IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Álvaro Fernández Villar
20/02/2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection
  - Data Wrangling
  - EDA with Data Visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a Dashboard wit Plotly Dash
  - Predictive analysis (Classification)
- Summary of all results
  - Exploratory Data Analysis Results
  - Interactive Analytics Demo in screenshots
  - Predictive Analysis Results

Introduction

- Project background and context
  - We predict if the Falcon 9 first stage will land successfully. 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.
- Problems you want to find answers
  - What influences if the rocket will land successfully?
  - The effect each relationship with certain rocket variables will impact in determining the success rate of a successful landing
  - What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX Rest API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - One Hot Encoding data fields for Machine Learning and dropping irrelevant columns
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Plotting: Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Methodology

The following datasets was collected by

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

Data Collection – SpaceX API

1. Getting Response from API
2. Converting Response to a .json file
3. Apply custom functions to clean data
4. Assign list to dictionary then dataframe
5. Filter dataframe and export to flat file (.csv)

```
In [52]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [53]: response = requests.get(spacex_url)

In [58]: # Get the head of the dataframe
         data = pd.json_normalize(response.json())
         data.head(5)

In [64]: # Call getLaunchSite
         getLaunchSite(data)

In [65]: # Call getPayloadData
         getPayloadData(data)

In [66]: # Call getCoreData
         getCoreData(data)

In [67]: 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,

In [68]: # Create a data from launch_dict
         data = pd.DataFrame.from_dict(launch_dict)

data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

GITHUB NOTEBOOK URL

Data Collection - Scraping

1. Getting Response from HTML
2. Creating BeautifulSoup Object
3. Finding Tables
4. Getting column names
5. Creation of dictionary
6. Appending data to keys
7. Converting dictionary to dataframe
8. Dataframe to .CSV

```
object = requests.get(static_url)
```

```
soup = BeautifulSoup(object.text)
```

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

```
column_names = []

# Apply find_all() function with `th` element
# Iterate each th element and apply the provi
# Append the Non-empty column name (`if name

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

```
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
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']=[]
```

```
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()
```

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

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

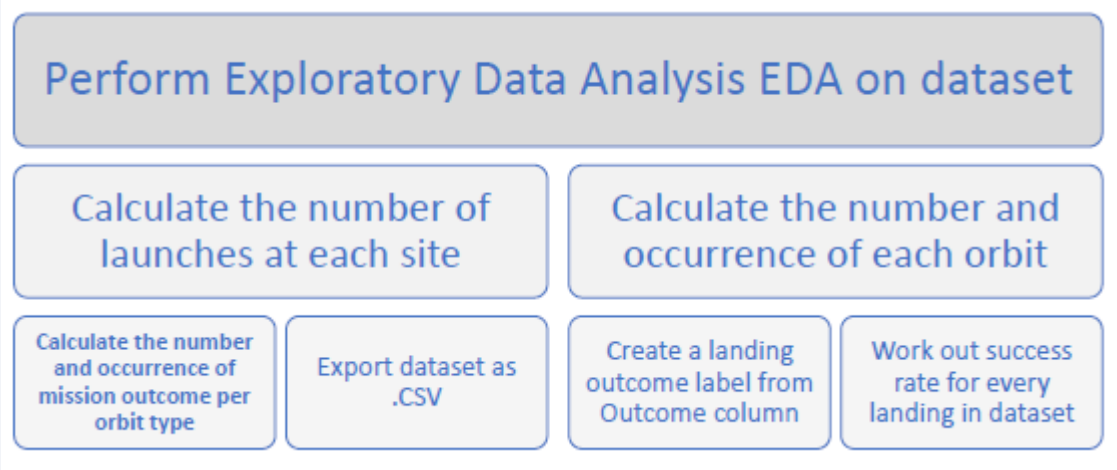
GITHUB NOTEBOOK URL

Data Wrangling

Introduction

In the dataset 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 mean the mission outcome was unsuccessfully landed to a ground pad. True ASDS mean the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship. We mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Process



GITHUB NOTEBOOK URL

EDA with Data Visualization

Scatter Graphs being drawn:

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

Bar Graph being drawn:

- Mean vs Orbit

Line Graph being drawn:

- Success Rate vs Year

GITHUB NOTEBOOK URL

EDA with SQL

- Performed SQL queries to gather information about the dataset.

For example of some questions we were asked about the data we needed information about. Which we are using SQL queries to get the answers in the dataset:

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

GITHUB NOTEBOOK URL

Build an Interactive Map with Folium

- To visualize the Launch Data into an interactive map.

We took the Latitude and Longitude Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.

- We assigned the dataframe launch outcomes(failures, successes) to classes 0 and 1 with Green and Red markers on the map in a MarkerCluster()
- Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find carious trends about what is around the Launch Site to measure patterns. Lines are drawn on the map to measure distance to landmarks.

GITHUB NOTEBOOK URL

Build a Dashboard with Plotly Dash

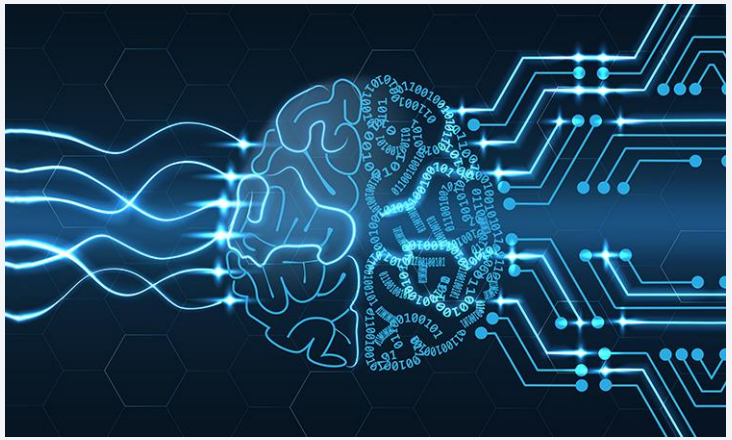
- The dashboard is built with Flask and Dash web framework.
  - Graphs
    - Pie Chart showing the total launches by a certain site/all sites
    - Display relative proportions of multiple classes of data
    - Size of the circle can be made proportional to the total quantity it represents
  - Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster versions
    - It shows the relationship between two variables
    - It is the best method to show you a non-linear pattern
    - The range of data flow can be determined
    - Observation and reading are straightforward

GITHUB NOTEBOOK URL

Predictive Analysis (Classification)

- BUILDING MODEL
  - Load our dataset into Numpy and Pandas
  - Transform Data
  - Split our data into training and test data sets
  - Check how many test samples we have
  - Decide which type of machine learning algorithms we want to use
  - Set out parameters and algorithms to GridSearchCV
  - Fit our datasets into the GridSearchCV objects and train our dataset
- EVALUATING MODEL
  - Chech accuracy for each model
  - Get tuned hyperparameters for each type of algorithms
  - Plot Confusion Matrix
- IMPROVING MODEL
  - Feature Engineering
  - Algorithm Tuning

