Winning Space Race with Data Science

Elina Zmeykina, Ph.D.
23 May, 2023

Outline

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

Executive Summary

Methodologies:

- Data collection with API and Webscraping
- Data wrangling
- Exploratory data analysis (EDA) with SQL and visualisation
- Interactive visual analytics and Dashboards
- Predictive analytics (Classification)

Summary of all results:

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

Introduction

- SpaceX own the breakthrough reusable rocket technology. The Falcon 9 rocket, with its advertised launch cost of $62 million, offers substantial savings compared to competing services costing upwards of $165 million per launch. The key factor driving these cost savings is SpaceX's ability to successfully land and reuse the Falcon 9's first stage.
- However, the question remains: **What conditions significantly influence the successful landing of the Falcon 9's first stage?**
- Understanding this could serve as a valuable insight for alternative companies aiming to compete with SpaceX. By identifying and examining the various factors that contribute to a successful landing, competitors could optimize their strategies to potentially achieve similar cost-efficiencies.

Research questions

1. What specific rocket variables and conditions directly influence the likelihood of successful Falcon 9 first stage landing?

2. How does each of these variables correlate with the success rate of the rocket landing?

3. What conditions and optimizations should SpaceX, or any alternative company, strive for to maximize the likelihood of successful rocket landing and, consequently, realize the greatest cost savings?

Section 1

Methodology

Methodology

- Data collection methodology:
  - SpaceX Rest API
  - Web scraping from Wikipedia
- Perform data wrangling
  - One-hot encoding data for ML
  - Dropping irrelevant columns
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Scatter plots, Bar plots to show relationships between variables and 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

Data Collection

The subsequent datasets were compiled through the following methods:

- We sourced data on SpaceX launches from the SpaceX REST API.
- This resource provides comprehensive information about each launch, including the specific rocket deployed, the payload delivered, detailed launch and landing specifications, and the outcome of the landing.
- Our primary objective with this data is to devise a model that can accurately predict if SpaceX will attempt to land a given rocket.
- The SpaceX REST API can be accessed via the URL that begins with api.spacexdata.com/v4/
- In addition to the SpaceX REST API, we also utilized web scraping techniques on Wikipedia to gather additional Falcon 9 Launch data. The tool of choice for this task was BeautifulSoup, a popular library for web scraping in Python.

Data Collection – SpaceX API

GitHub URL to Notebook

1. Request and parse the SpaceX launch data using the GET request

2. Apply custom functions:
   - `payload` - learn the mass of the payload and the orbit that it is going to
   - `launchpad` - know the name of the launch site being used, the longitude, and the latitude.
   - `cores` - learn the outcome of the landing, the type of the landing, number of flights with that core, etc.

3. Filter the dataframe to only include `Falcon 9` launches

4. Data wrangling - Dealing with Missing Values

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

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

```
# Call getLaunchSite
getLaunchSite(data)
getPayloadData(data)
getCoreData(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}
```

```
data_falcon9 = data[data['BoosterVersion']=='Falcon 9']
data_falcon9.head()
```

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

```
Out[83]: FlightNumber      0
         Date              0
         BoosterVersion    0
         PayloadMass       0
         Orbit             0
         LaunchSite        0
         Outcome           0
         Flights           0
         GridFins          0
         Reused            0
         Legs              0
         LandingPad       26
         Block             0
         ReusedCount       0
         Serial            0
         Longitude         0
         Latitude          0
         dtype: int64
```

Data Collection - Scraping

[GitHub URL to Notebook]

