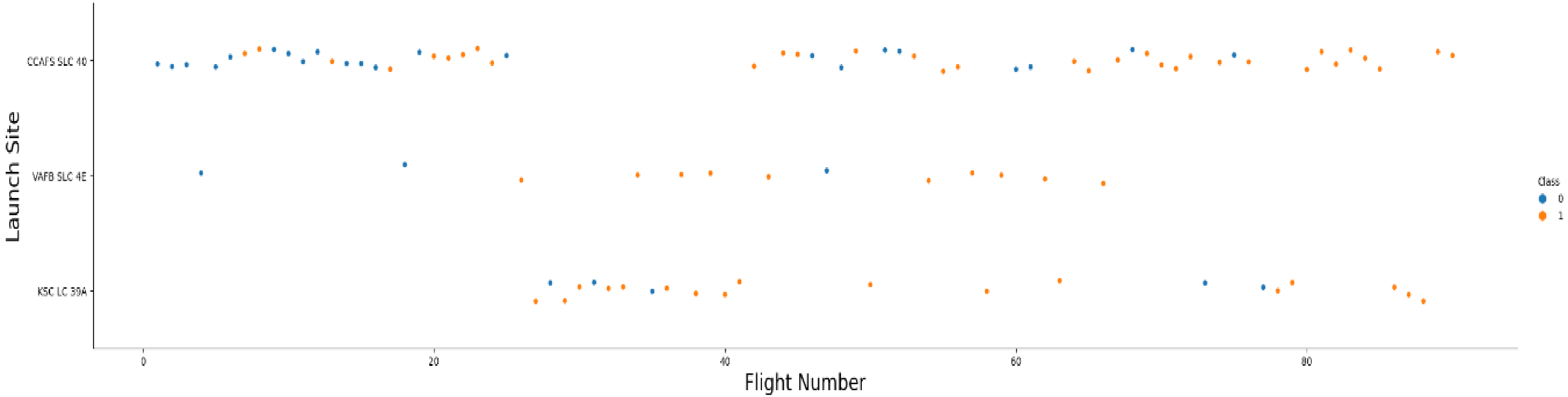
- Exploratory data analysis results:
  - ➢ Space X uses 4 different launch sites;
  - ➢ The first launches were done to Space X itself and NASA;
  - ➢ The average payload of the F9 v1.1 booster is 2,928 kg;
  - ➢ The first successful landing outcome happened in 2015 five years after the first launch;
  - ➢ Many Falcon 9 booster versions were successful at landing in drone ships having payloads above the average;
  - ➢ Almost 100% of mission outcomes were successful;
  - ➢ Two booster versions failed at landing in drone ships in 2015: F9 v1.1 B1012 and F9 v1.1 B1015;
  - ➢ The number of landing outcomes became as better as the years passed.
- Using interactive analytics was possible to identify that launch sites use to be in safety places, near sea, for example and have a good logistic infrastructure around. Most launches happens at east cost launch sites.
- Predictive Analysis showed that the Decision Tree Classifier is the best model to predict successful landings, having accuracy over 87% and accuracy for test data over 94%.

Section 2

Insights drawn from EDA

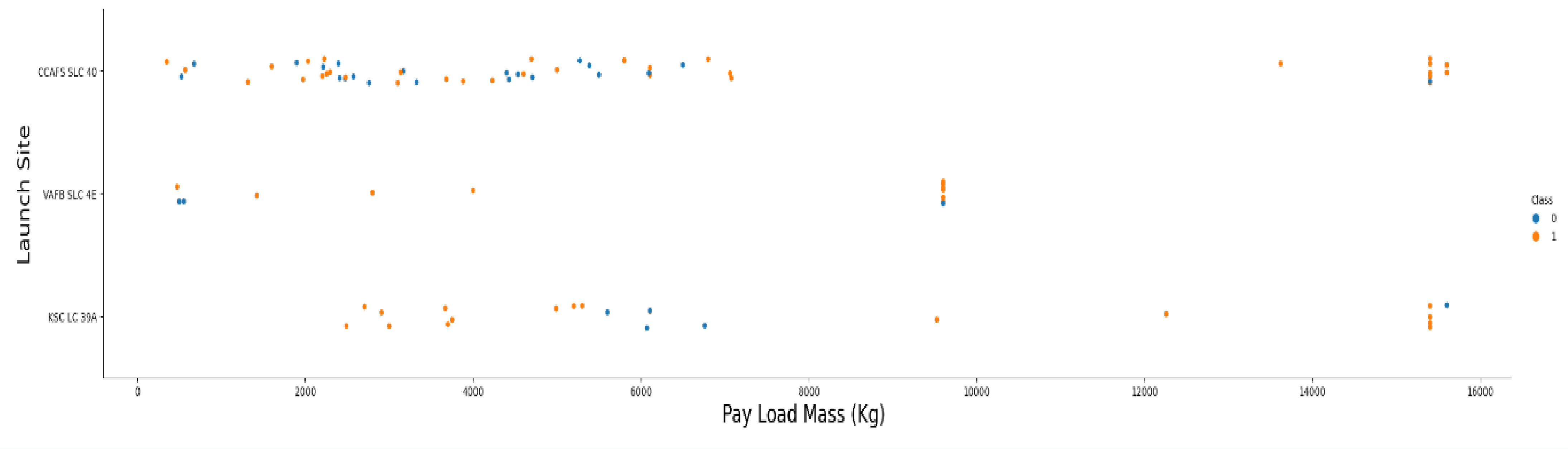


Flight Number vs. Launch Site



- According to the plot above, it’s possible to verify that the best launch site nowadays is CCAF5 SLC 40, where most of recent launches were successful;
- In second place VAFB SLC 4E and third place KSC LC 39A;
- It’s also possible to see that the general success rate improved over time.