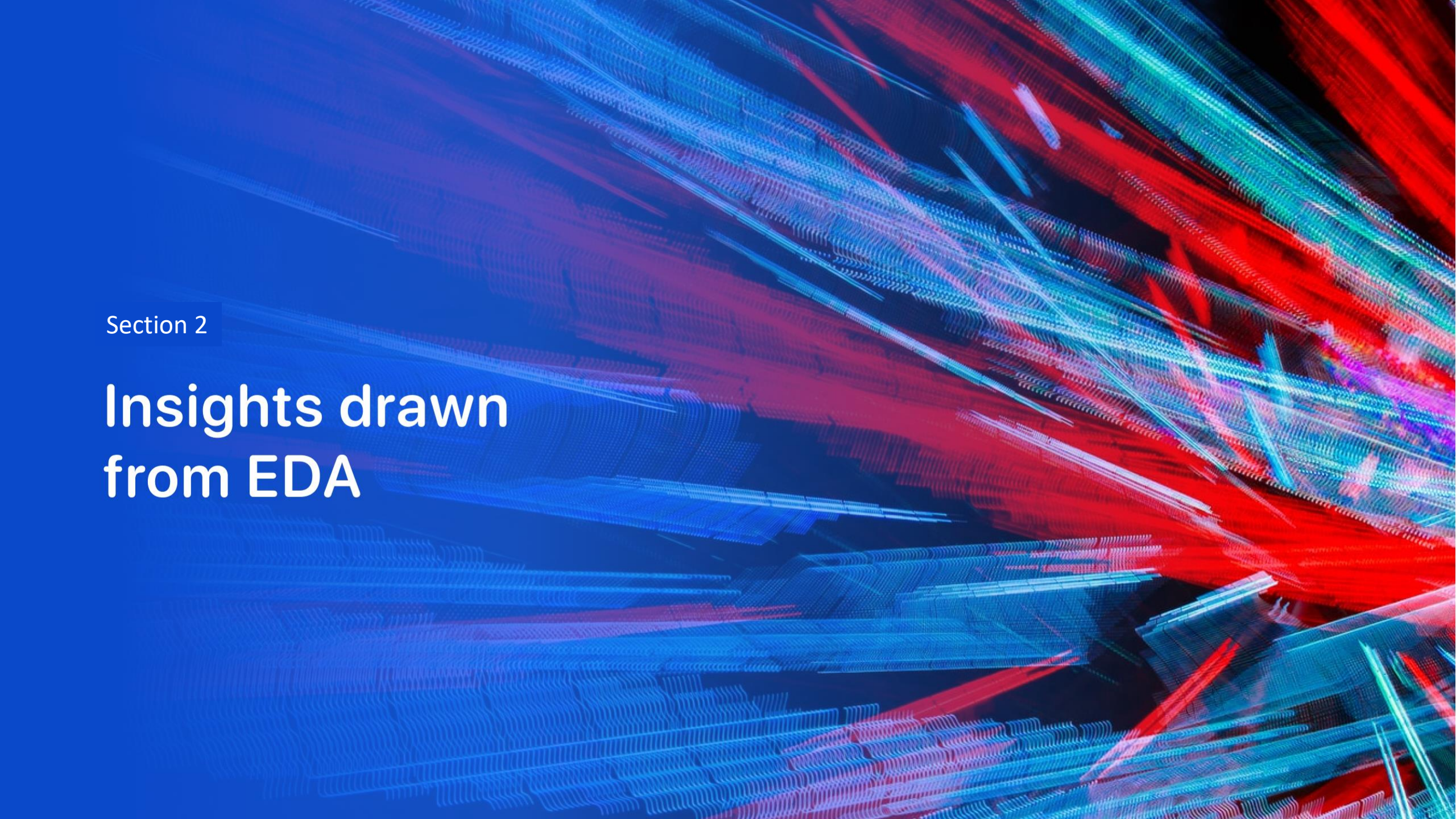
- FINDING THE BEST PERFORMING CLASSIFICATION MODEL
  - The model with the best accuracy score wins the best performing model
  - In the notebook there is a dictionary of algorithms with scores at the bottom of the notebook



