Winning Space Race with Data Science

Falowo Gbolahan

14-09-2021

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 with Plotly Dash
- Predictive analysis (Classification)

❑ Summary of all results

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

Introduction

❑ Project background

- We predicted 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

❑ Data collection methodology:

- SpaceX Rest API
- (Web Scrapping) from Wikipedia

- Performed data wrangling (Transforming data for Machine Learning)

❑ One Hot Encoding data fields for Machine Learning and dropping irrelevant columns

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

❑ Plotting : Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data.

- Performed interactive visual analytics using Folium and Plotly Dash
- Performed predictive analysis using classification models

➢ How to build, tune, evaluate classification models

Data Collection – SpaceX API

1.Getting Response from API

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

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

2.Converting to json format

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

3.Cleaning data

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

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

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

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

4. Create dataframe from 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}
```

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

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

5.Filter and Export to csv file

```
data_falcon_9 = df.loc[df['BoosterVersion']!="Falcon 1"]
data_falcon_9.to_csv('dataset_part_1.csv', index=False)
```

[Github]

Data Collection - Scraping

1.Receiving response from html

```
page = requests.get(static_url)
page.status_code
```

```
200
```

2.Creating beautiful soup instance

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

3.Finding the tables

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

4.Get column names

```
names_of_columns = []
# 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
th_elements = soup.find_all('th')
for x in range(len(th_elements)):
    try:
        name = extract_column_from_header(th_elements[x])
        if (name is not None and len(name) > 0):
            names_of_columns.append(name)
    except:
        pass
```

5.Dictionary creation

```
launch_dict= dict.fromkeys(names_of_columns)

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

# Let's initial the launch_dict with each va
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
```

6.Appending of Data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('tal
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as nu
        if rows.th:
            if rows.th.string:
```

7.Converting to dataframe and then csv file

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

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

[Github]

Data Wrangling

Introduction

In our data set we have several different cases where the booster does not land successfully. Sometimes landing has been attempted but failed due to an accident; Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean.

True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad . True ASDS means the mission outcome was successfully landed to a drone ship False ASDS means the mission outcome was unsuccessfully landed to a drone ship. None ASDS and None None these represent a failure to land.

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.

1.Calculate the number of launches on each site

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

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

2.Calculate number and occurrence of each orbit

```
# Apply value_counts on Orbit column
df["Orbit"].value_counts("Orbit")
df["Orbit"].value_counts()
```

```
GTO      27
ISS      21
VLEO     14
```

3.Calculate mission outcome per orbit type

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

```
True ASDS      41
None None      19
True RTLS      14
```

4.Landing Outcome label

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for key,value in df["Outcome"].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

5.Exporting por datatset as a csv file

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

Github

EDA with Data Visualization

Bar Graph being drawn:

Mean VS. Orbit

A bar diagram makes it easy to compare sets of data between different groups at a glance.
The graph represents categories on one axis and a discrete value in the other.
The goal is to show the relationship between the two axes.
Bar charts can also show big changes in data over time.

Line Graph being drawn:

Success Rate VS. Year

Line graphs are good in representing data variables and trends very clearly and can help to make predictions about the results of data not yet recorded .

Scatter Graphs being drawn:

Flight Number VS. Launch Site

Flight Number VS. Payload Mass

Payload VS. Launch Site

Payload VS. Orbit Type

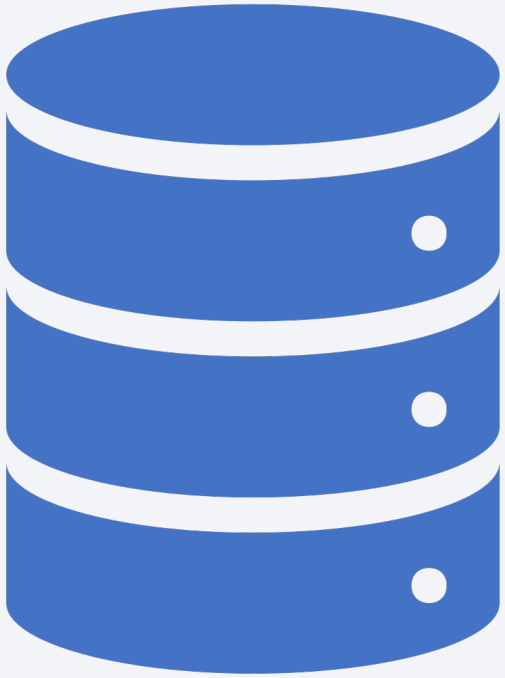
Orbit VS. Payload Mass

Orbit VS. Flight Number

Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation . Scatter plots usually consist of a large body of data.

[Github]

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 average payload mass carried by booster version F9 v1.1
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster
- Ranking the count of successful landing outcomes between the date 2010-06-04 and 2017-03-20 in descending
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the booster versions which have carried the maximum payload mass.
- versions, launch_site for the months in year 2017

Github

Build an Interactive Map with Folium

We added a circle marker around each launch site with a a label of the name of the launch sit to represent the launch data into an interactive map.

Represented launch outcome successes with green markers and failures with red markers for easy identification on the map.

To find the distance from launch sites to landmarks we use the Haversines formula to observe different trends about what we have around the launch site. Blue lines have been drawn to highlight distances between entities.

After you plot distance lines to the proximities, you can answer the following questions easily:

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

Github

Build a Dashboard with Plotly Dash

Graphs

- Pie Chart showing the total launches by sites.
- Display relative proportions of multiple classes of data. - size of the circle can be made proportional to the total quantity it represents.

A Scatter Graph was used to highlight the relationship with Payload Mass (Kg) and Outcome for the different Booster Versions

- It shows the relationship between two given variables.
- A very good way to communicate a non-linear pattern
- It highlights the data flow.(the min and max values can be determined)
- Reading and observations are easy to spot.

Github

Predictive Analysis (Classification)

- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

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 our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train our dataset.

EVALUATING MODEL

- Check accuracy for each model
- Get tuned hyperparameters for each type of algorithms
- Plot Confusion Matrix