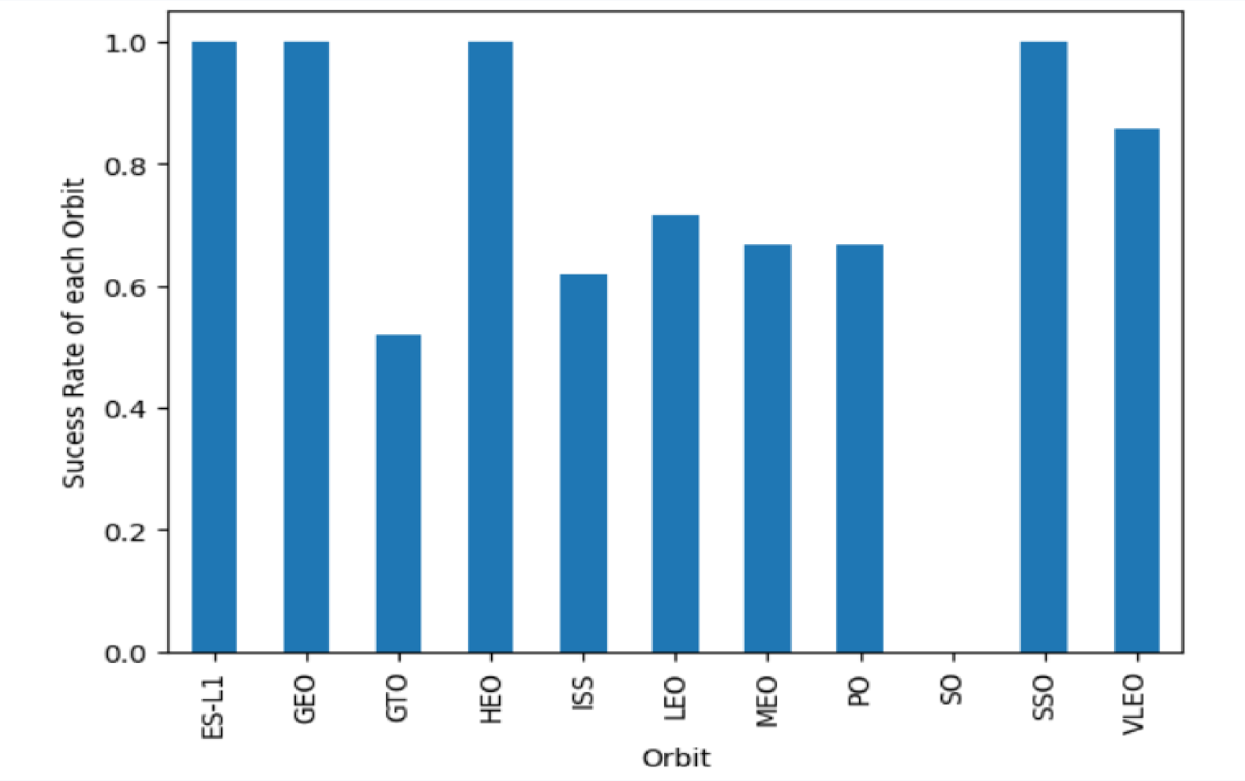
Payload vs. Launch Site



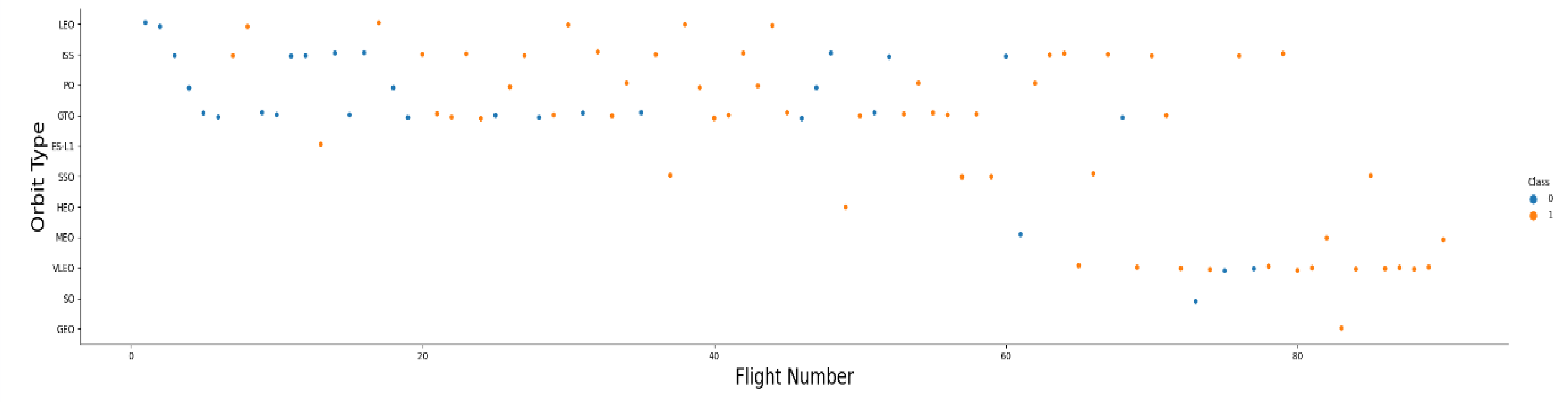
- ➢ Payloads over 9,000kg (about the weight of a school bus) have excellent success rate;
- ➢ Payloads over 12,000kg seems to be possible only on CCAFS SLC 40 and KSC LC 39A launch sites.

Success Rate vs. Orbit Type

- The biggest success rates happen to orbits:
  - ➢ ES-L1;
  - ➢ GEO;
  - ➢ HEO; and
  - ➢ SSO.
  - ➢ VLEO (above 80%); and
  - ➢ LFO (above 70%).

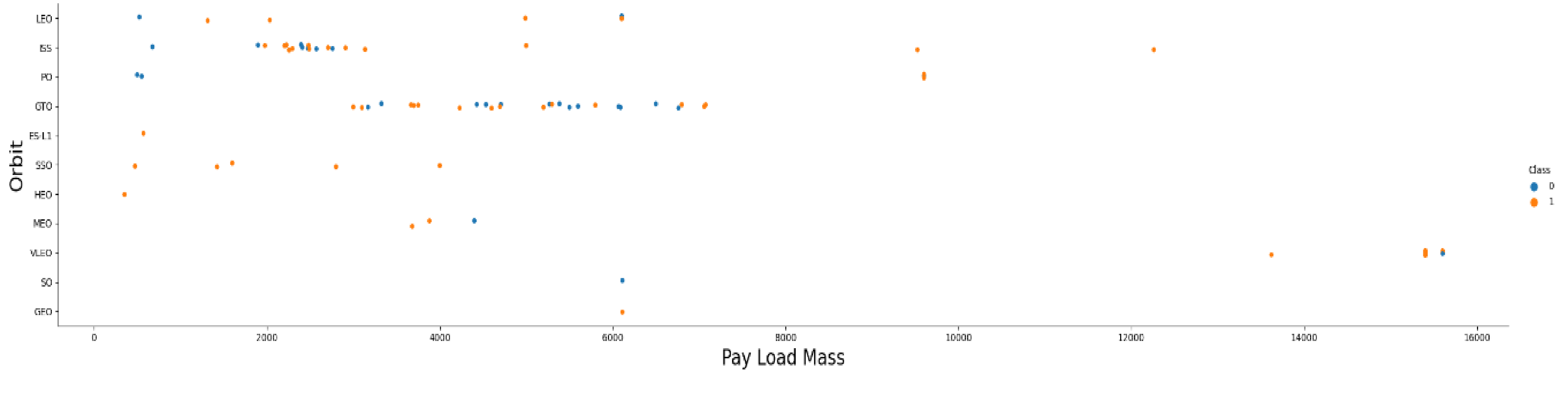


Flight Number vs. Orbit Type



- ➢ Apparently, success rate improved over time to all orbits;
- ➢ VLEO orbit seems a new business opportunity, due to recent increase of its frequency.

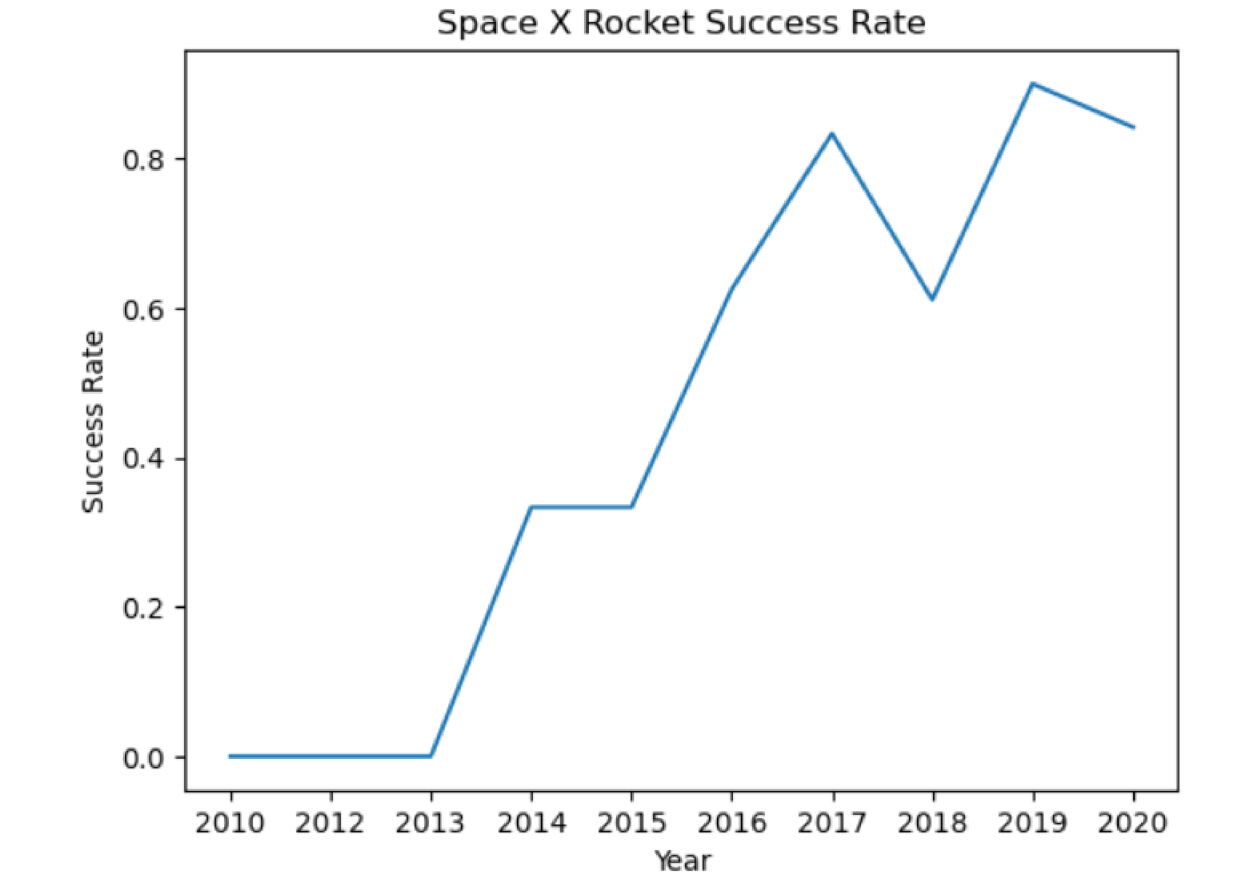
Payload vs. Orbit Type



- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

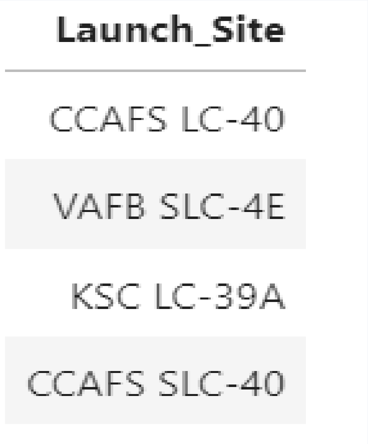
Launch Success Yearly Trend

- Success rate started increasing in 2013 and kept until 2020;



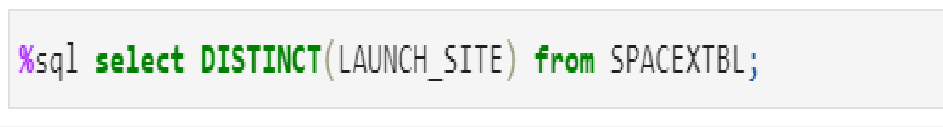
All Launch Site Names

- According to data, there are four launch sites:



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

- They are obtained by selecting unique occurrences of “LAUNCH_SITE” values from the dataset.