- Request the Falcon9 Launch Wiki page from its URL
- Create BeautifulSoup Object
- Get column names
- Create dictionary
- Append data to keys
- Convert dictionary to dataframe

```
# use requests.get() method with the provided static_url
# assign the response to a object
r = requests.get(static_url)
r.content
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup=BeautifulSoup(r.content)
soup.prettify()
```

```
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]

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

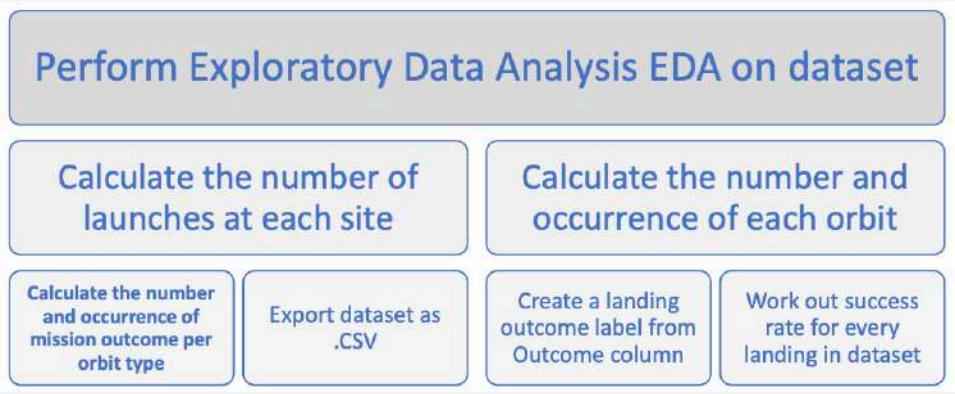
```
for columns, values in launch_dict.items():
    print(f'lenght of {columns}: {len(values)}')
#df=pd.DataFrame(launch_dict)
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

```
lenght of Flight No.: 1500
lenght of Launch site: 1500
lenght of Payload: 1500
lenght of Payload mass: 1500
lenght of Orbit: 1500
lenght of Customer: 1267
lenght of Launch outcome: 1497
lenght of Version Booster: 1500
lenght of Booster landing: 1493
lenght of Date: 1500
lenght of Time: 1500
```

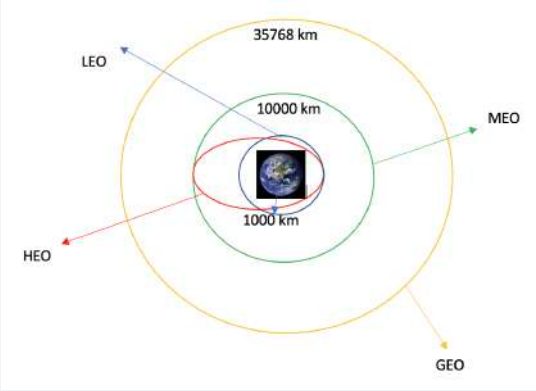
Data Wrangling

GitHub URL to Notebook

- Our dataset includes various instances where the booster landing was unsuccessful.
- "True Ocean" implies successful ocean landing; "False Ocean" denotes failure.
- "True RTLS" indicates successful landing to a ground pad, "False RTLS" signifies failure.
- "True ASDS" means successful landing on a drone ship; "False ASDS" represents failure.
- We've converted these outcomes into Training Labels: '1' for a successful landing and '0' for an unsuccessful one.



Each launch aims to an dedicated orbit, and here are some common orbit types:



Common orbit types for SpaceX

EDA with Data Visualization

GitHub URL to Notebook

- Scatter Graphs being drawn:
  - Flight Number VS. Payload Mass
  - Flight Number VS. Launch Site
  - Payload VS. Launch Site
  - Orbit VS. Flight Number
  - Payload VS. Orbit Type
  - Orbit VS. Payload Mass
- Bar Graph being drawn:
  - Mean VS. Orbit
- Line Graph being drawn:
  - Success Rate VS. Year

- 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.
- 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 graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded

EDA with SQL

GitHub URL to Notebook

- 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 booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was
- 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 names of the booster_versions which have carried the maximum payload mass.
- Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Ranking the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

Build an Interactive Map with Folium

GitHub URL to Notebook

- 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 various trends about what is around the Launch Site to measure patterns. Lines are drawn on the map to measure distance to landmarks
- Example of some trends in which the Launch Site is situated in:
    - 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**

Build a Dashboard with Plotly Dash

GitHub URL to Notebook

- The dashboard is built with TheiaBlueprint and Dash web framework.
- Pie Chart:
  - Pie Chart shows 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 is the best method to show you a non-linear pattern.
  - The range of data flow, i.e. maximum and minimum value, can be determined.
  - Observation and reading are straightforward.

Predictive Analysis (Classification)