IMPROVING MODEL

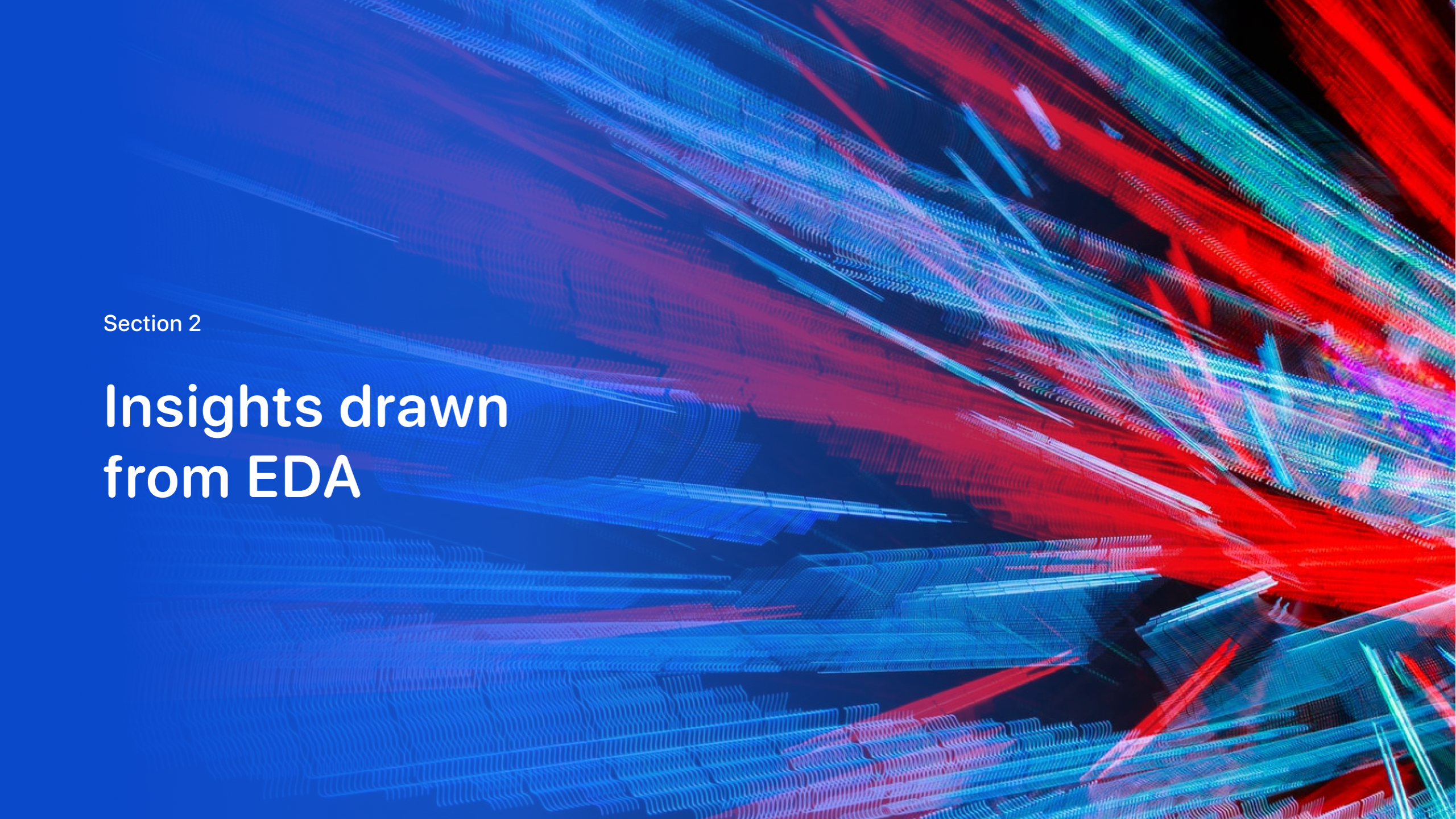
- Feature Engineering
- Algorithm Tuning

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.

Results

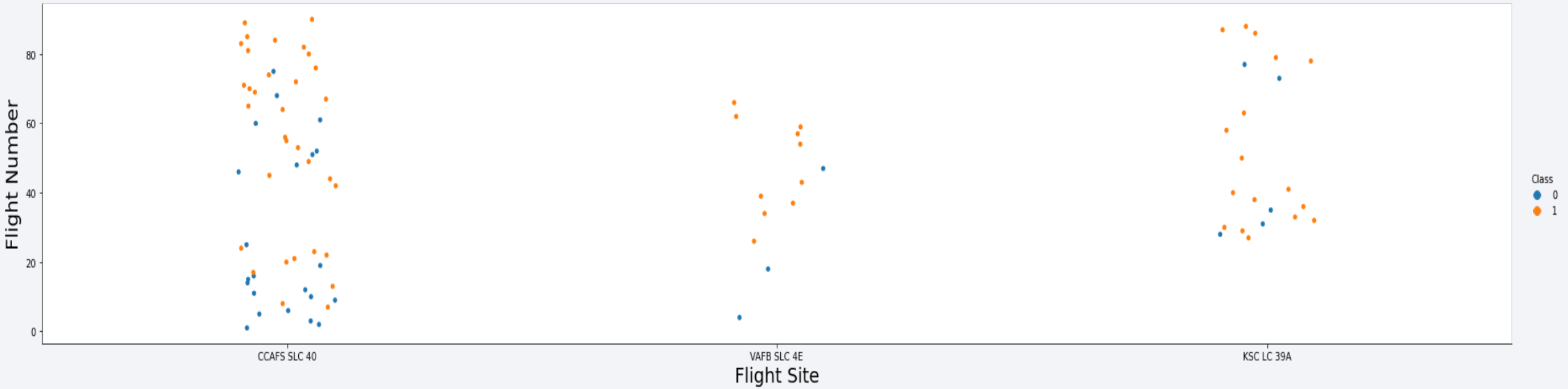
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