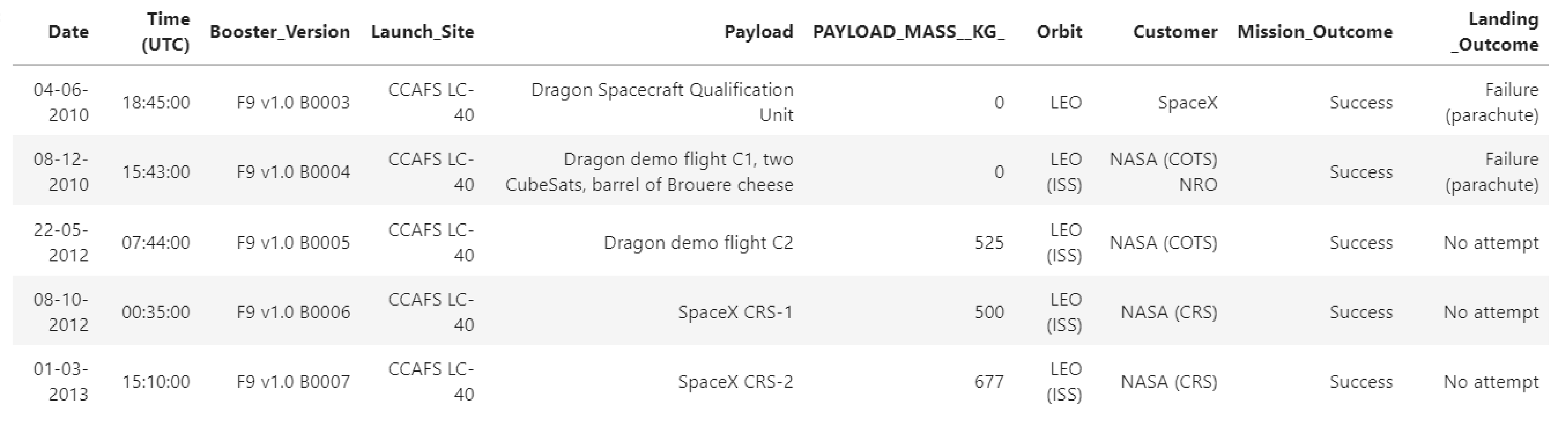


```
%sql select DISTINCT(LAUNCH_SITE) from SPACEXTBL;
```

- The use of DISTINCT in the query allows to remove duplicate LAUNCH_SITE.

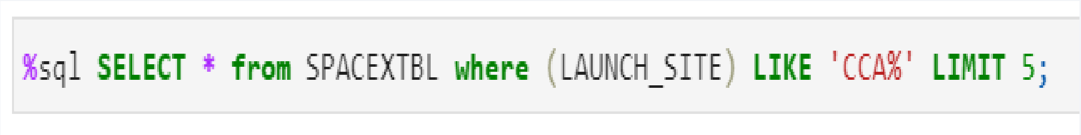
Launch Site Names Begin with 'CCA'

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



Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

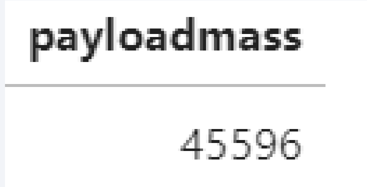
- The WHERE clause followed by LIKE clause filters launch sitesthat contain the substring CCA. LIMIT 5 shows 5 records from filtering.



```
%sql SELECT * from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

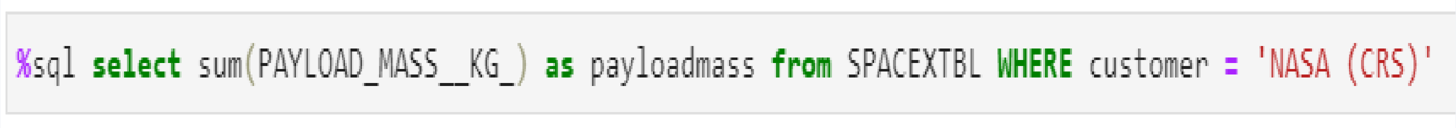
Total Payload Mass

- Total payload mass (Kg) carried by boosters from NASA:



payloadmass
45596

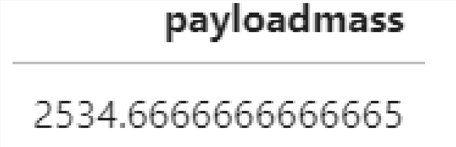
- Total payload calculated above, by summing all payloads whose codes contain ‘CRS’, which corresponds to NASA.



```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL WHERE customer = 'NASA (CRS)'
```

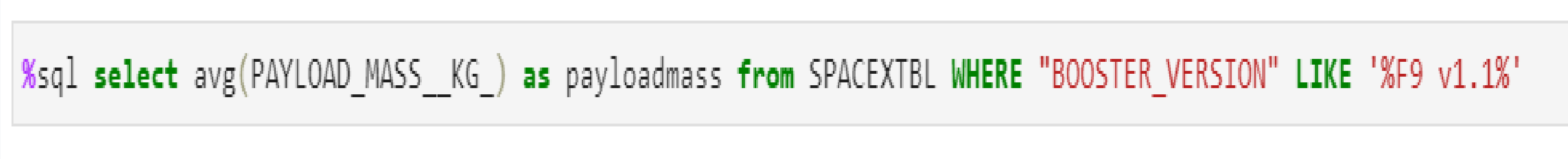
Average Payload Mass by F9 v1.1

- Average payload mass (Kg) carried by booster version F9 v1.1:



payloadmass
2534.6666666666665

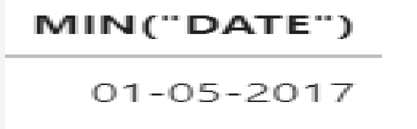
- Filtering data by the booster version above and calculating the average payload mass we obtained the value of 2534.67 Kg.



```
%sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL WHERE "BOOSTER_VERSION" LIKE '%F9 v1.1%'
```

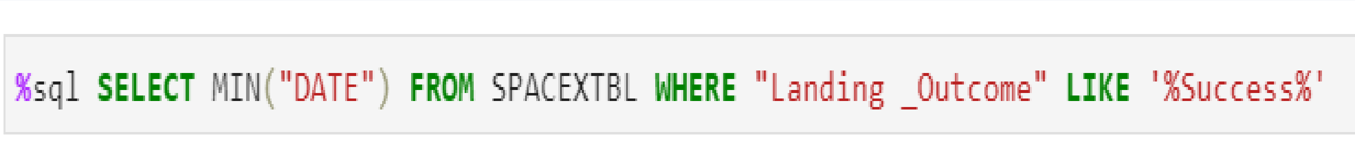
First Successful Ground Landing Date

- First successful landing outcome on ground pad:



MIN("DATE")
01-05-2017

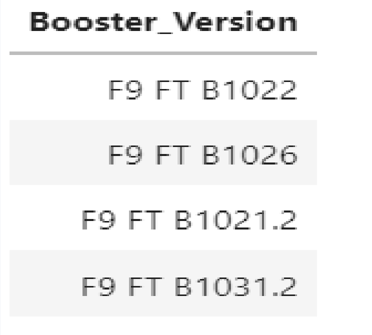
- With this query, we select the oldest successful landing. The WHERE clause filters dataset in order to keep only records where landing was successful. With the MIN function, we select the record with the oldest date.



```
%sql SELECT MIN("DATE") FROM SPACEXTBL WHERE "Landing _Outcome" LIKE '%Success%'
```

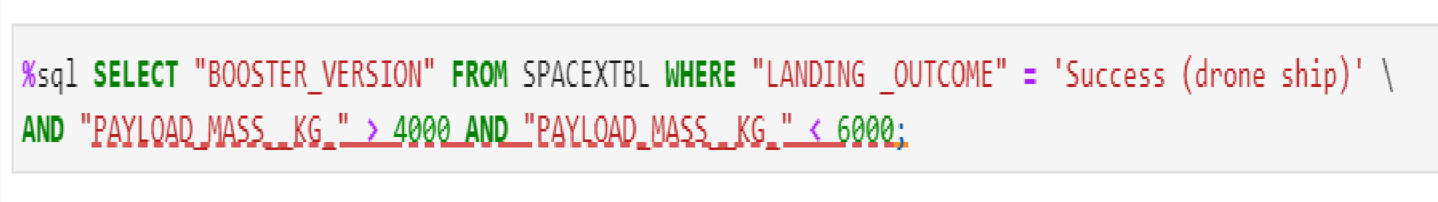
Successful Drone Ship Landing with Payload between 4000 and 6000

- Boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000



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

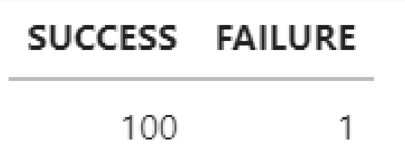
- The WHERE and AND clauses filter the dataset.



```
%sql SELECT "BOOSTER_VERSION" FROM SPACEXTBL WHERE "LANDING _OUTCOME" = 'Success (drone ship)' \
AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000;
```

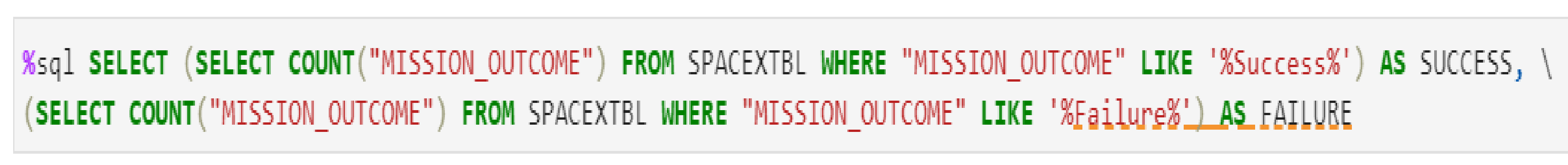
Total Number of Successful and Failure Mission Outcomes

- Number of successful and failure mission outcomes:



SUCCESS	FAILURE
100	1

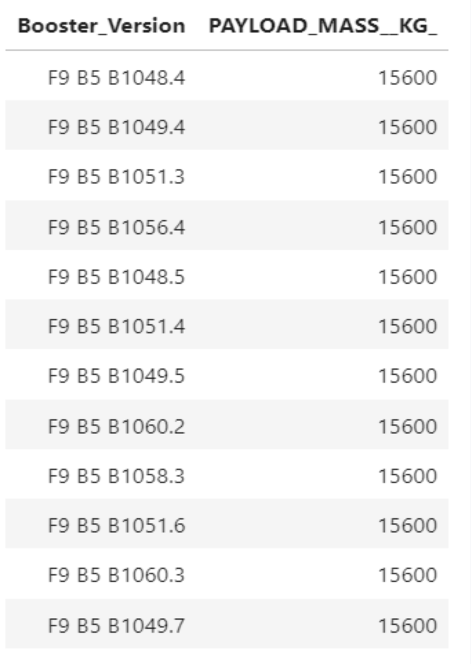
- With the first SELECT, we show the subqueries that return results. The first subquery counts the successful mission. The second subquery counts the unsuccessful mission. The WHERE clause followed by LIKE clause filters mission outcome. The COUNT function counts records filtered.