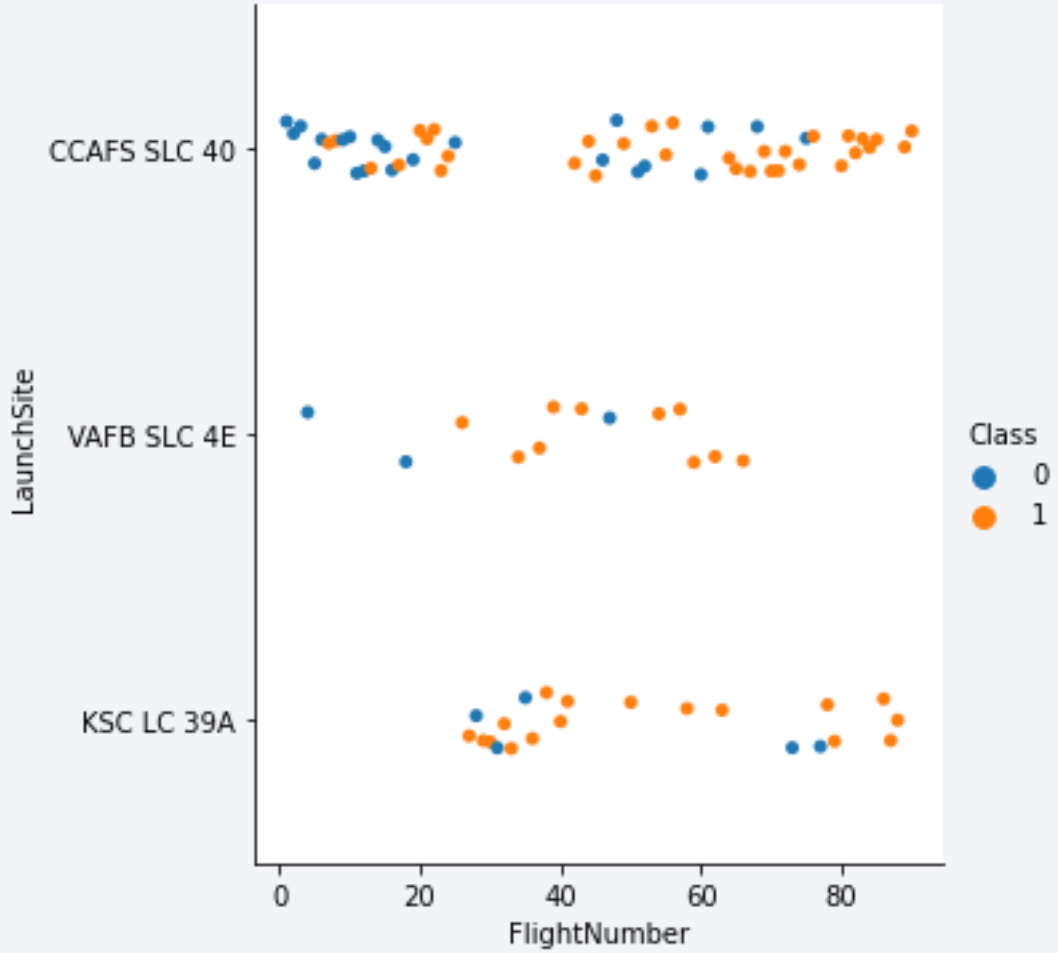
GITHUB NOTEBOOK URL

Results

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

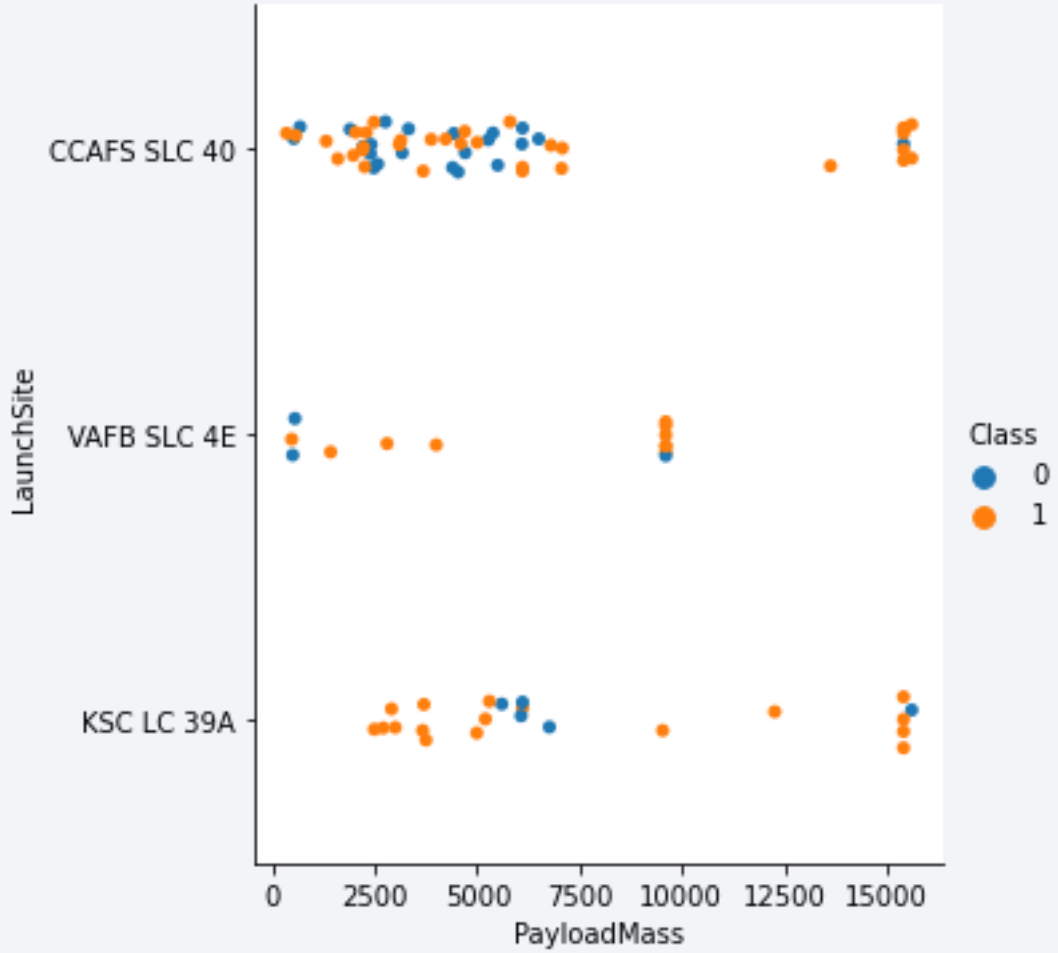

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



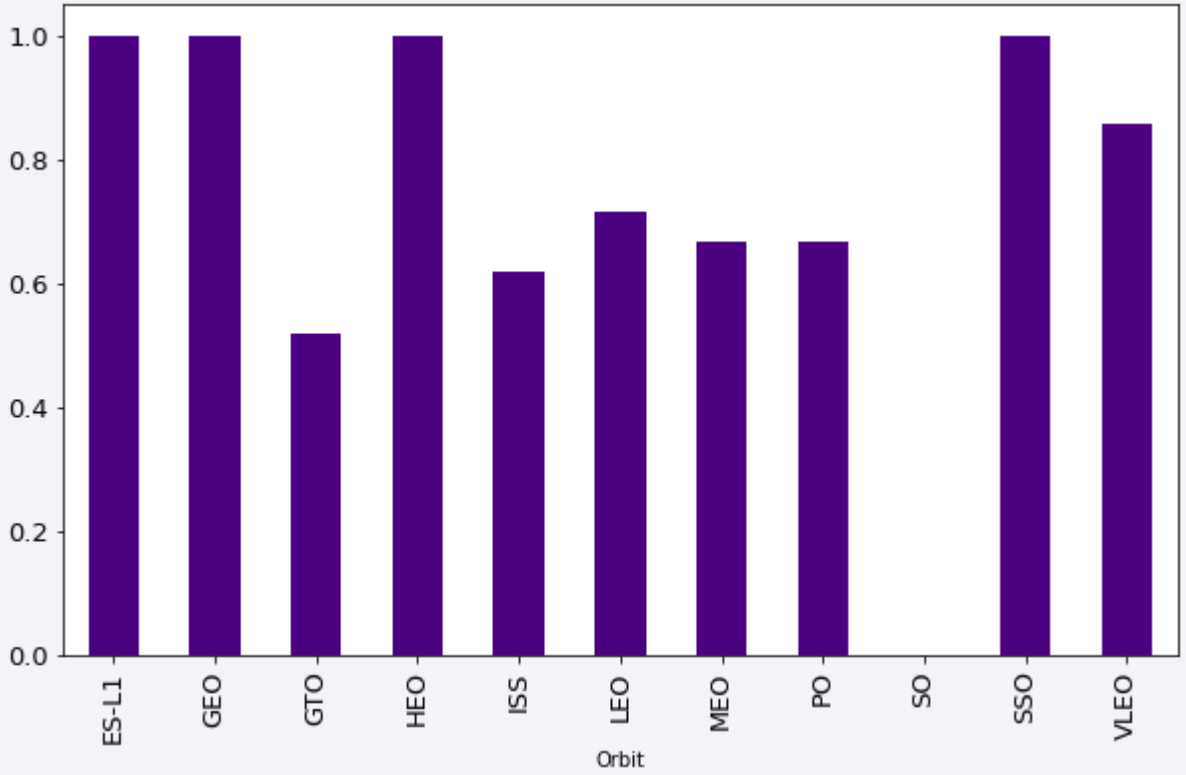