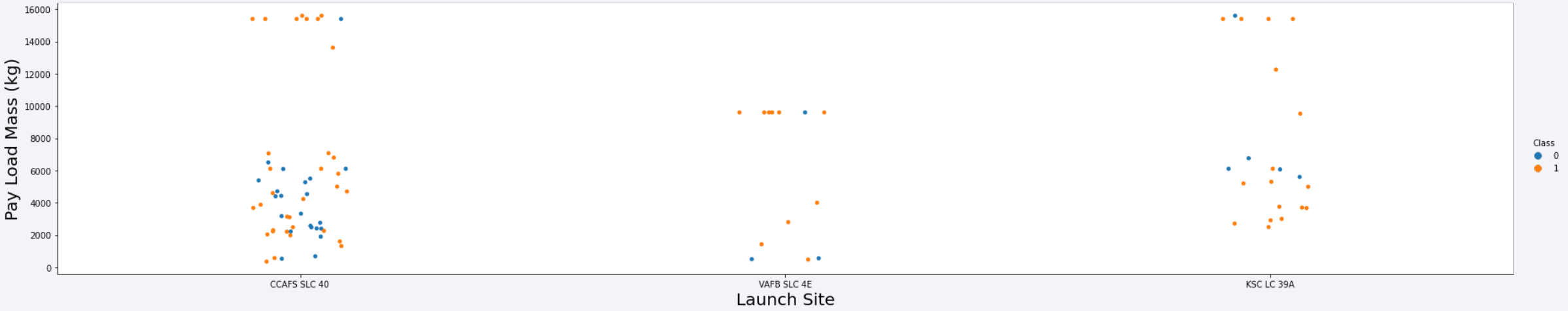
Insights drawn from EDA

Flight Number vs. Launch Site



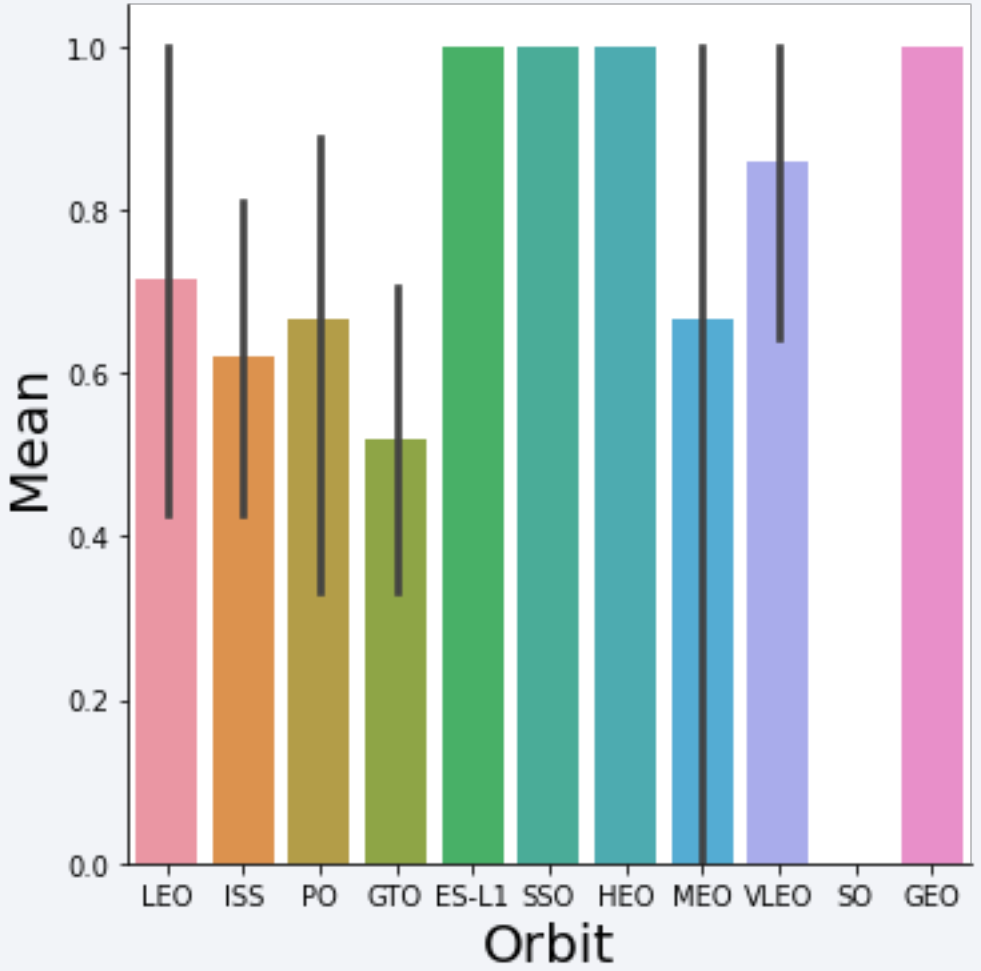
The increase in the amount of flights at a launch site the the higher succes rate at a launch site

Payload vs. Launch Site



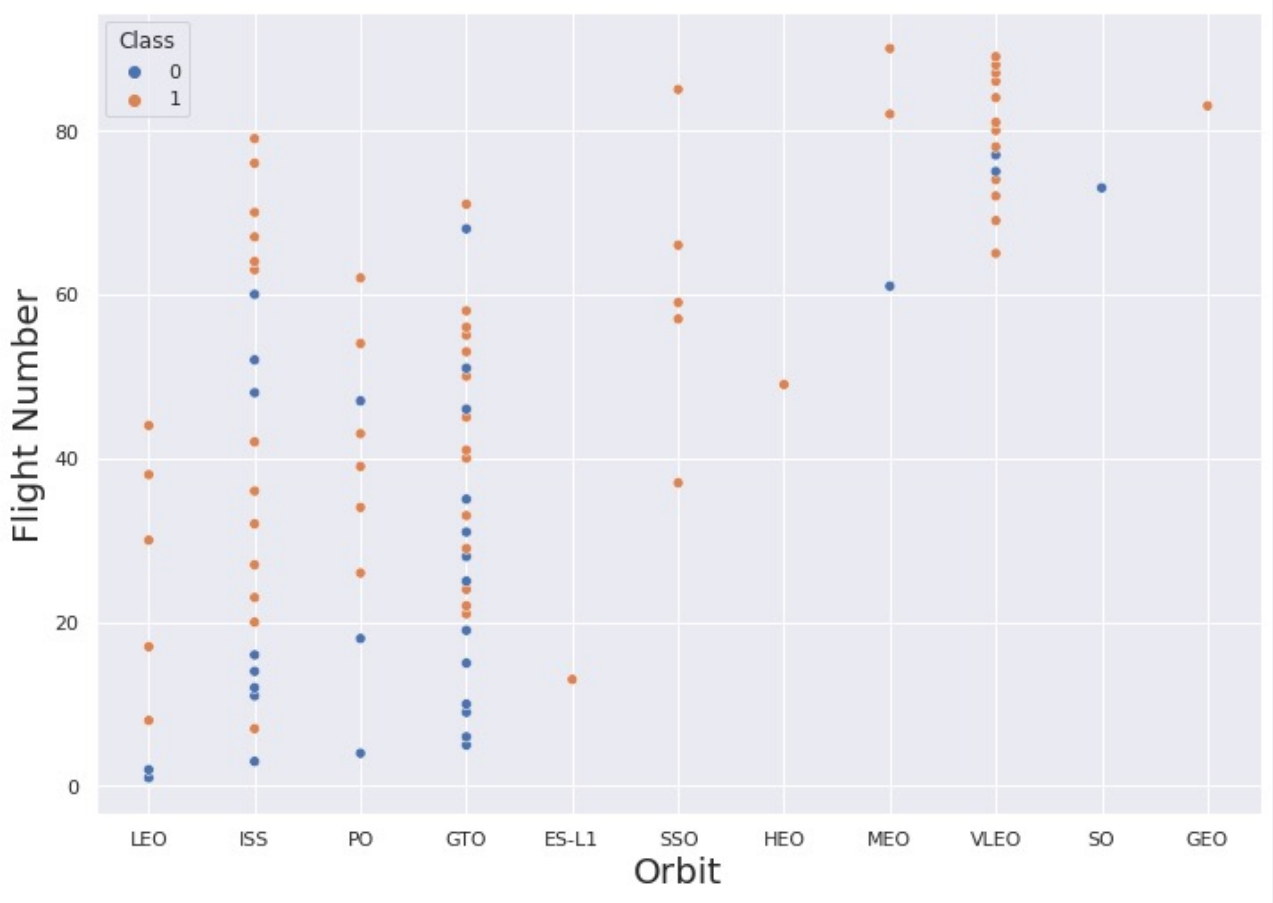
The increase in payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket.
A clear pattern has not been found using this visualization to infer that the Launch Site is dependant on the Pay Load Mass for a success launch.