```
%sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Success%') AS SUCCESS, \
(SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS FAILURE
```

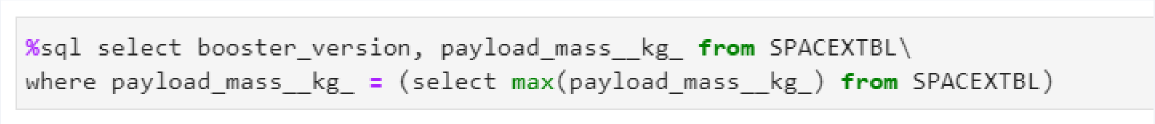
Boosters Carried Maximum Payload

- Boosters which have carried the maximum payload mass



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

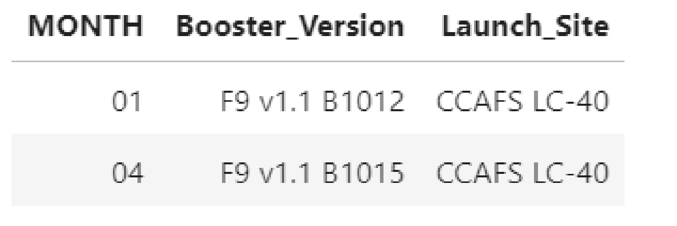
- We used a subquery to filter data by returning only the heaviest payload mass with MAX function. The main query uses subquery results and returns unique booster version (SELECT DISTINCT) with the heaviest payload mass.



```
%sql select booster_version, payload_mass__kg_ from SPACEXTBL\
where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTBL)
```

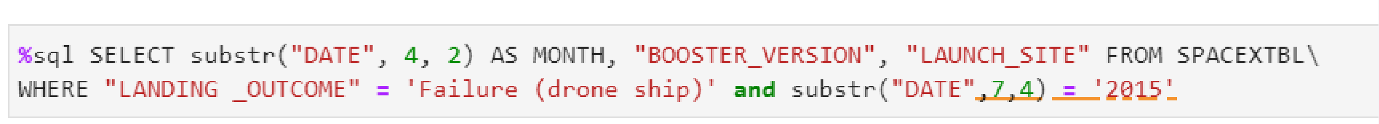
2015 Launch Records

- Failed landing outcomes in drone ship, their booster versions, andlaunch site names for in year 2015



MONTH	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

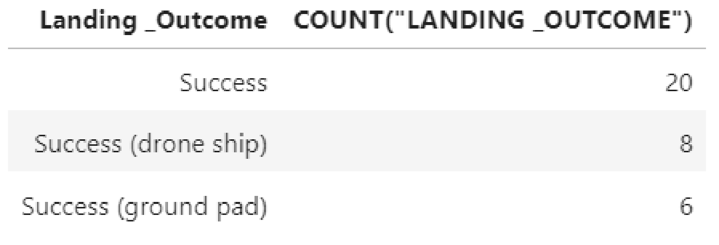
- This query returns month, booster version, launch site where landing was unsuccessful and landing date took place in2015. Substr function process date in order to take month or year. Substr(DATE, 4, 2) shows month. Substr(DATE,7, 4) shows year.