The more amount of flights at a launch site the greater the success rate at a launch site.

Payload vs. Launch Site



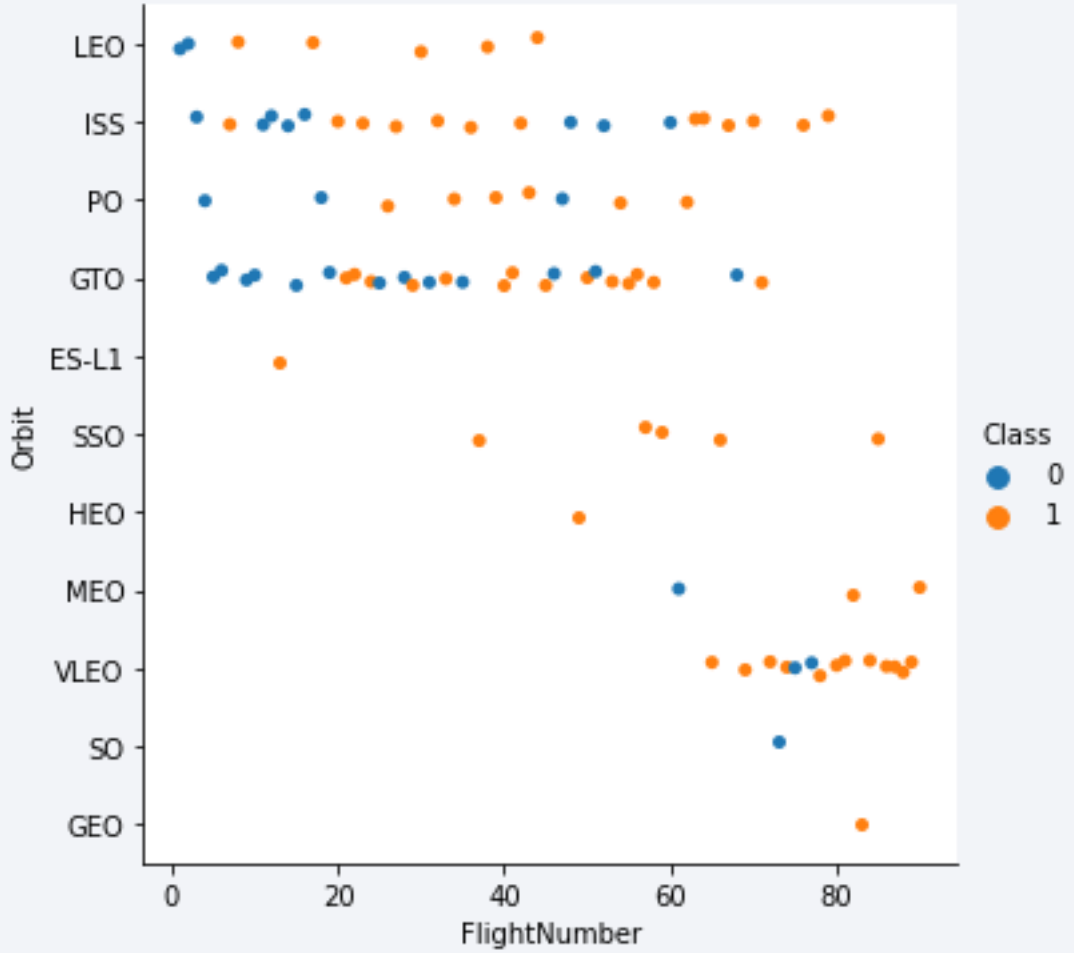
- *The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket.*
- *There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependant on Pay Load Mass for a success launch.*