Success Rate vs. Orbit Type



Orbit GEO,HEO,SSO,ES-L1 have the best Success Rates

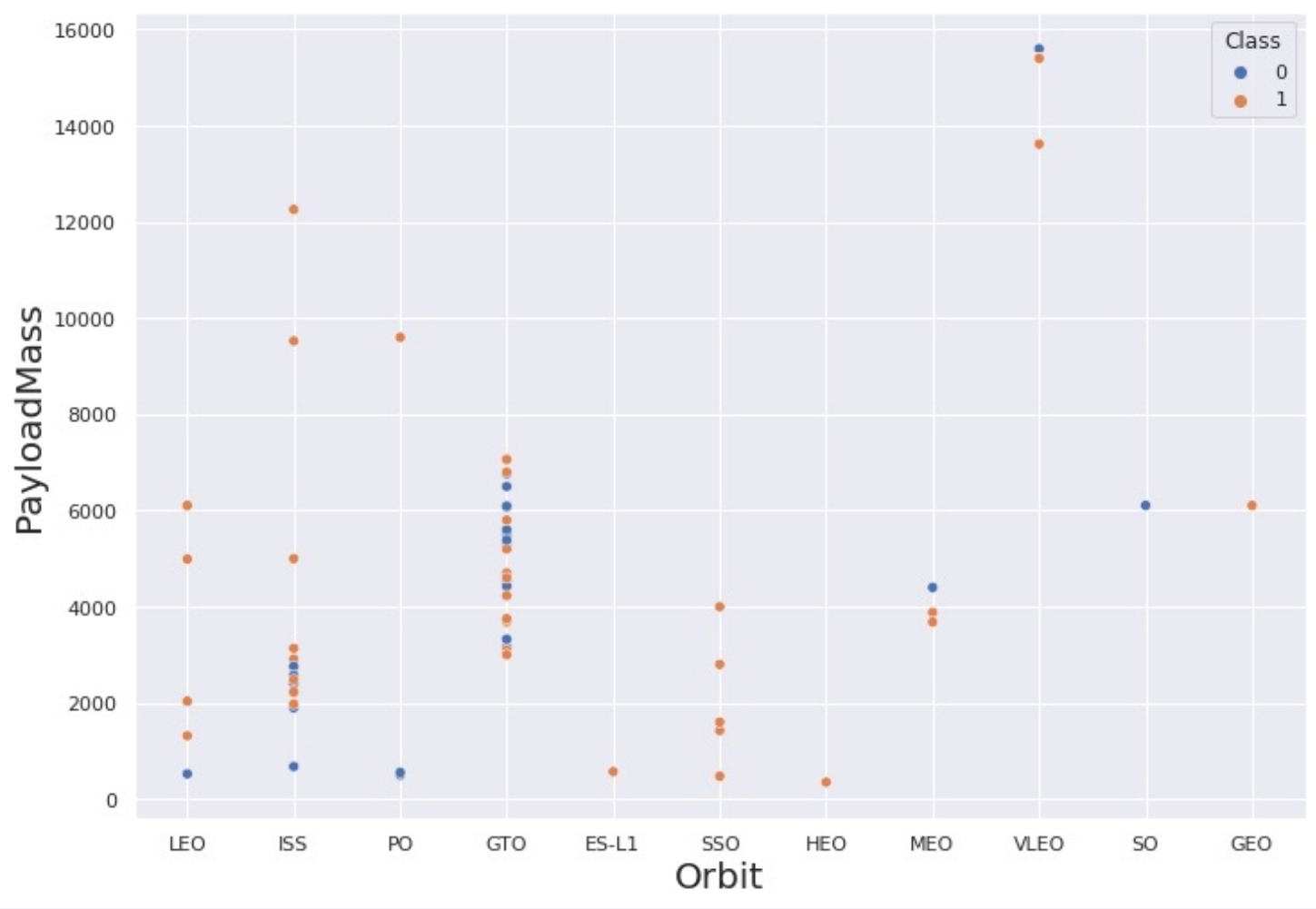
Flight Number vs. Orbit Type



Observations indicate that in the LEO orbit the Success rate appears to relate to the number of flights

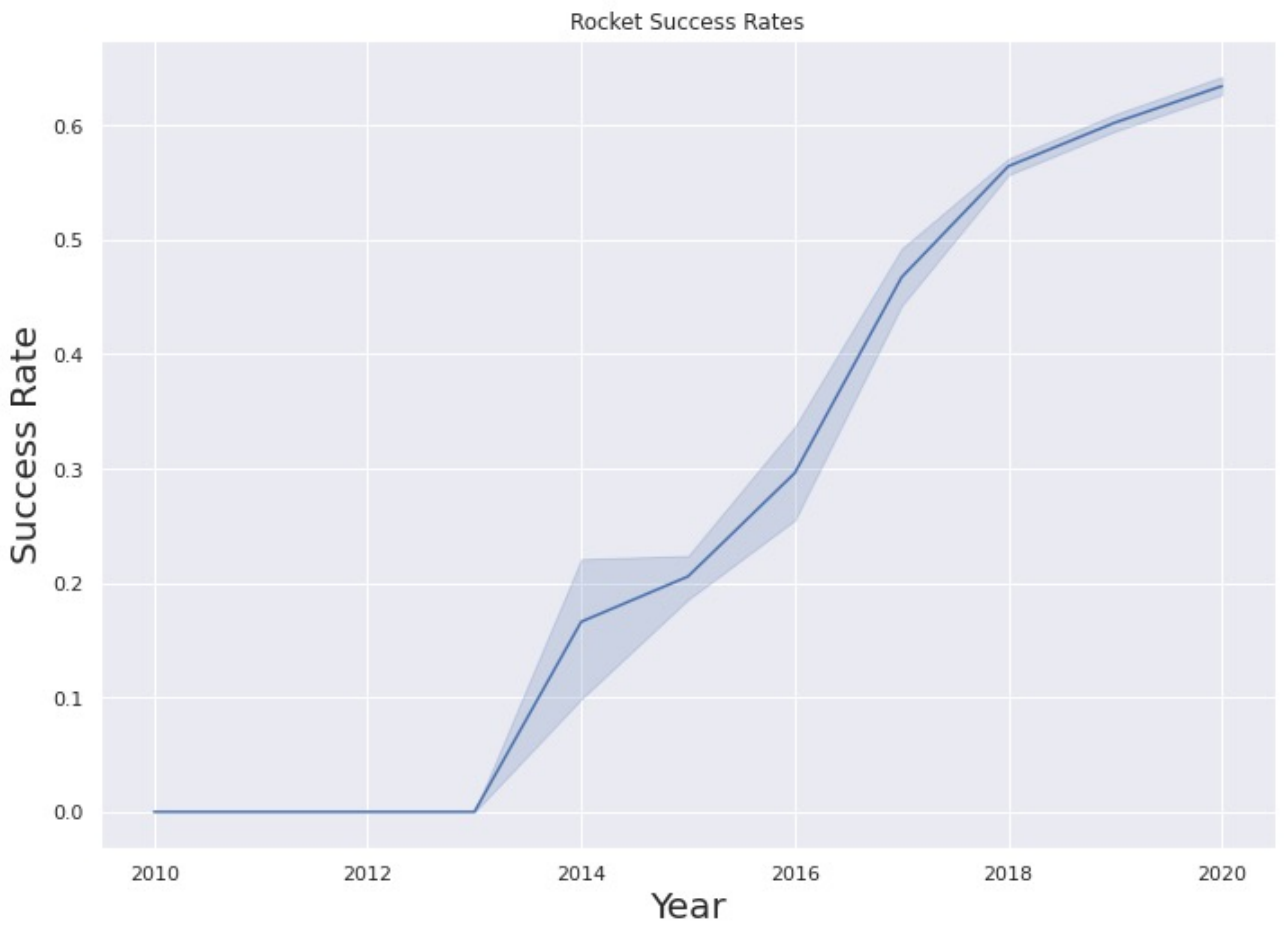
Alternatively, there seems to be no clear relationship between flight number when in GTO orbit

Payload vs. Orbit Type



Observation indicates Heavy payloads have a negative effect on GTO orbits and positive effect on GTO and Polar LEO (ISS) orbits.