```
%sql SELECT substr("DATE", 4, 2) AS MONTH, "BOOSTER_VERSION", "LAUNCH_SITE" FROM SPACEXTBL\
WHERE "LANDING _OUTCOME" = 'Failure (drone ship)' and substr("DATE",7,4) = '2015'
```

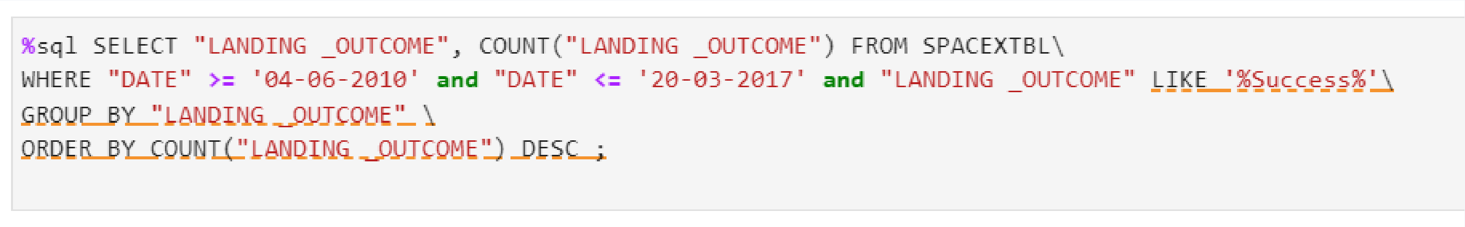
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- Ranking of all landing outcomes between the date 2010-06-04 and 2017-03-20:



Landing _Outcome	COUNT("LANDING _OUTCOME")
Success	20
Success (drone ship)	8
Success (ground pad)	6

- This query returns landing outcomes and their countwhere mission was successful and date is between 04/06/2010 and 20/03/2017. The GROUP BY clause groups results by landing outcome and ORDER BY COUNTDESC shows results in decreasing order.

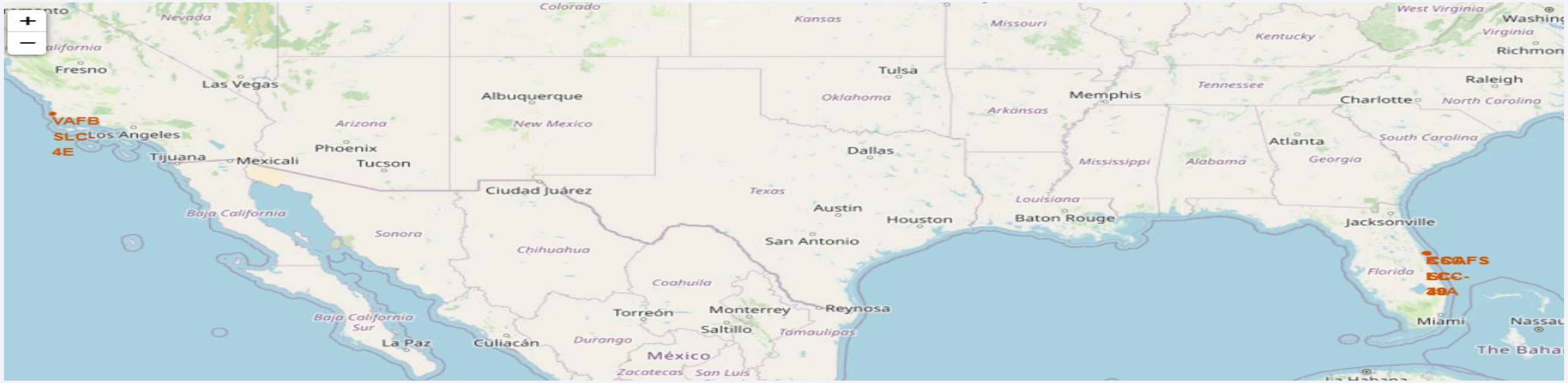


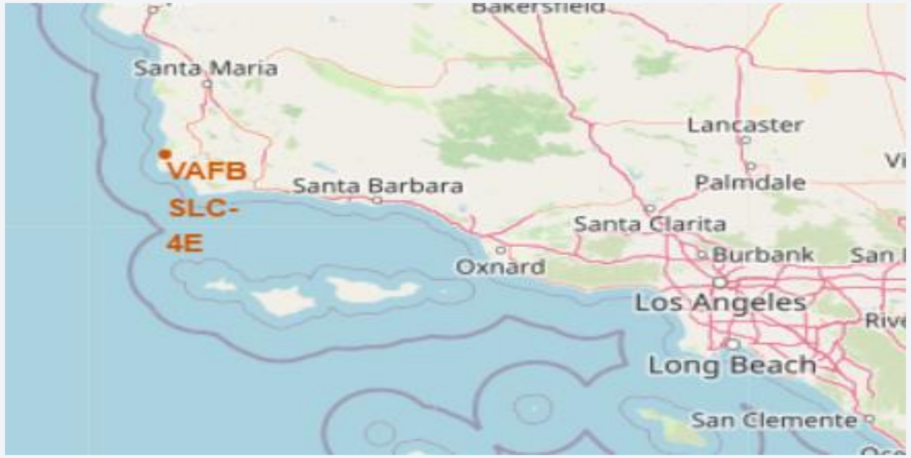
```
%sql SELECT "LANDING _OUTCOME", COUNT("LANDING _OUTCOME") FROM SPACEXTBL\
WHERE "DATE" >= '04-06-2010' and "DATE" <= '20-03-2017' and "LANDING _OUTCOME" LIKE '%Success%'\