Success Rate vs. Orbit Type



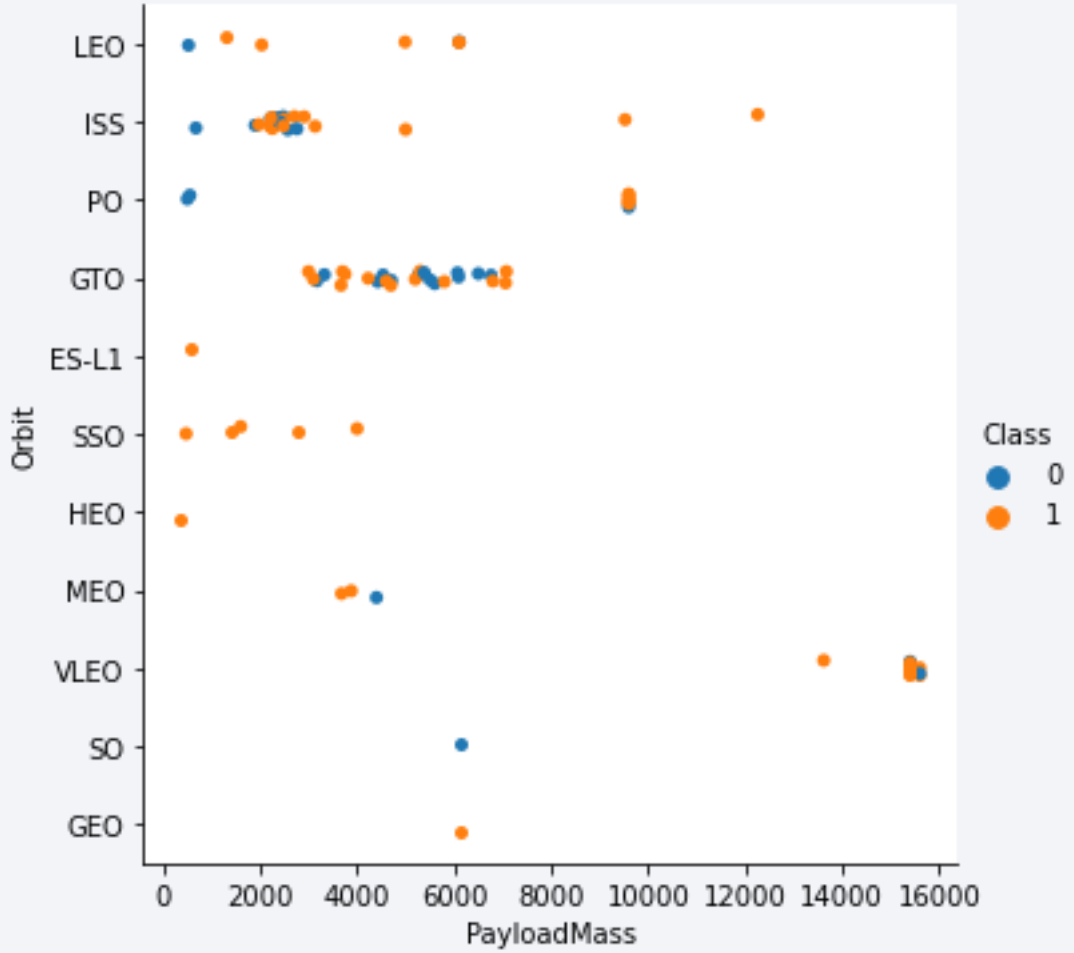
Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

Flight Number vs. Orbit Type



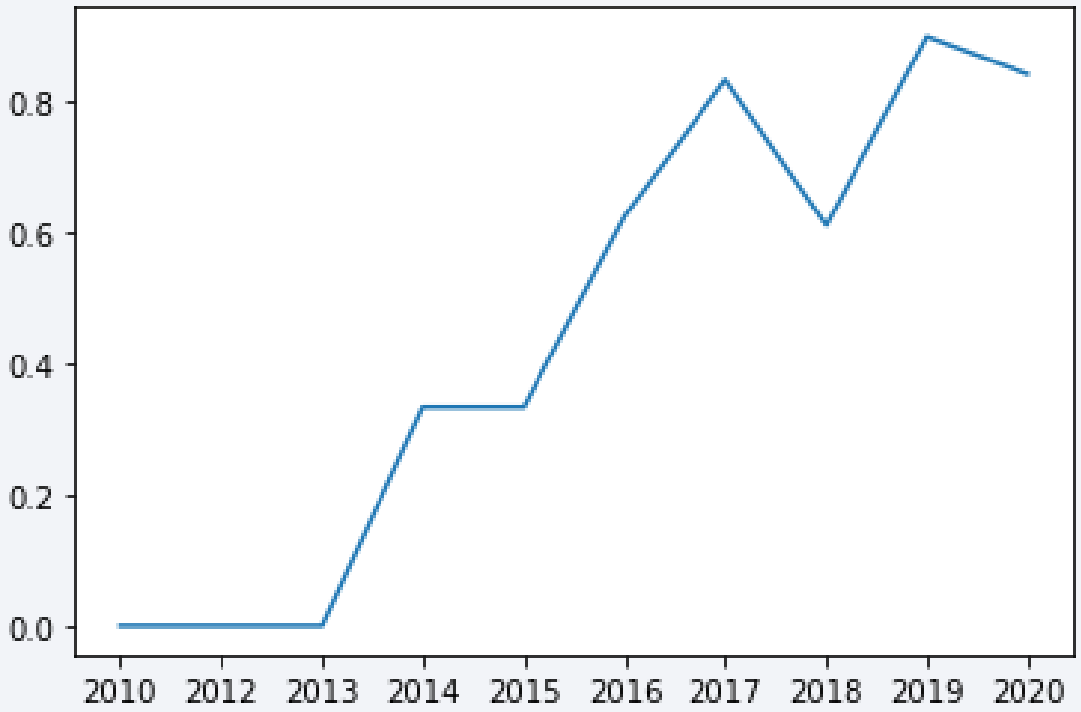
You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit

Payload vs. Orbit Type



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

Launch Success Yearly Trend



You can observe that the success rate since 2013 kept increasing till 2020

All Launch Site Names

SQL QUERY

```
%sql select distinct LAUNCH_SITE
from SPACEXTBL
```

RESULT

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

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

Launch Site Names Begin with 'CCA'

SQL QUERY

```
%sql select * from SPACEXTBL where
launch_site LIKE 'CCA%' limit 5
```

Using the word ***limit 5*** in the query means that it will only show 5 records from ***tblSpaceX*** and ***LIKE*** keyword has a wild card with the words ***'CCA%'*** the percentage in the end suggests that the Launch_Site name must start with CCA.