Launch Success Yearly Trend



Observations indicate that the success rate from 2013 kept increasing till 2020

All Launch Site Names

```
SELECT DISTINCT Launch_Site from SPACEXDATASET
```

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

QUERY EXPLAINATION

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

Launch Site Names Begin with 'CCA'

```
%sql SELECT * from SPACEXDATASET WHERE launch_site LIKE 'CCA%'LIMIT 5
```

```
 * ibm_db_sa://fdc12884:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud
:32536/bludb
Done.
```

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

QUERY EXPLAINATION

Using the word ***LIMIT*** in the query means that it will only show 5 records from ***SPACEXDATASET*** 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.

Total Payload Mass

```
tpm = %sql SELECT SUM(PAYLOAD_MASS__KG_) TotalPayloadMass from SPACEXDATASET WHERE Customer  = 'NASA (CRS)'
tpm= pd.DataFrame(tpm)
tpm.columns=['Total Payload Mass']
tpm
```

```
 * ibm_db_sa://fdc12884:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud
:32536/bludb
Done.
```

	Total Payload Mass
0	45596

QUERY EXPLAINATION

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

```
apm = %sql SELECT AVG(PAYLOAD_MASS__KG_) AveragePayloadMass from SPACEXDATASET WHERE Booster_Version = 'F9 v1.1'
apm=pd.DataFrame(apm)
apm.columns=['average payload mass']
apm
```

```
 * ibm_db_sa://fdc12884:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud
:32536/bludb
Done.
```

	average payload mass
0	2928

QUERY EXPLAINATION

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

```
date_of_successful_landing = %sql SELECT MIN(Date)  from SPACEXDATASET WHERE landing__outcome='Success (drone sh
date_of_successful_landing=pd.DataFrame(date_of_successful_landing)
date_of_successful_landing.columns=['Date which first Successful landing outcome in drone ship was acheived.']
date_of_successful_landing
```

```
 * ibm_db_sa://fdc12884:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud
:32536/bludb
Done.
```

	Date which first Successful landing outcome in drone ship was acheived.
0	2016-04-08

QUERY EXPLAINATION

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

The ***WHERE*** clause filters the dataset to only perform calculations on ***Landing__Outcome Success (drone ship)***

Successful Drone Ship Landing with Payload between 4000 and 6000

```
boosters_of_successful_landing= %sql SELECT booster_version  from SPACEXDATASET WHERE landing__outcome='Success (
boosters_of_successful_landing=pd.DataFrame(boosters_of_successful_landing)
boosters_of_successful_landing.columns=['Date which first Successful landing outcome in drone ship was acheived.'
boosters_of_successful_landing
```

```
 * ibm_db_sa://fdc12884:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud
:32536/bludb
Done.
```

	Date which first Successful landing outcome in drone ship was acheived.
0	F9 FT B1032.1
1	F9 B4 B1040.1
2	F9 B4 B1043.1

%sql SELECT booster_version from SPACEXDATASET WHERE landing__outcome='Success (ground pad)' AND payload_mass__kg_ > 4000 AND payload_mass__kg_ < 6000

QUERY EXPLAINATION

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

	Successful_Mission_Outcomes	Failure_Mission_Outcomes
0	100	1

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

QUERY EXPLAINATION

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.

Boosters Carried Maximum Payload

```
oster_mpl=%sql SELECT DISTINCT booster_version,payload_mass__kg_ from SPACEXDATASET  ORDER BY payload_mass__kg_ DESC
oster_mpl=pd.DataFrame(booster_mpl)
oster_mpl.rename(columns={0: "booster_version", 1: "Maximum Payload Mas"})
```

```
 * ibm_db_sa://fdc12884:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:325
36/bludb
Done.
```

	booster_version	Maximum Payload Mas
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600

QUERY EXPLAINATION

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

ORDER BY puts the list in order set to a certain condition. ***DESC*** means its arranging the dataset into descending order

2015 Launch Records

```
SELECT DATENAME(month, DATEADD(month, MONTH(CONVERT(date, Date, 105)), 0) - 1) AS Month,
Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXDATASET
WHERE (Landing_Outcome LIKE N'%Success%') AND (YEAR(CONVERT(date, Date, 105)) = ‘2015')
```

QUERY EXPLAINATION

a much more complex query as I had my ***Date*** fields in SQL Server stored as ***NVARCHAR*** the ***MONTH*** function returns name month. The function ***CONVERT*** converts ***NVARCHAR*** to ***Date***. ***WHERE*** clause filters ***Year*** to be 2015

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

SELECT Count(landing__outcome) AS sl FROM SPACEXDATASET WHERE (landing__outcome LIKE '%Success%') AND (DATE >'2010-06-04') AND (DATE < '2017-03-20’)

	Successful Landing Outcomes Between 2010-06-04 and 2017-03-20
0	8

QUERY EXPLAINATION

Function ***COUNT*** counts records in column ***WHERE*** filters data

LIKE (wildcard)