GROUP BY "LANDING _OUTCOME" \
ORDER BY COUNT("LANDING _OUTCOME") DESC ;
```

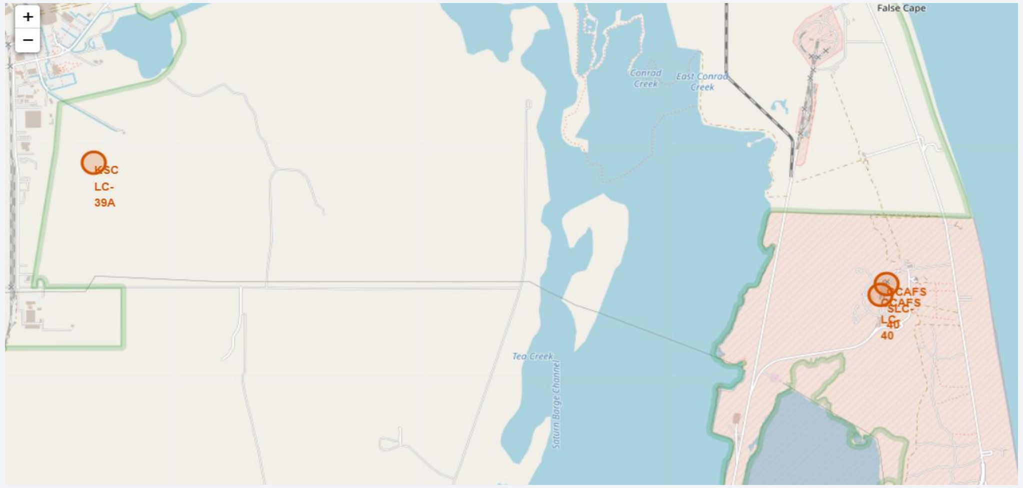
Section 3

Launch Sites Proximities Analysis

Location of All launch sites

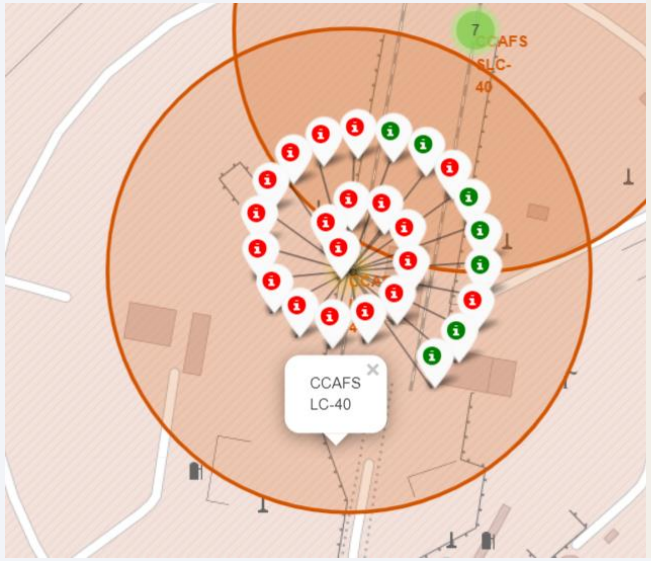


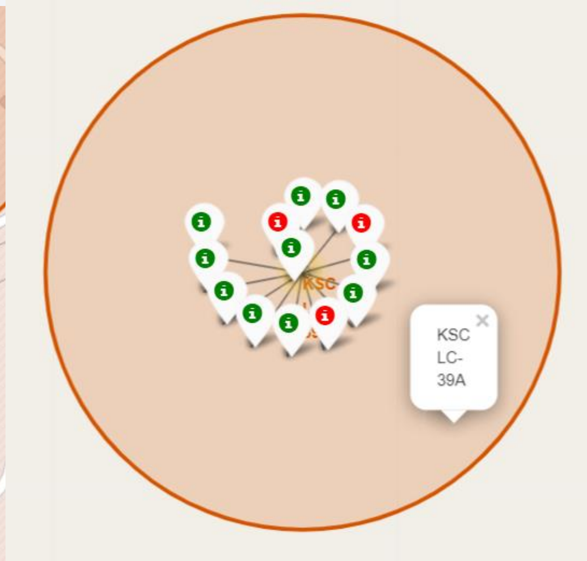


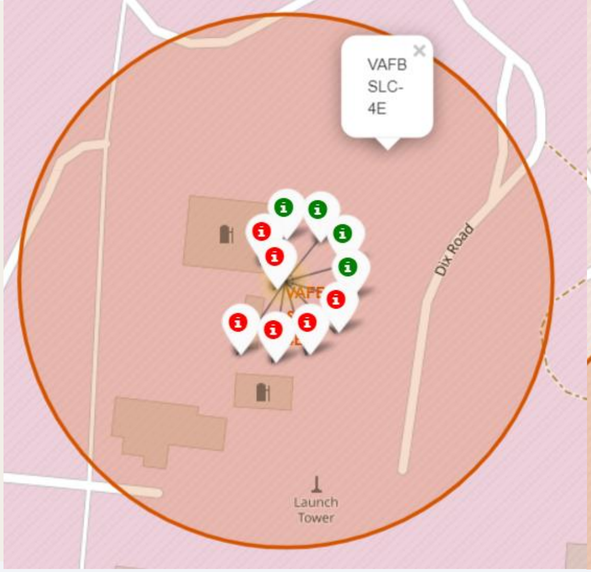


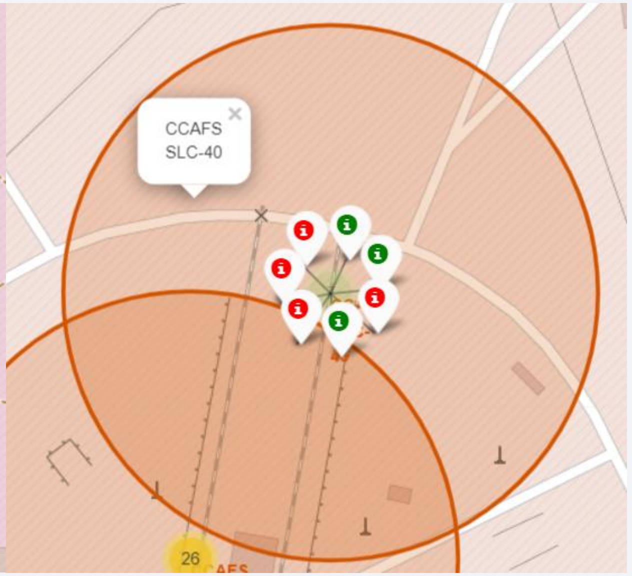
- Launch sites are near sea, probably by safety, but not too far from roads and railroads.

Launch Outcomes by All Launch Site



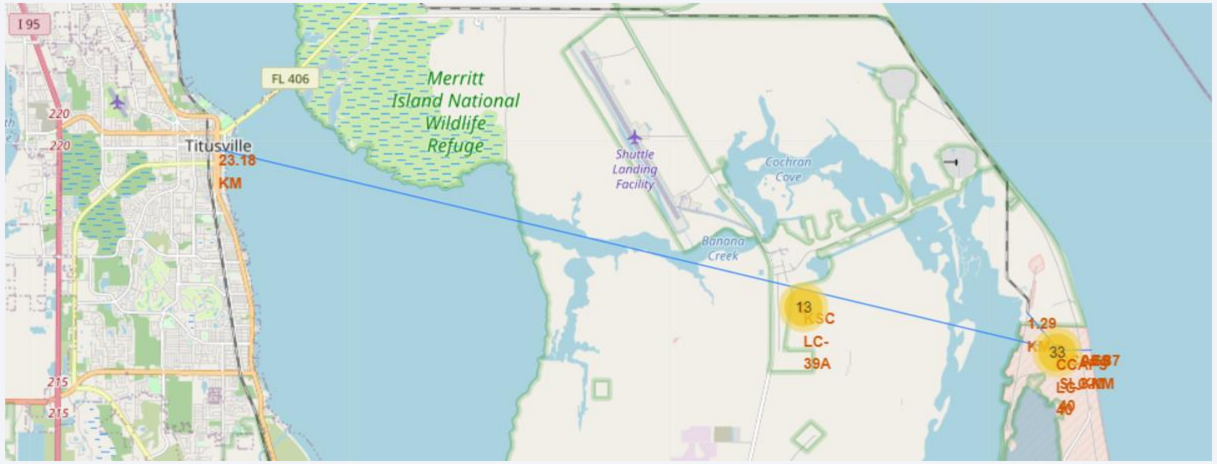


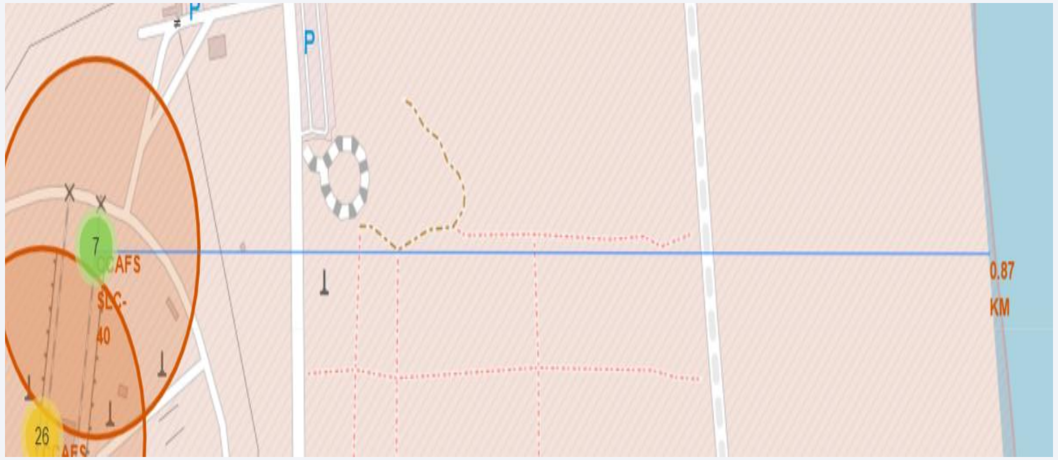


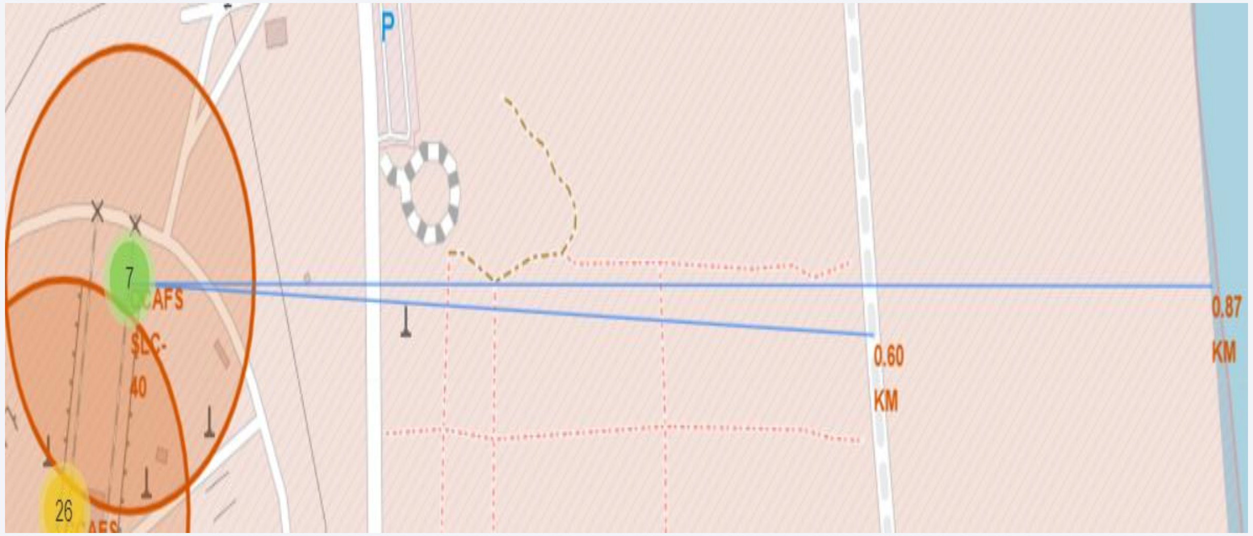


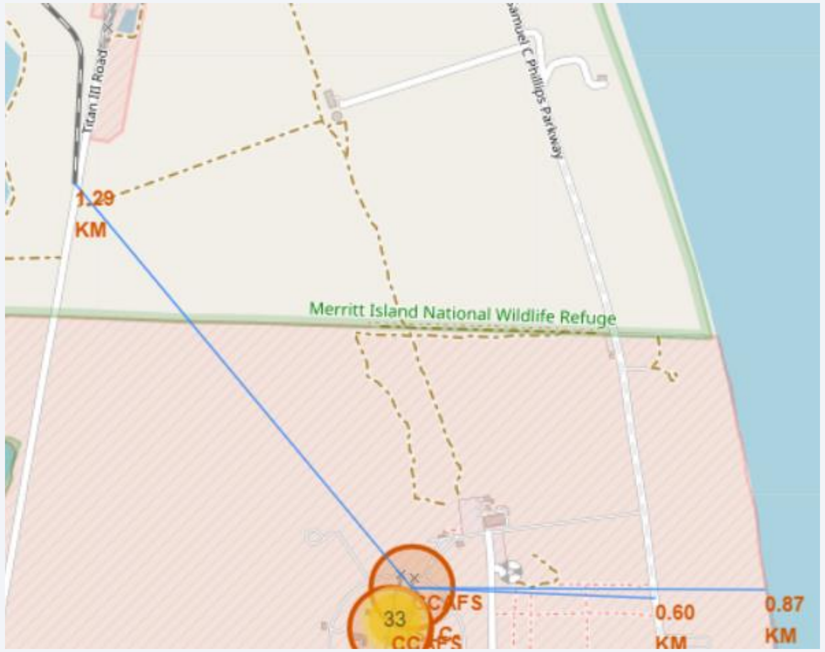
- Greenmarker represents successful launches. Redmarker represents unsuccessful launches. We note that KSC LC-39A has a higher launch success rate.

Distances between CCAFS SLC-40 and its proximities