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

SQL QUERY

```
%sql select sum(payload_mass__kg_)
as SUM_PAYLOAD_MASS from SPACEXTBL
where CUSTOMER LIKE 'NASA%'
```

sum_payload_mass
99980

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

Average Payload Mass by F9 v1.1

SQL QUERY

```
%sql select
avg(payload_mass__kg_) as
AVG_PAYLOAD_MASS from SPACEXTBL
where BOOSTER_VERSION LIKE 'F9
v1.1%'
```

avg_payload_mass
2534

Using the function ***AVG*** works out the average in the column ***PAYLOAD_MASS_KG_***

The ***WHERE*** clause filters the dataset to only perform calculations on ***Booster_version F9 v1.1***

First Successful Ground Landing Date

SQL QUERY

```
%sql select DATE from SPACEXTBL
where landing__outcome LIKE
'%ground pad%' limit 1
```

DATE
2015-12-22

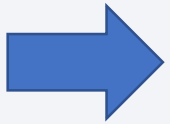
Using the function ***LIMIT*** works out the minimum date in the column ***Date***

The ***WHERE*** clause filters the dataset to only perform calculations on ***Landing_Outcome ground pad***

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL QUERY

```
%sql select distinct
booster_version from SPACEXTBL
where landing__outcome LIKE '%drone
ship%' and payload_mass__kg_
between 4000 and 6000
```



booster_version
F9 FT B1021.2
F9 FT B1031.2
F9 FT B1020
F9 FT B1022
F9 FT B1026

Selecting only ***Booster_Version***

The ***WHERE*** clause filters the dataset to ***Landing_Outcome = Success (drone ship)***

The ***AND*** clause specifies additional filter conditions ***Payload_MASS_KG_*** > 4000 AND ***Payload_MASS_KG_ < 6000***

Total Number of Successful and Failure Mission Outcomes

SQL QUERY

```
%sql SELECT(SELECT
Count(Mission_Outcome) from tblSpaceX
where Mission_Outcome LIKE '%Success%')
as Successful_Mission_Outcomes,
(SELECT Count(Mission_Outcome) from
tblSpaceX where Mission_Outcome LIKE
'%Failure%') as
Failure_Mission_Coutcomes
```

	Successful_Mission_Outcomes	Failure_Mission_Outcomes
0	100	1

A much harder query I must say, we used subqueries here to produce the results.

The ***LIKE '%foo%'*** wildcard shows that in the record the ***foo*** phrase is in any part of the string in the records for example.

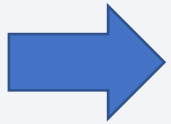
PHRASE "(Drone Ship was a Success)"
LIKE '%Success%'

Word 'Success' is in the phrase the filter will include it in the dataset

Boosters Carried Maximum Payload

SQL QUERY

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



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

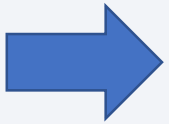
Using the word ***DISTINCT*** in the query means that it will only show Unique values in the ***Booster_Version*** column from ***tblSpaceX***

*Using a **subquery** with the **Where** clause we select only the booster_version with the **max** payload_mass_kg*

2015 Launch Records

SQL QUERY

```
%sql select launch_site,
booster_version from SPACEXTBL where
landing__outcome LIKE '%drone ship%'
and year(DATE) = 2015
```



launch_site	booster_version
CCAFS LC-40	F9 v1.1 B1012
CCAFS LC-40	F9 v1.1 B1015
CCAFS LC-40	F9 v1.1 B1018

Using the word ***where and LIKE*** in the query means that it will select only records with the string “drone ship” in the variable landing_outcome.

Using year we can filter the year in the variable date.

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

SQL QUERY

```
%sql select date,
landing__outcome from SPACEXTBL
where landing__outcome LIKE
'Failure (drone ship)' or
landing__outcome LIKE 'Success
(ground pad)' and DATE between
'2010-06-04' and '2017-03-20'
order by date desc
```

DATE	landing__outcome
2017-02-19	Success (ground pad)
2016-07-18	Success (ground pad)
2016-06-15	Failure (drone ship)
2016-03-04	Failure (drone ship)
2016-01-17	Failure (drone ship)
2015-12-22	Success (ground pad)
2015-04-14	Failure (drone ship)
2015-01-10	Failure (drone ship)

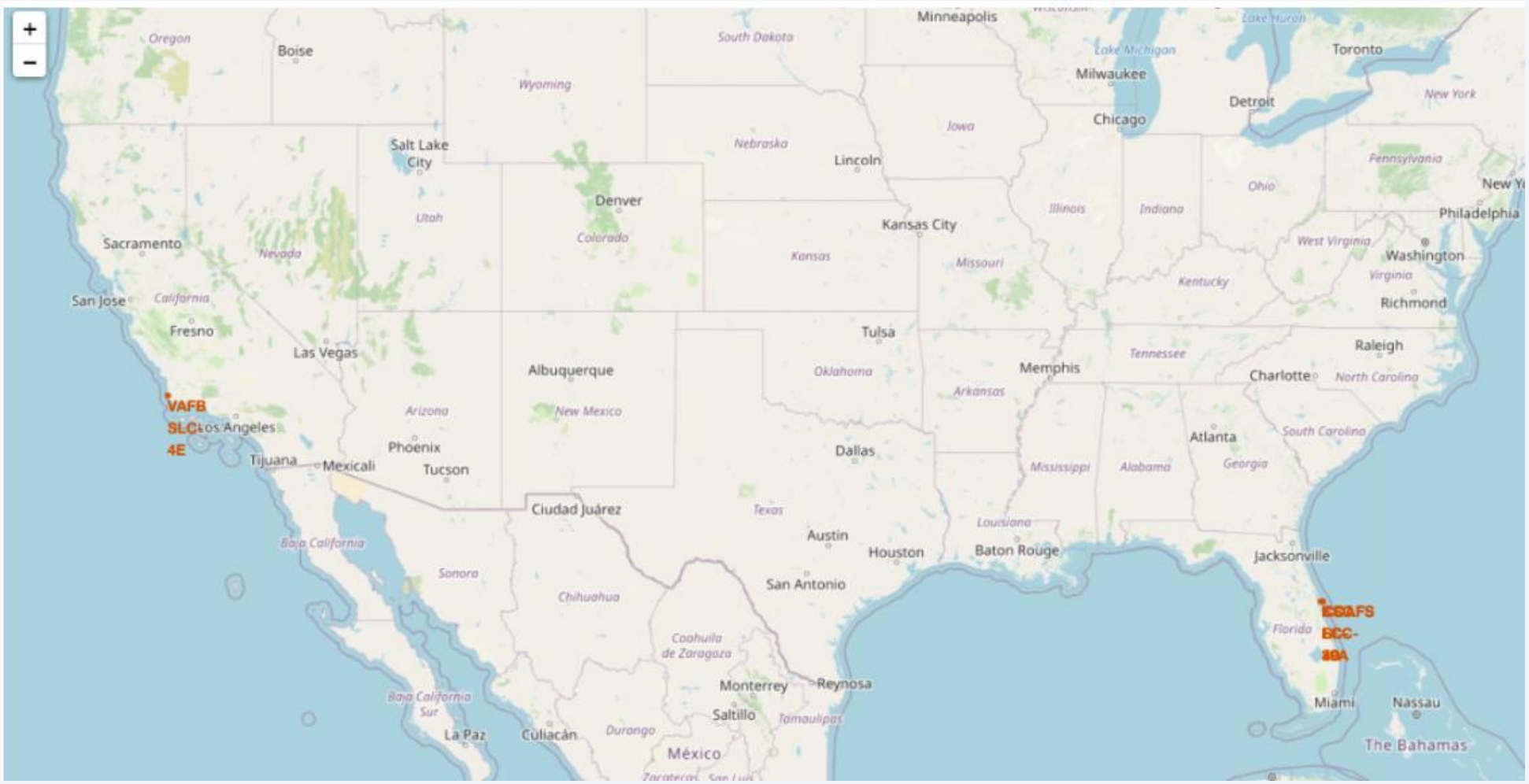
Using the word ***where, LIKE and OR*** in the query means that it will select only records with the string “Failure (drone ship) ” or “Success (ground pad) in the variable landing_outcome.