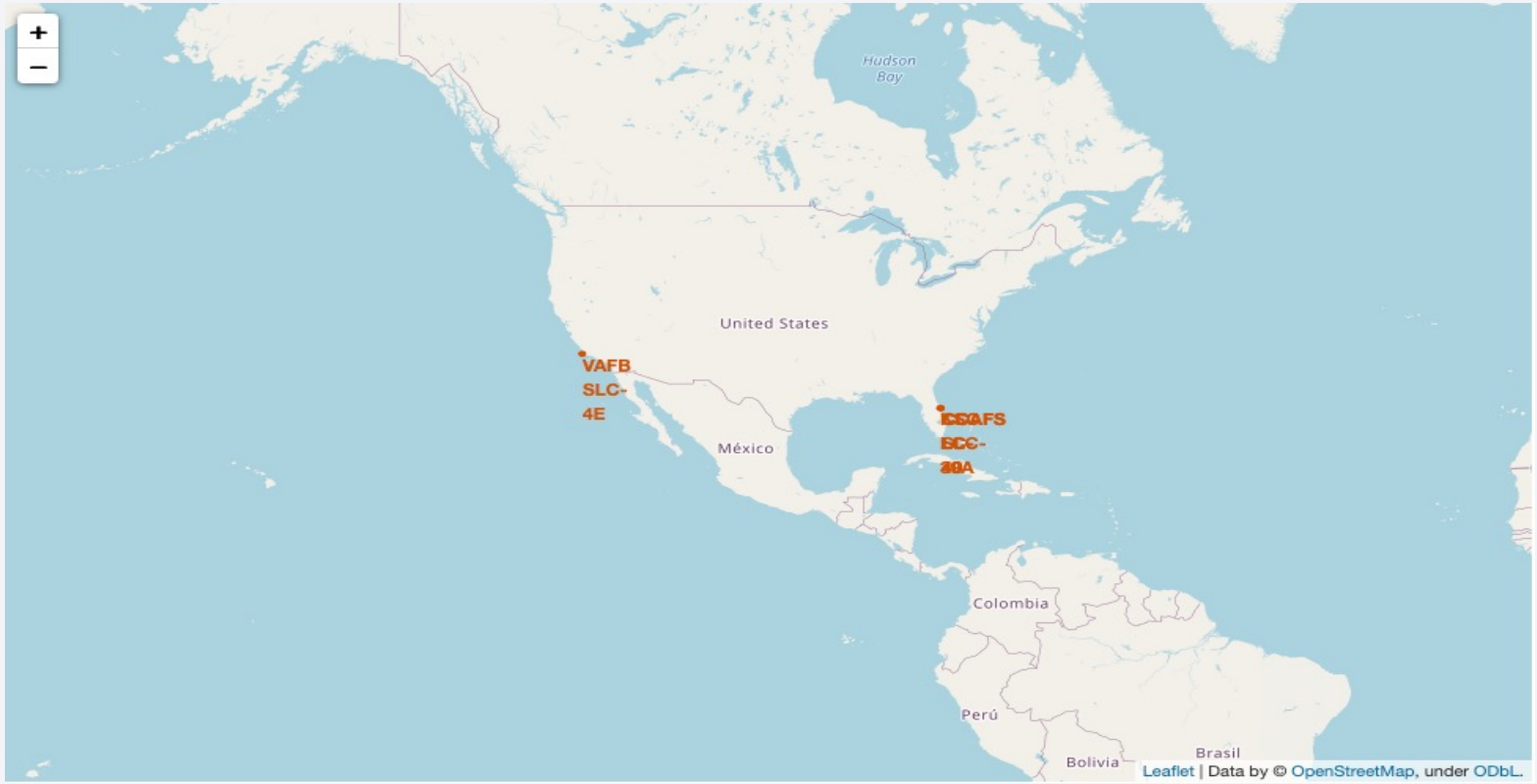
AND (conditions)

AND (condition

Section 4

Launch Sites Proximities Analysis

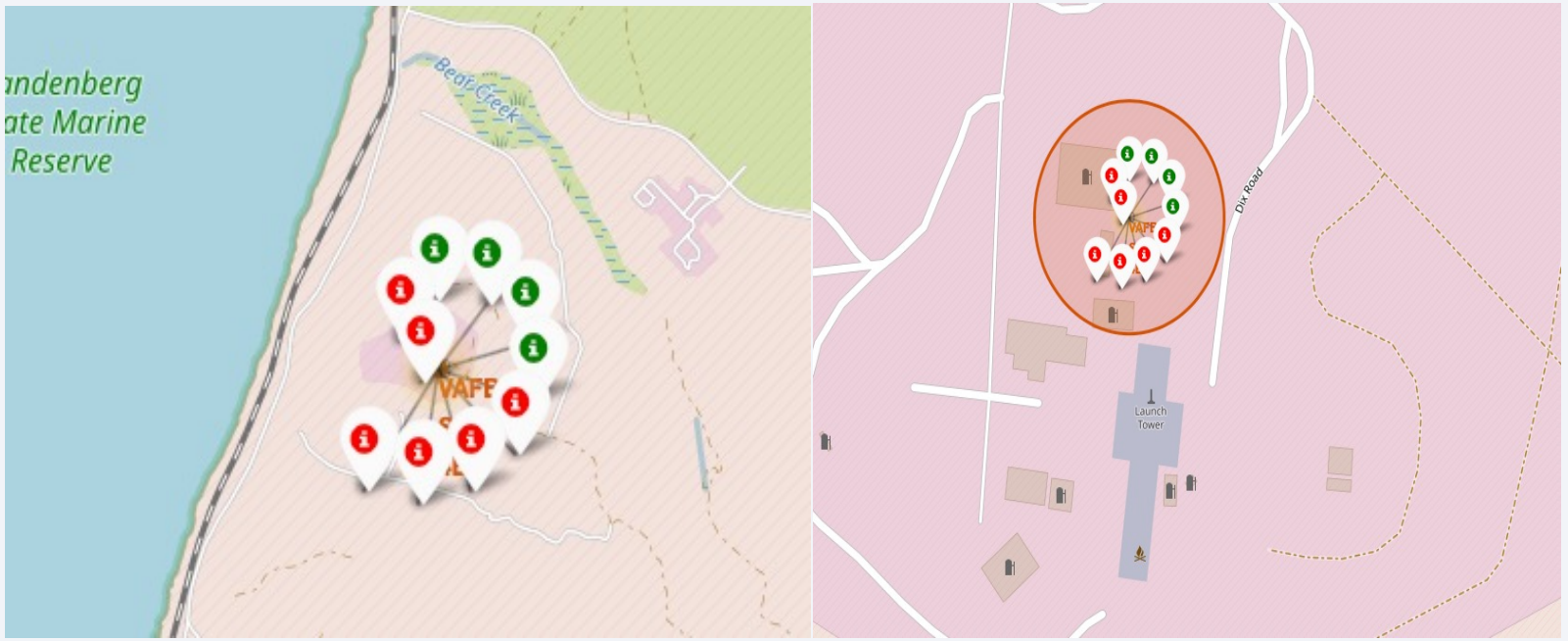
All launch sites global map markers



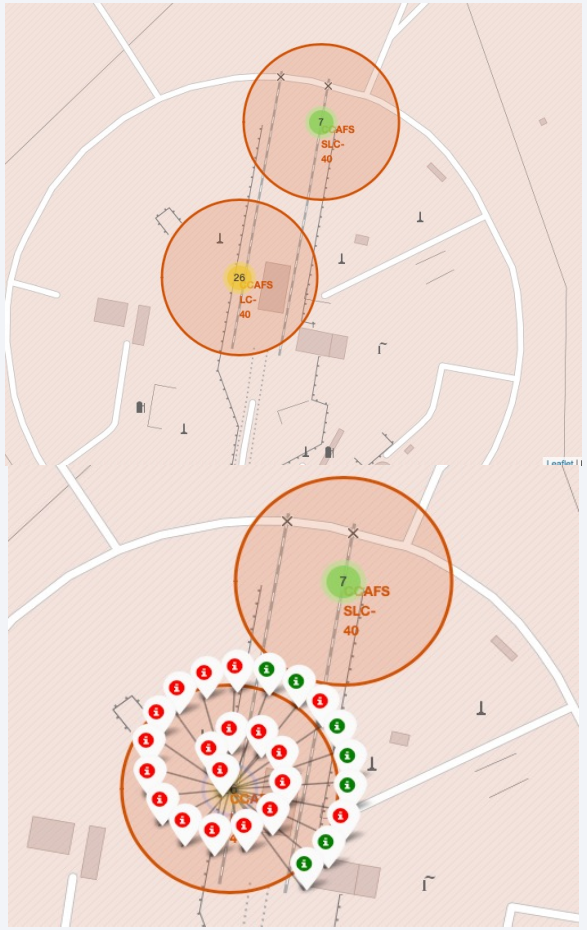
We can observe that the launch sites are in the United States of America. California,Florida