GitHub URL to Notebook

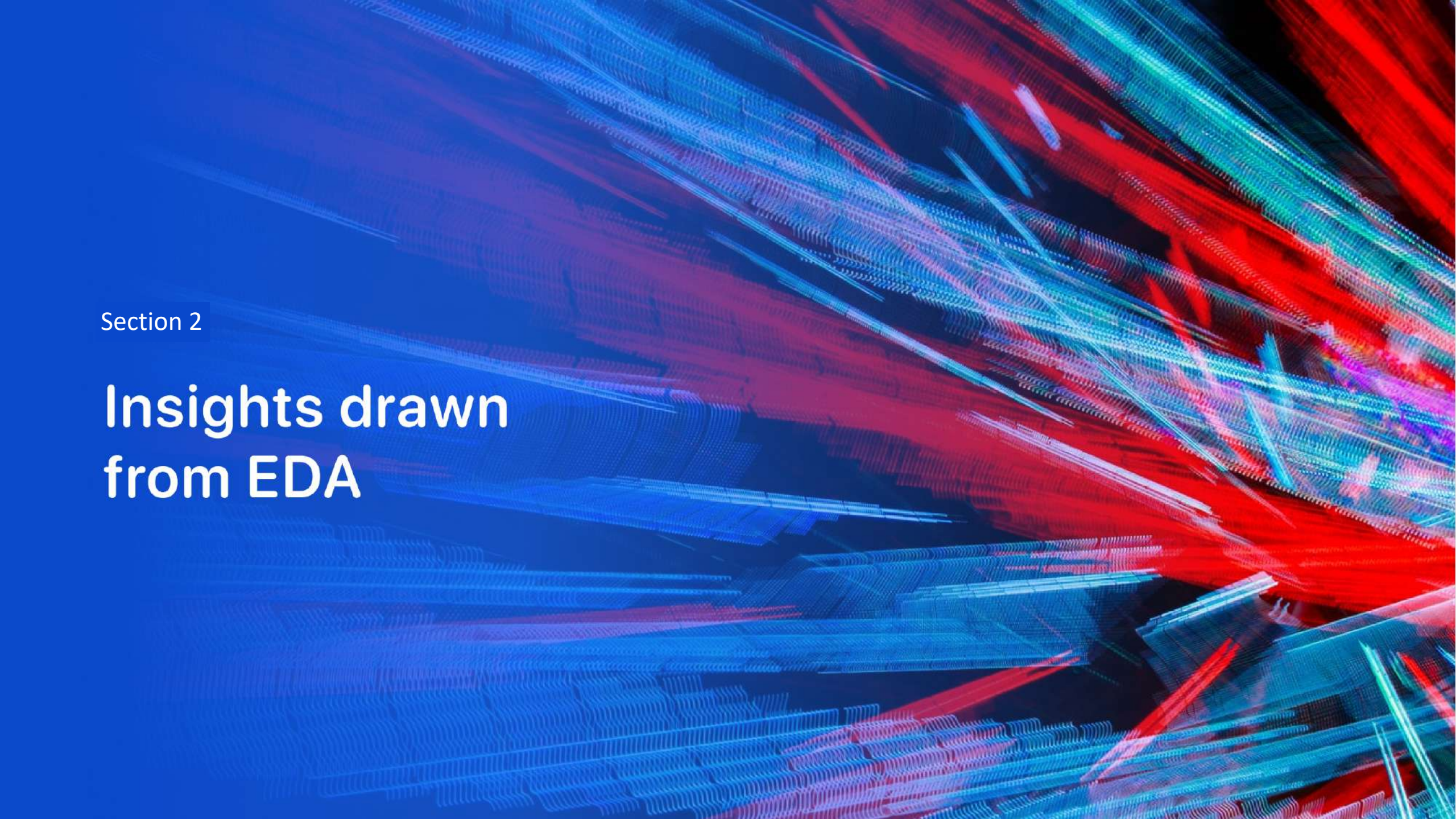
- Building model
  - Load our dataset into NumPy and Pandas
  - Transform Data
  - Split our data into training and test data sets
  - Check how many test samples we have
  - Decide which type of machine learning algorithms we want to use
  - Set our parameters and algorithms to GridSearchCV
  - Fit 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
- 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

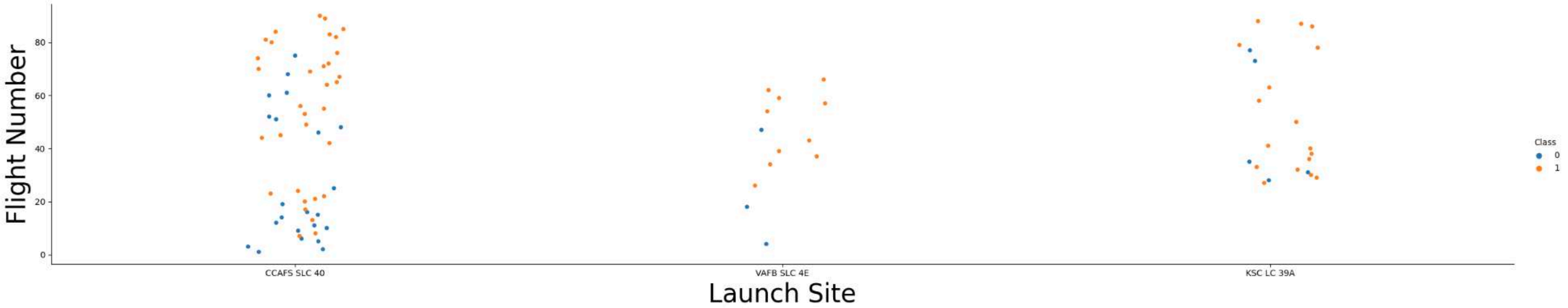
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