Using the word ***AND*** *and* ***BETWEEN*** add one more condition of filter in the WHERE clause by date.

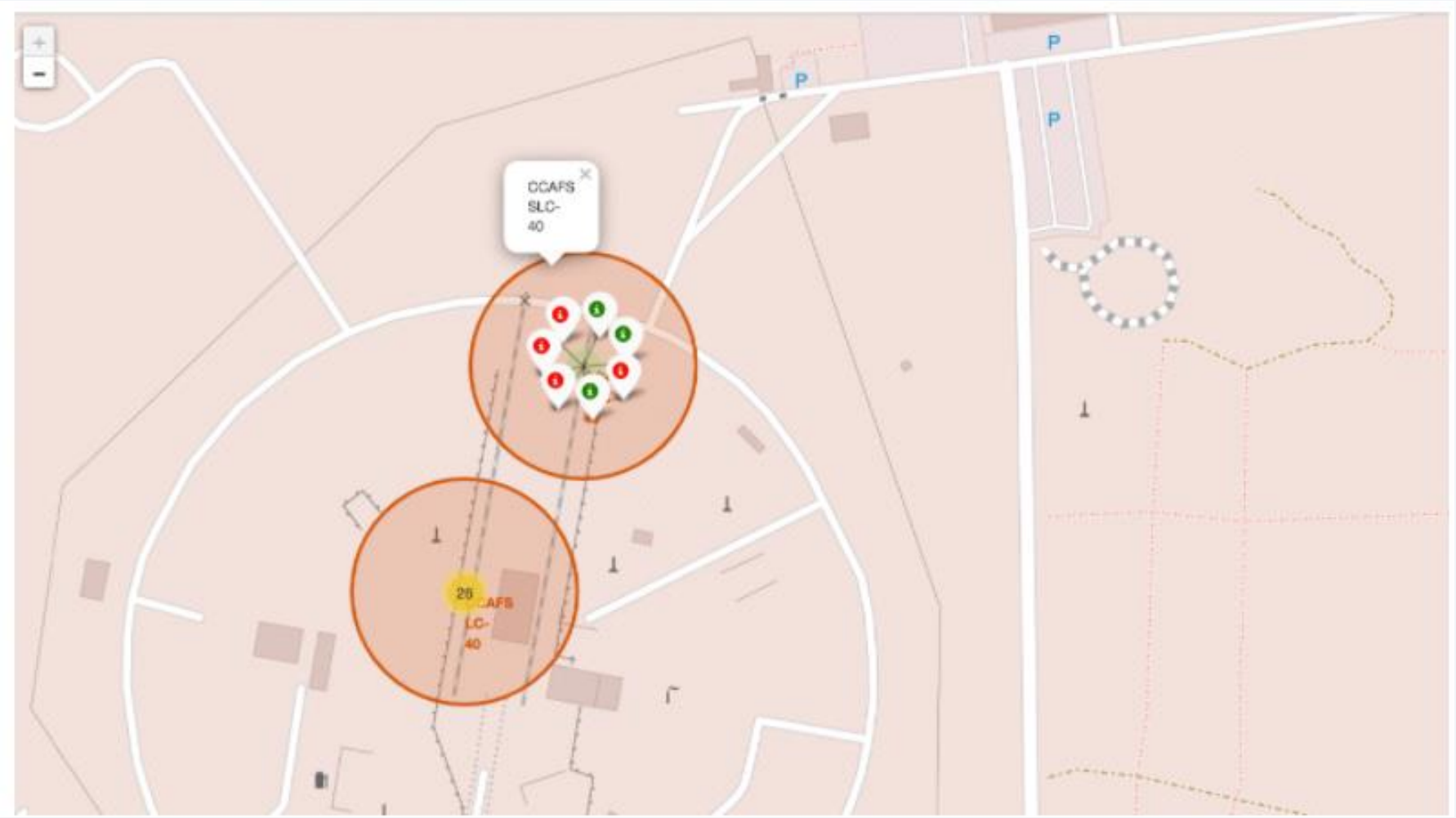
Section 3

Launch Sites Proximities Analysis

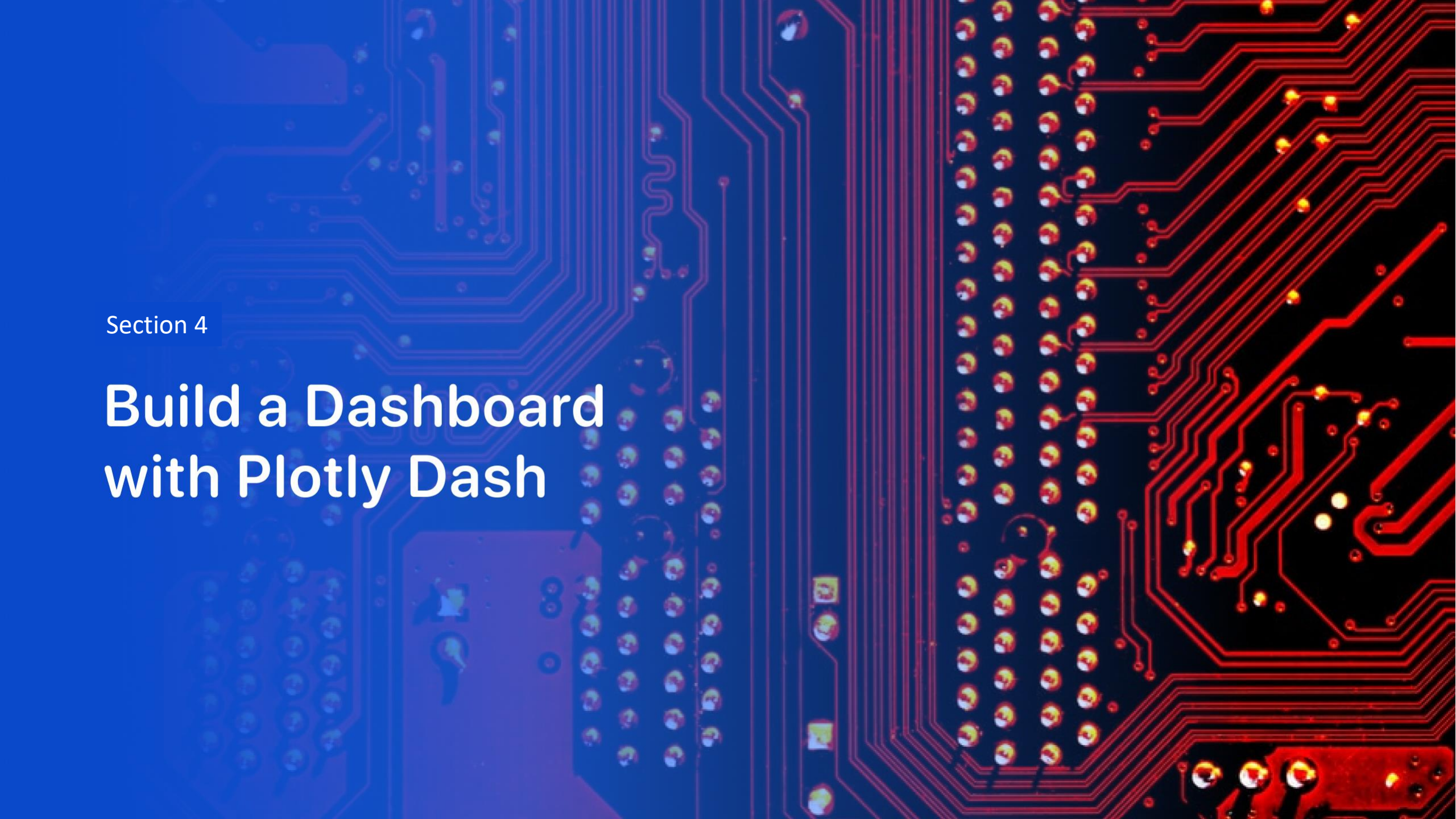
All launch sites global map markers



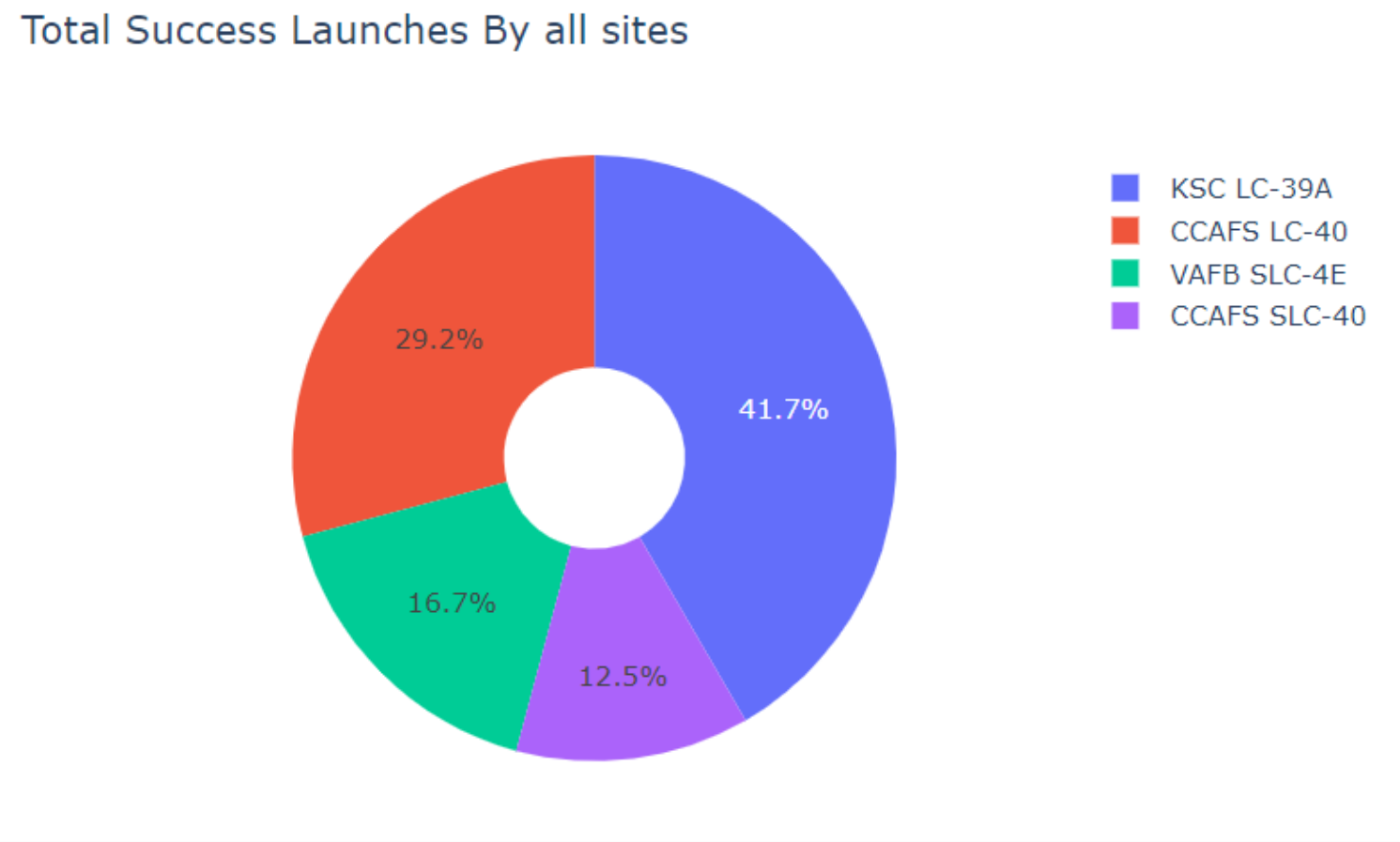
Mark the success/failed launches for each site on the map