Color Labelled Markers

California Launch Sites



Florida Launch Sites

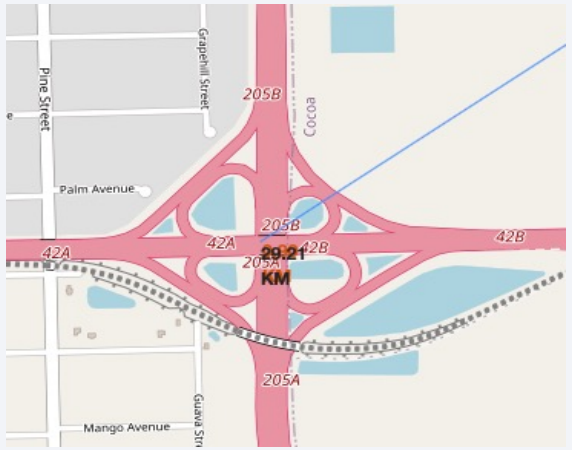


Green Marker shows successful Launches and Red Marker shows Failures

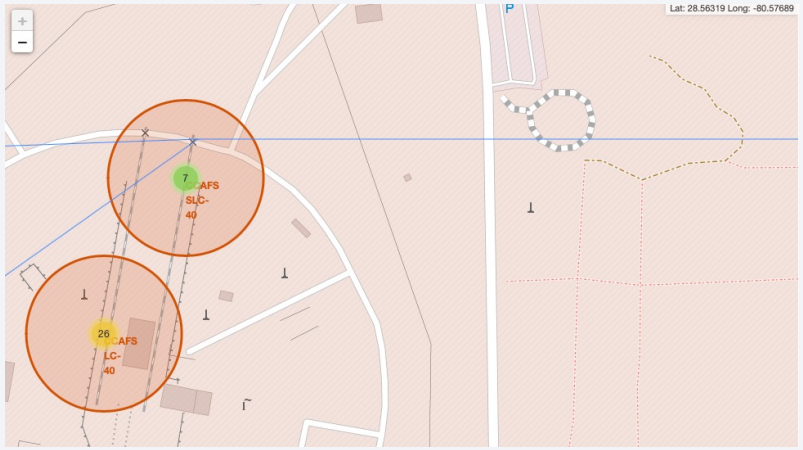
Determining Launch Site Distance using Haversine formula

Distance to City

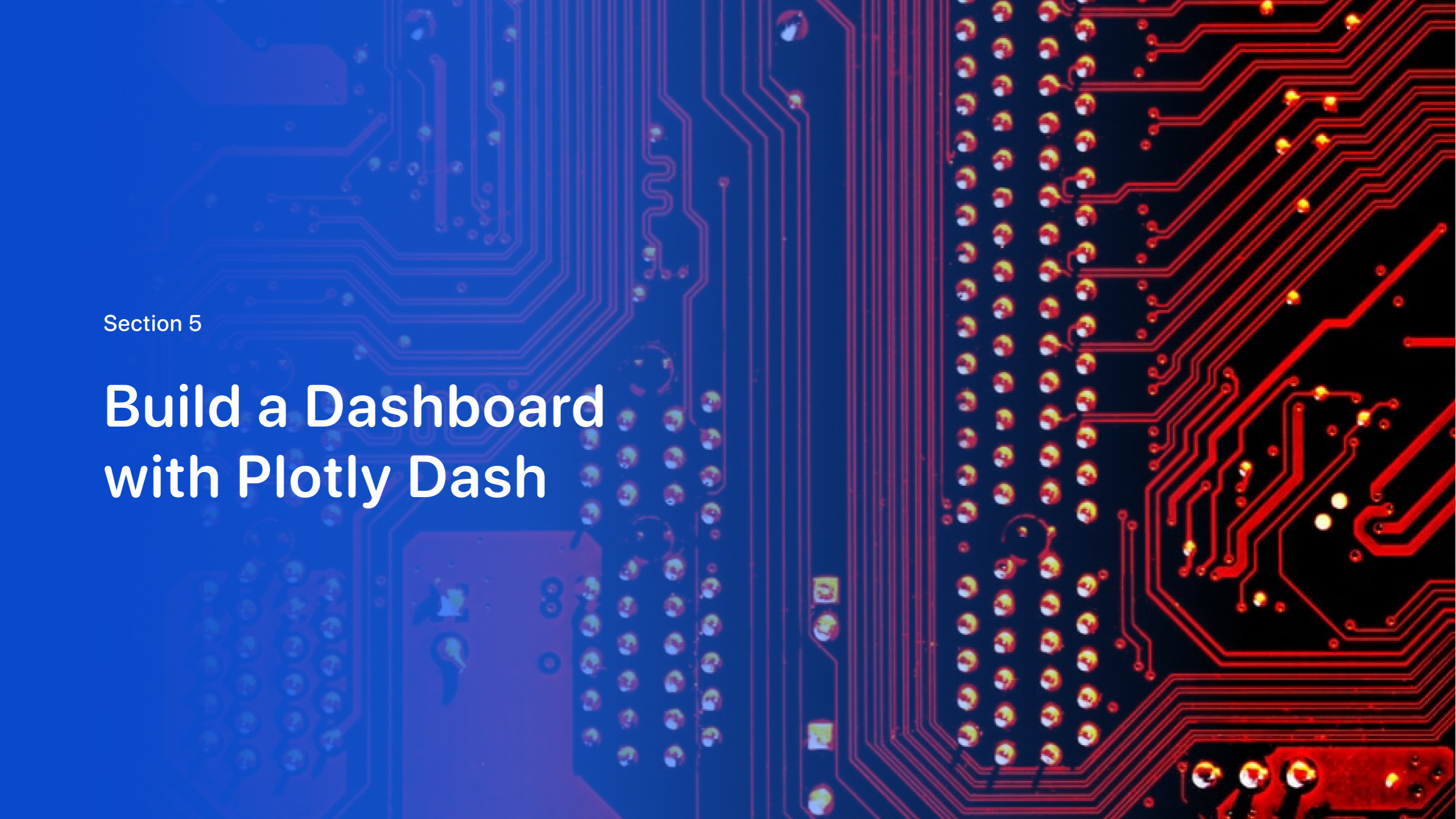
Distance to the closest Highway



Distance to the coast



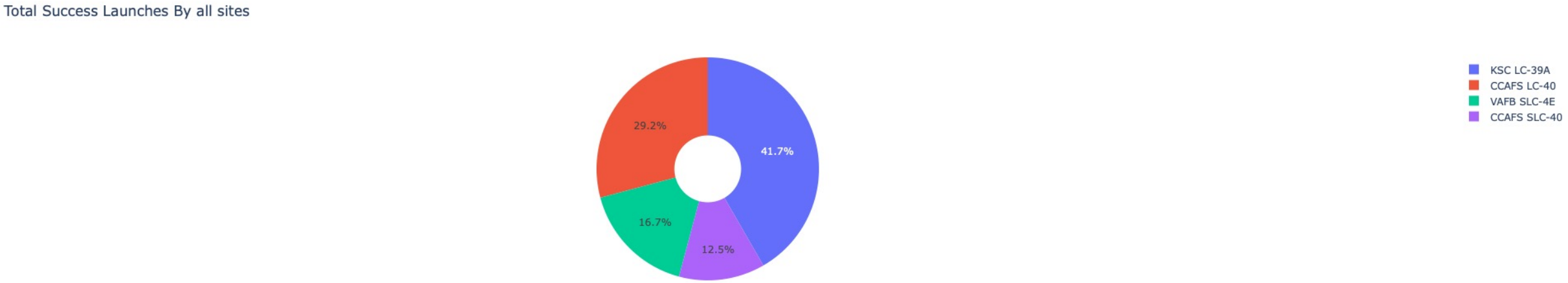
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes