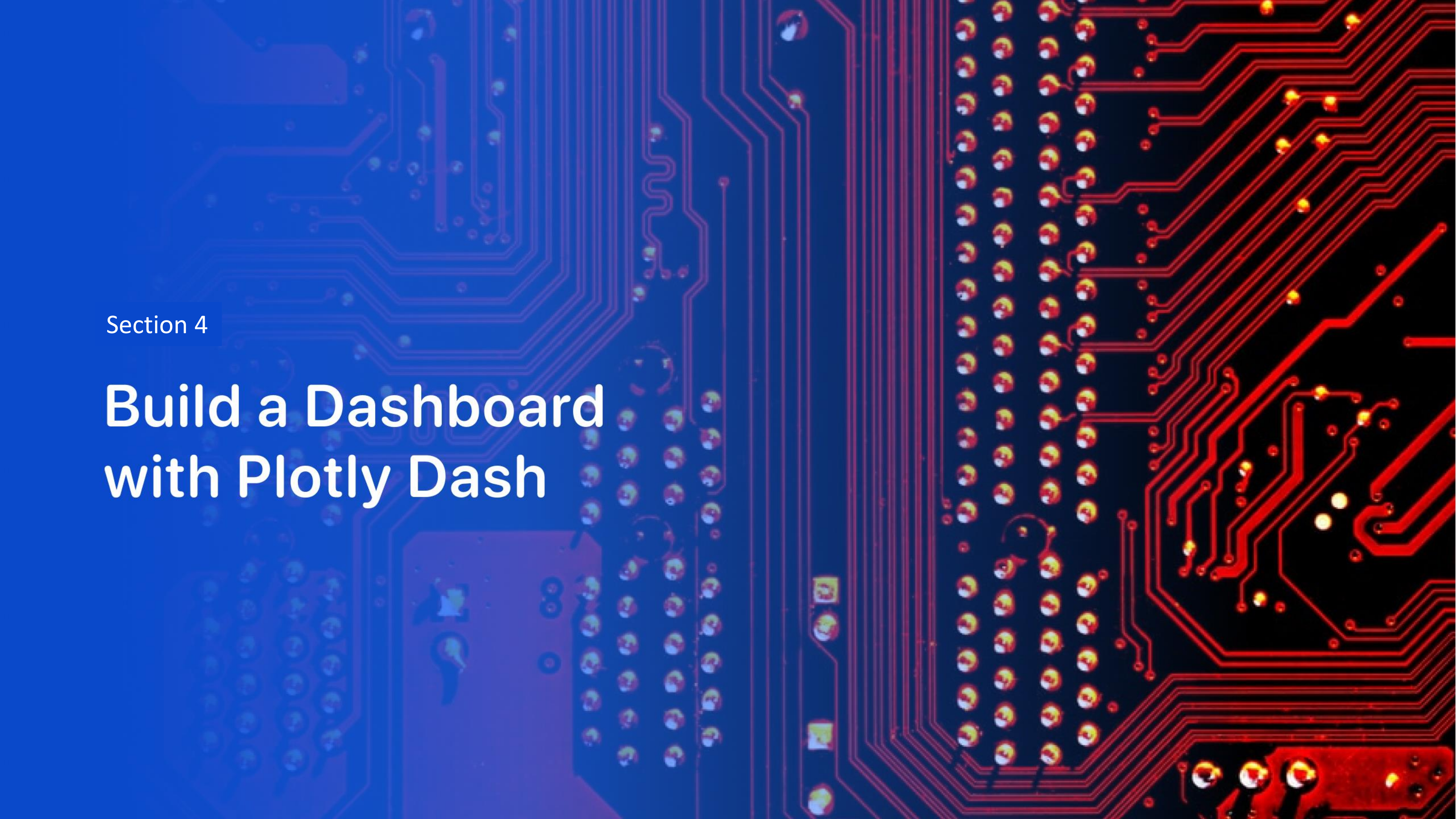




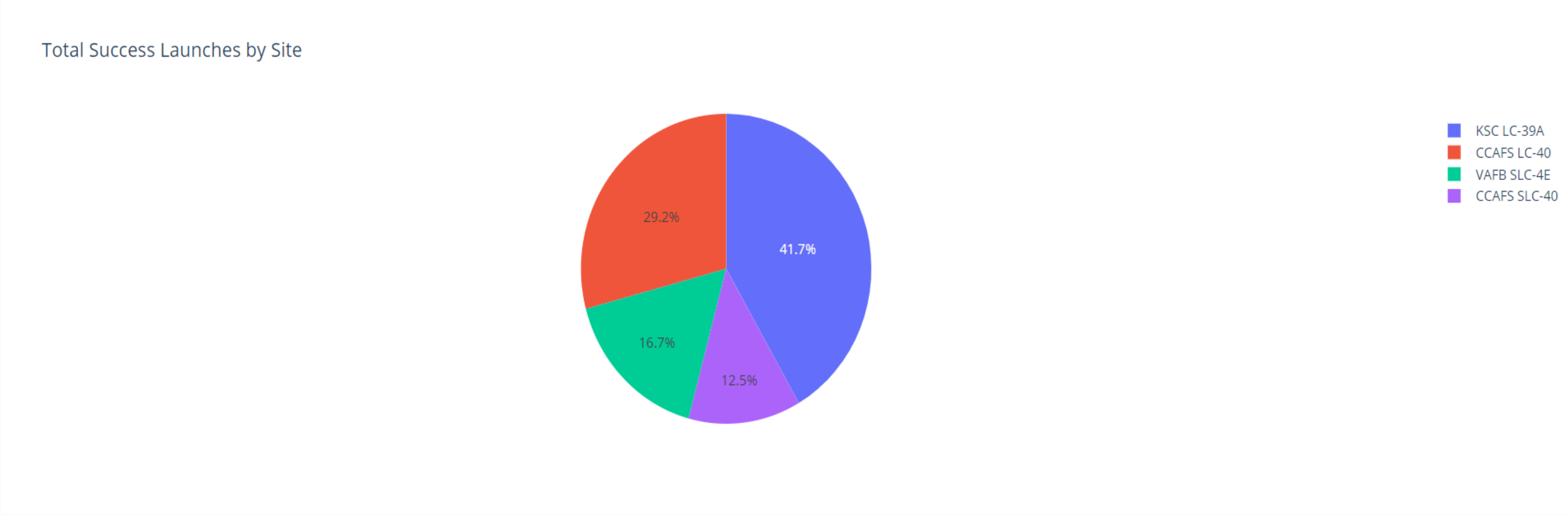

Titan III Road
Samuel C Phillips Parkway
1.29 KM
Merritt Island National Wildlife Refuge
CCAFS
33
0.60 KM
0.87 KM

Section 4

Build a Dashboard with Plotly Dash

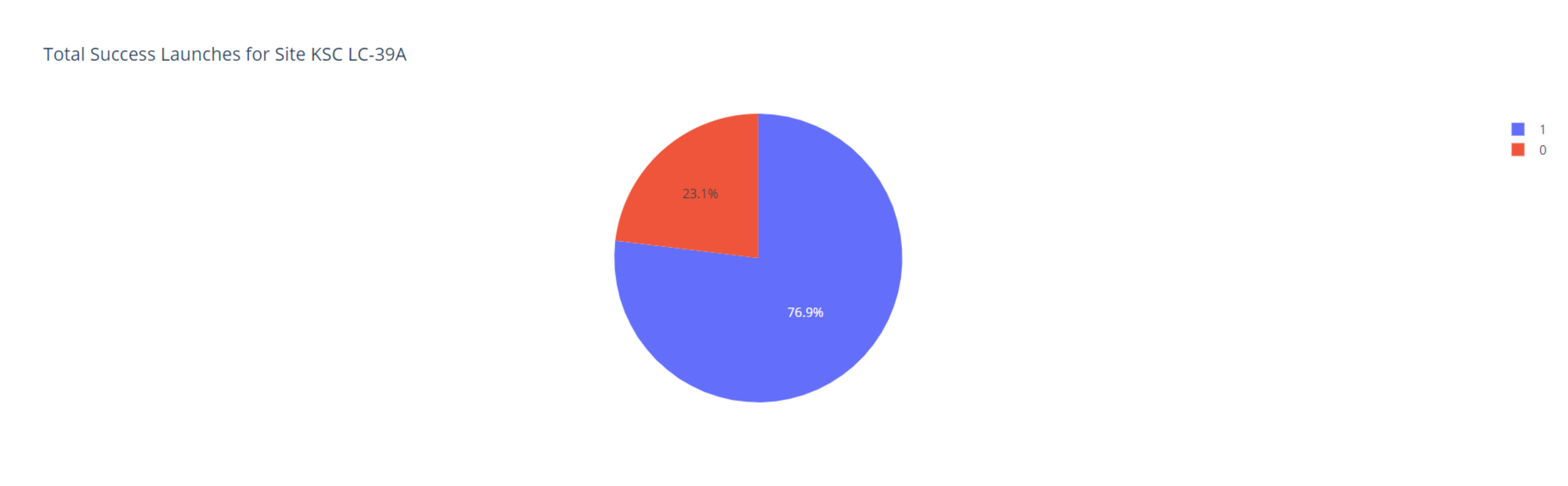


Successful Launches by Site



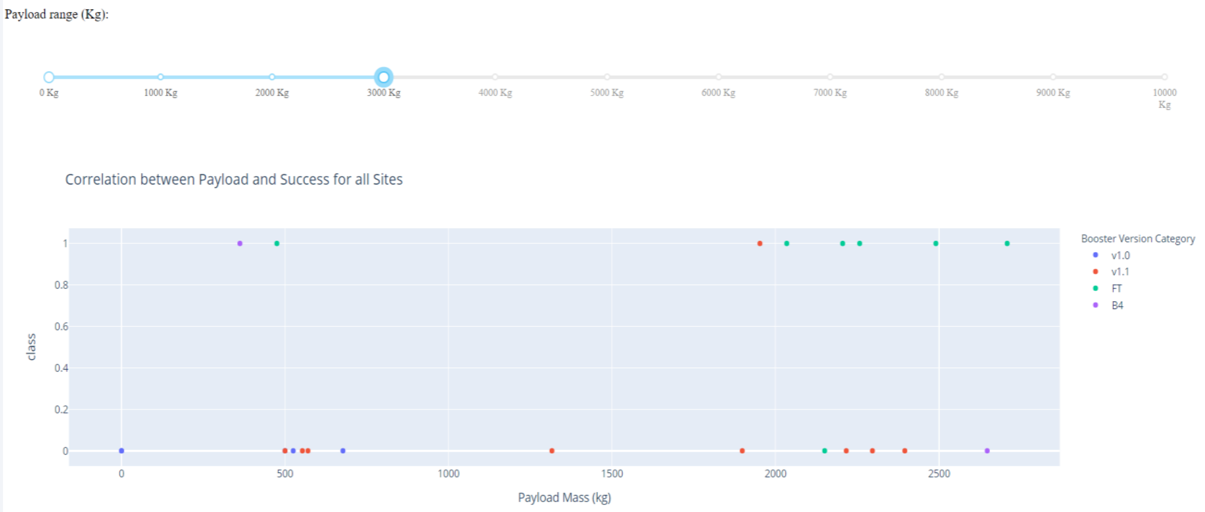
- The place from where launches are done seems to be a very important factor of the success of missions.

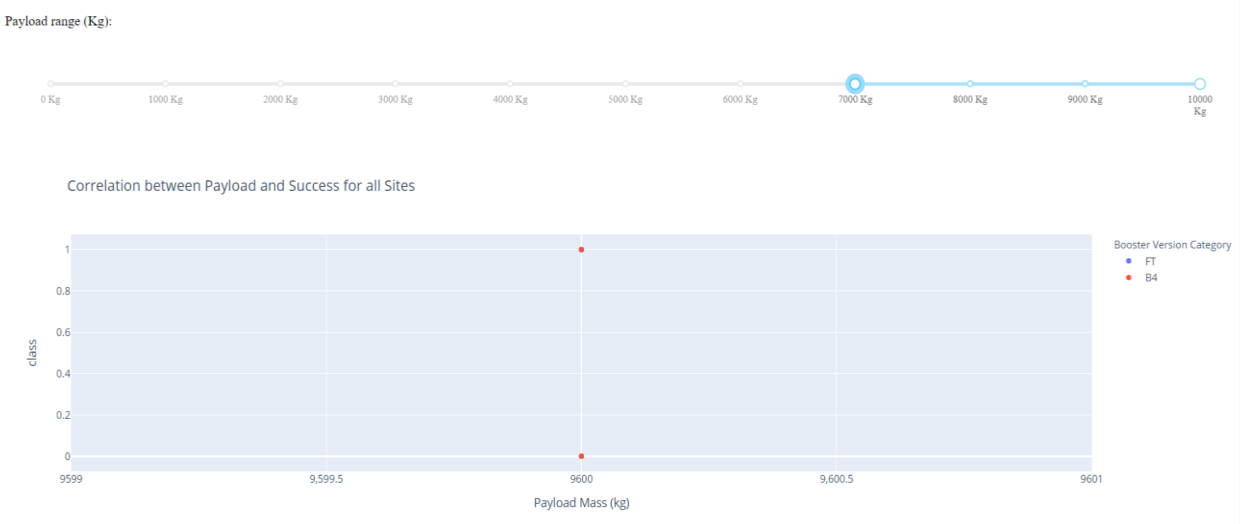
Launch Success Ratio for KSC LC-39A

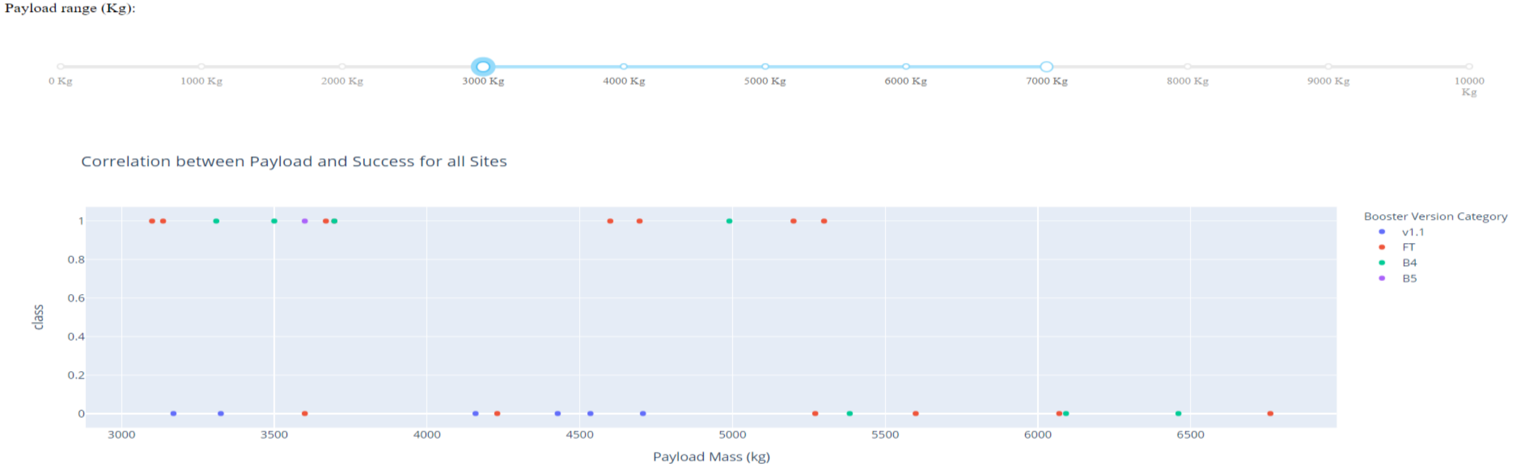


- 76.9% of launches are successful in this site.

Payload vs. Launch Outcome







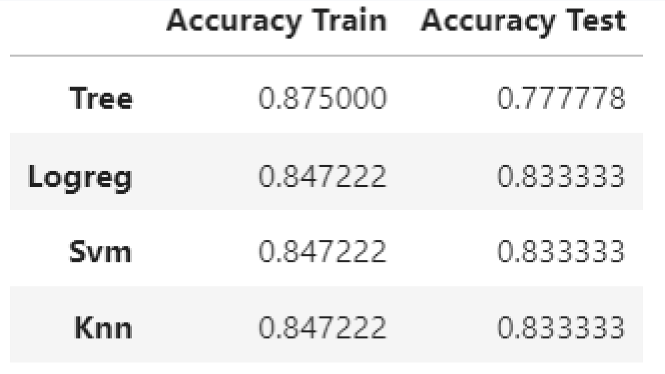
- Low weighted payloads have a better success rate than the heavy weighted payloads.

Section 5

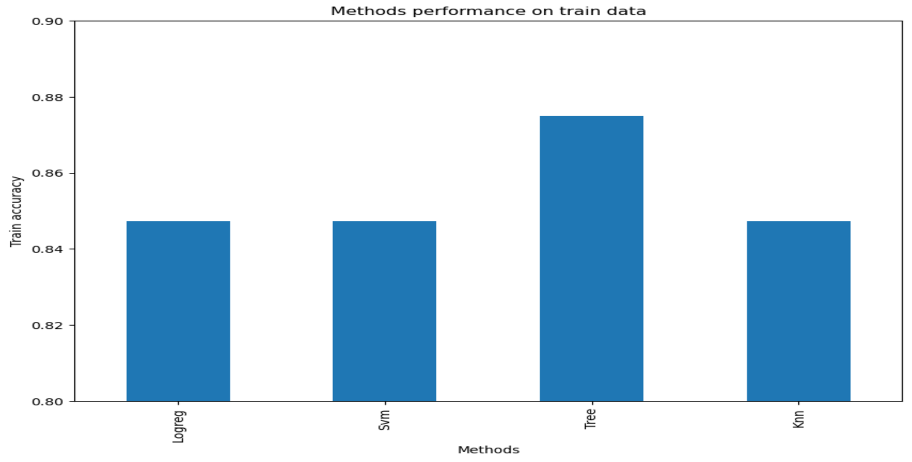
Predictive Analysis (Classification)

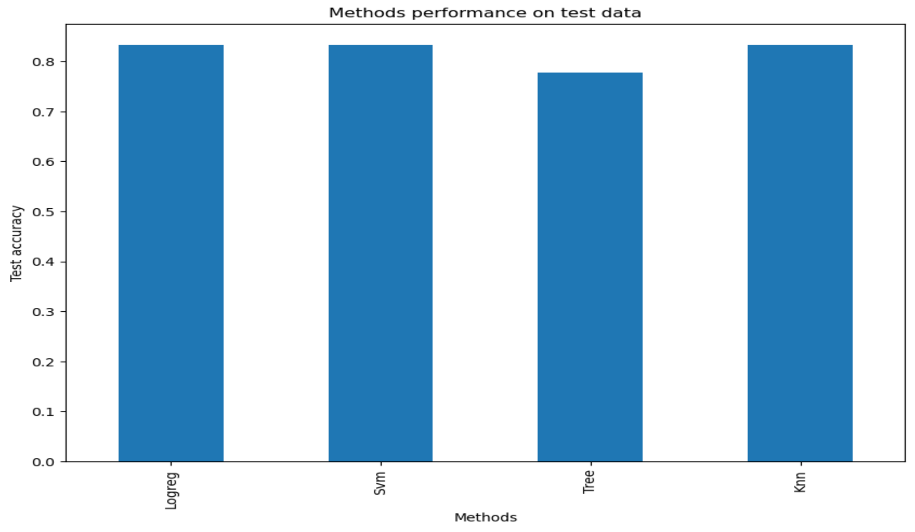