Calculate the distances between a launch site to its proximities

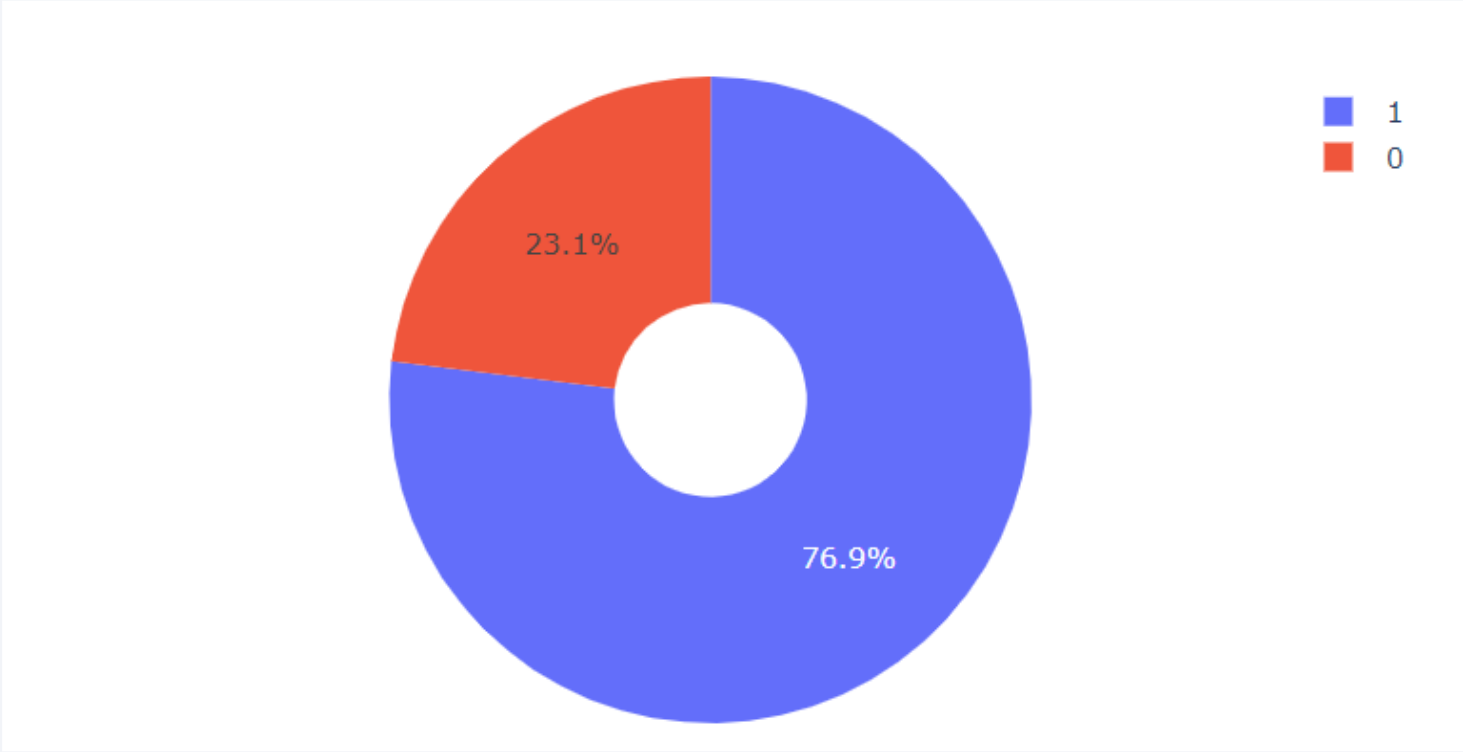

Section 4
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



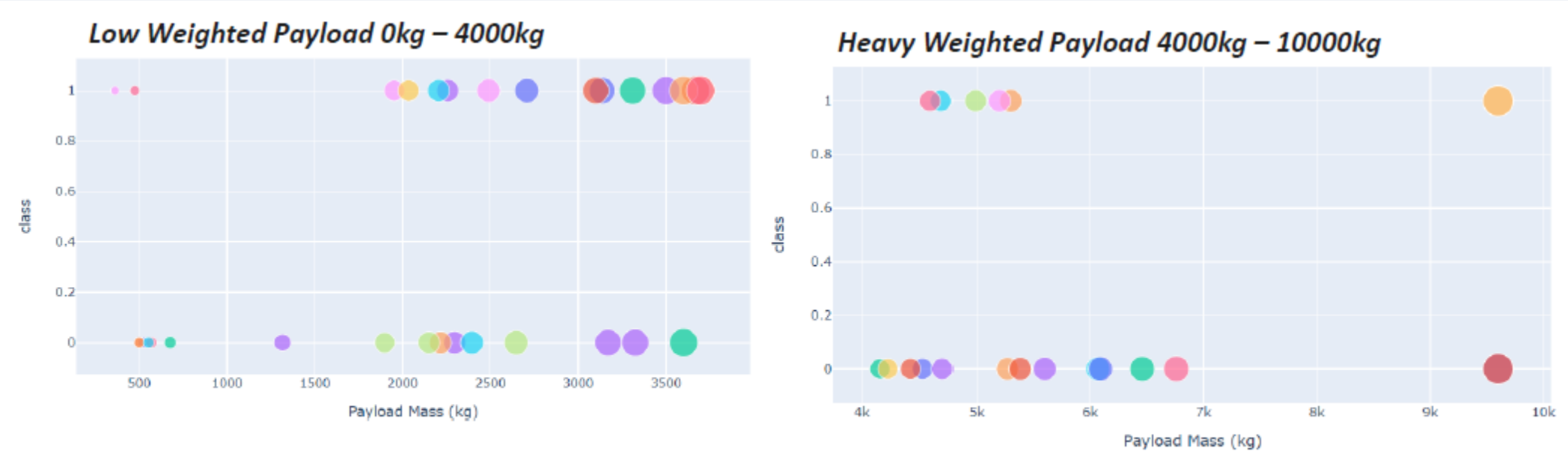
We can see that KSC LC-39A had the most successful launches from all the sites

Pie chart for the launch site with highest launch success ratio



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

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



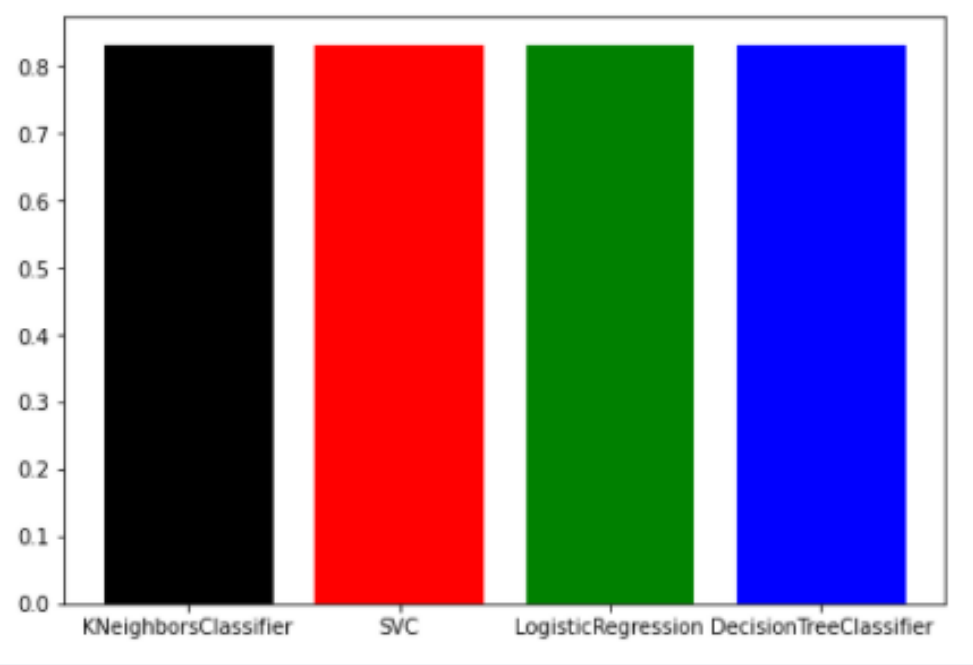
We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

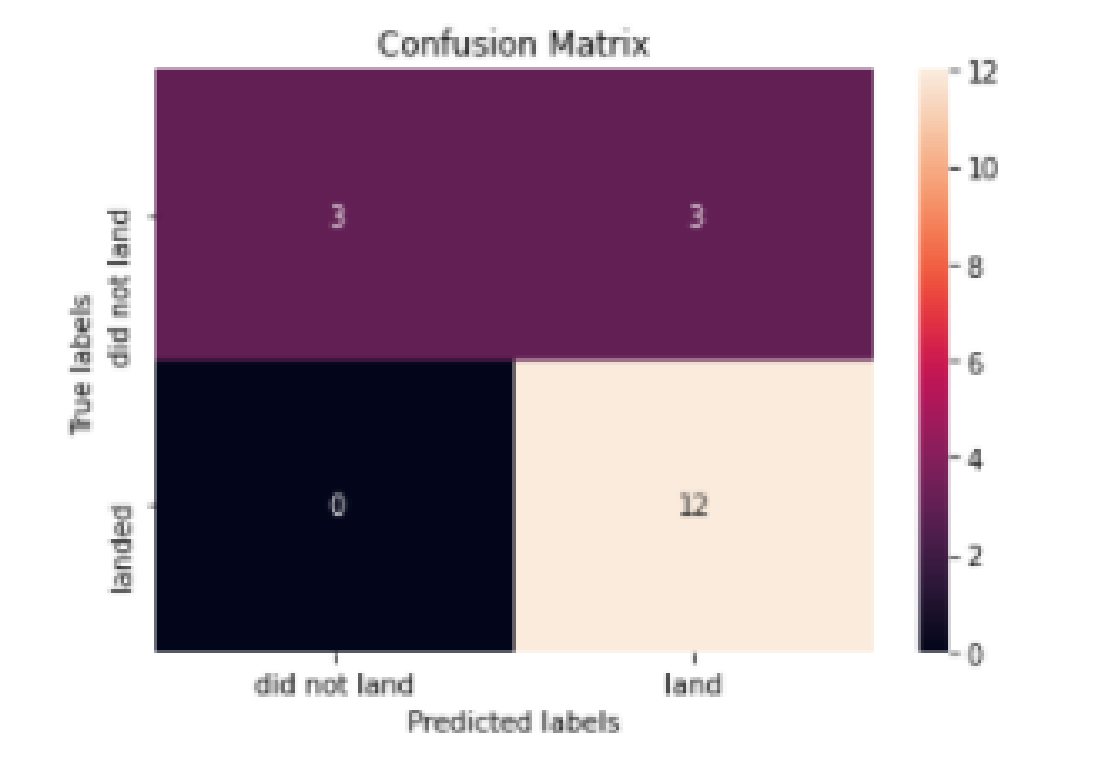
Predictive Analysis (Classification)

Classification Accuracy

After selecting the best hyperparameters for the algorithms using the validation data, we achieved 83.33% accuracy on the test data for all of them. The four algorithms are just as accuracy.



Confusion Matrix



We can see the major problem is the False Positives. The algorithm classify three samples as land when really they did not land

Conclusions

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

Thank you!