Section 5

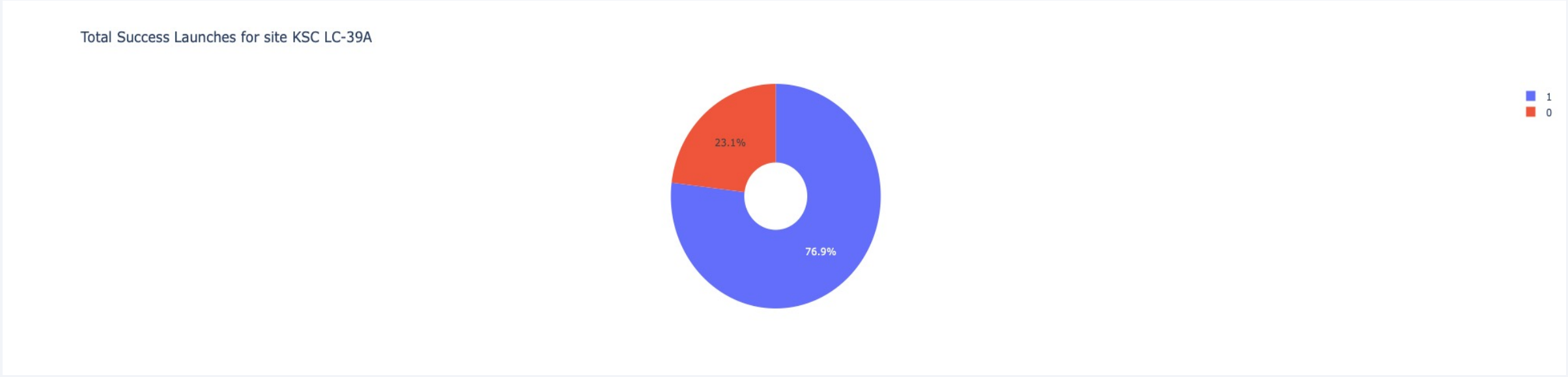
Build a Dashboard with Plotly Dash

Dashboard with Plotly Dash



It can be observed that KSC LC-39A has the most successful launches from all the sites

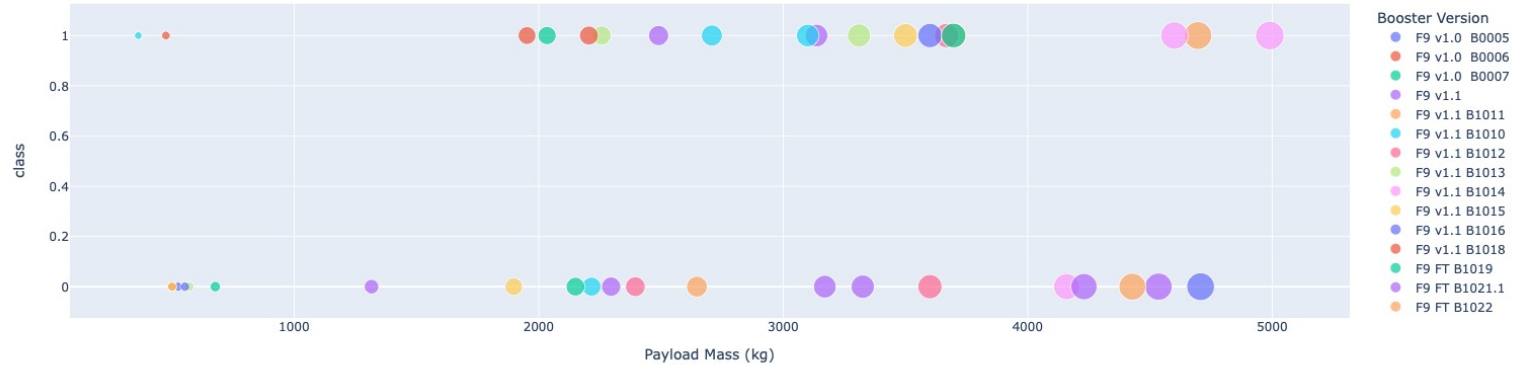
Launch site with highest launch success ratio



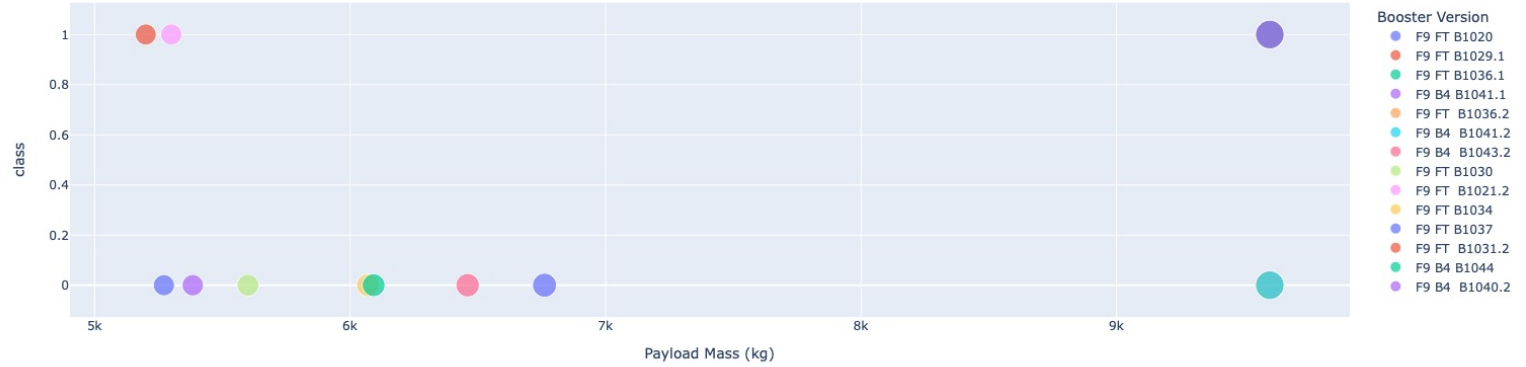
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

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

Low Weighted Payload 0kg – 5000kg



Heavy Weighted Payload 5000kg – 10000kg



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

Section 6

Predictive Analysis (Classification)

Classification Accuracy


```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 16, 'max_features': 'sqrt', 'min_sampl
es_leaf': 2, 'min_samples_split': 5, 'splitter': 'best'}
accuracy : 0.8892857142857145
```

Best Algorithm is D_Tree with a score of 0.8892857142857145

Confusion Matrix

Examining the confusion matrix, we see that Tree can distinguish between the different classes. We see that the major problem is false positives.

Conclusions

- 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
- The Tree Classifier Algorithm is the best for Machine Learning for this dataset.

Thank you!