Classification Accuracy

- The model with the highest classification accuracy is Decision Tree Classifier, which has accuracies over than 87%.

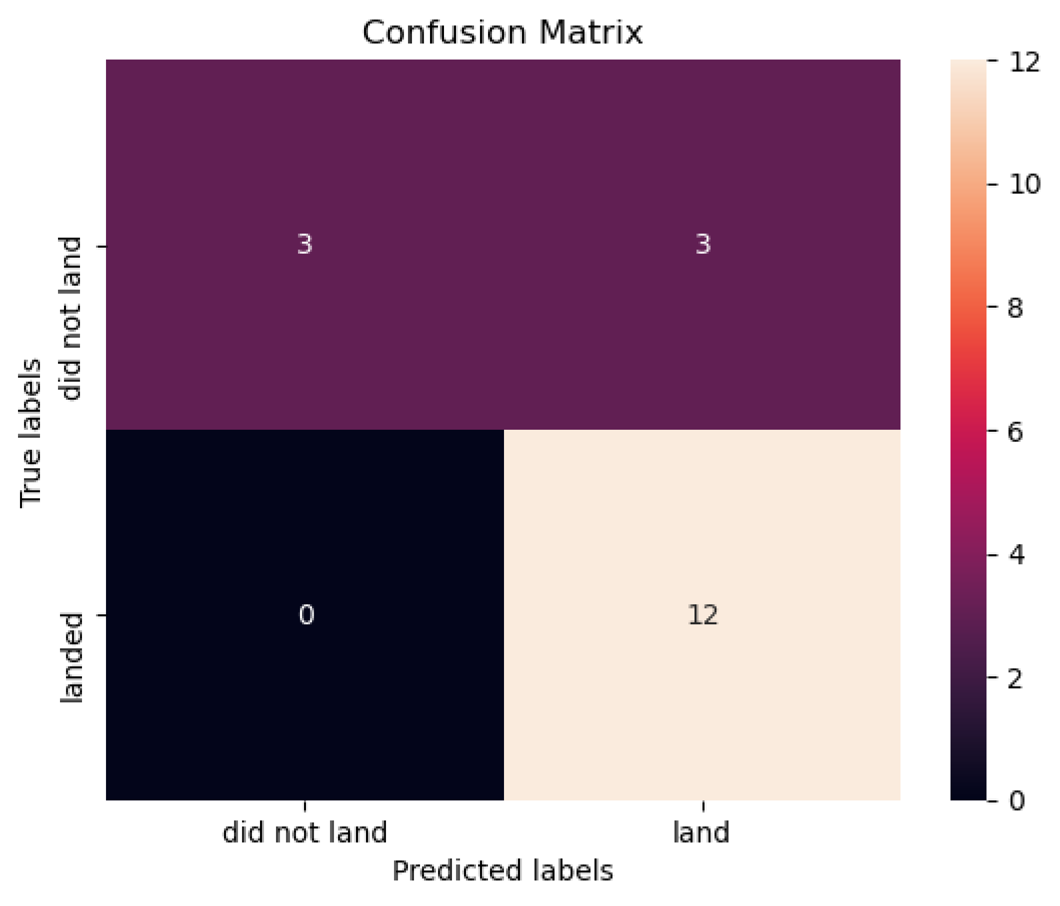


	Accuracy Train	Accuracy Test
Tree	0.875000	0.777778
Logreg	0.847222	0.833333
Svm	0.847222	0.833333
Knn	0.847222	0.833333





Confusion Matrix



- As the test accuracy are all equal, the confusion matrices are also identical.The main problem of these models are false positives.

Conclusions

- Different data sources were analyzed, refining conclusions along the process;
- The best launch site is KSC LC-39A;
- Launches above 7,000kg are less risky;
- Although most of mission outcomes are successful, successful landing outcomes seem to improve over time, according the evolution of processes and rockets;
- Decision Tree Classifier can be used to predict successful landings and increase profits.

Thank you!