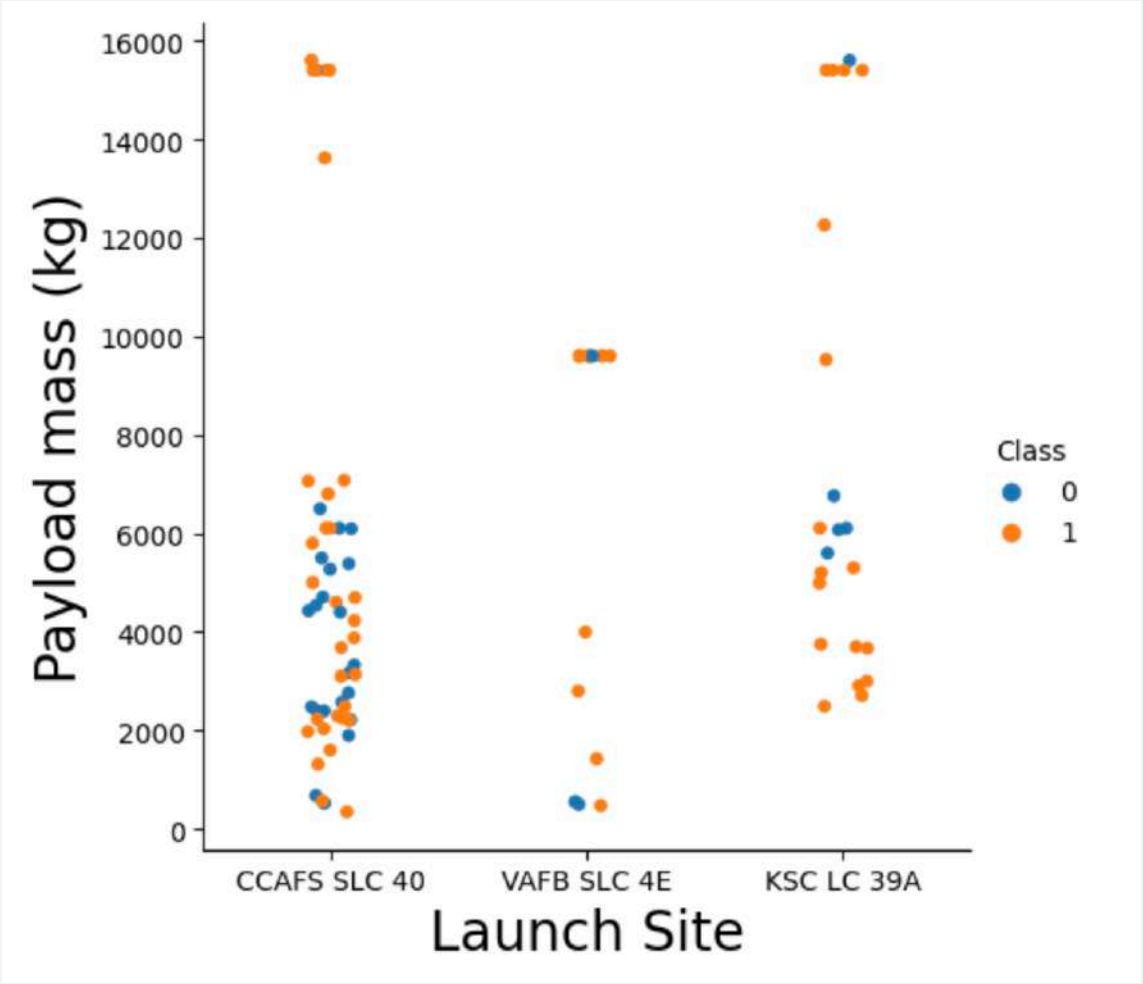


Flight Number vs. Launch Site



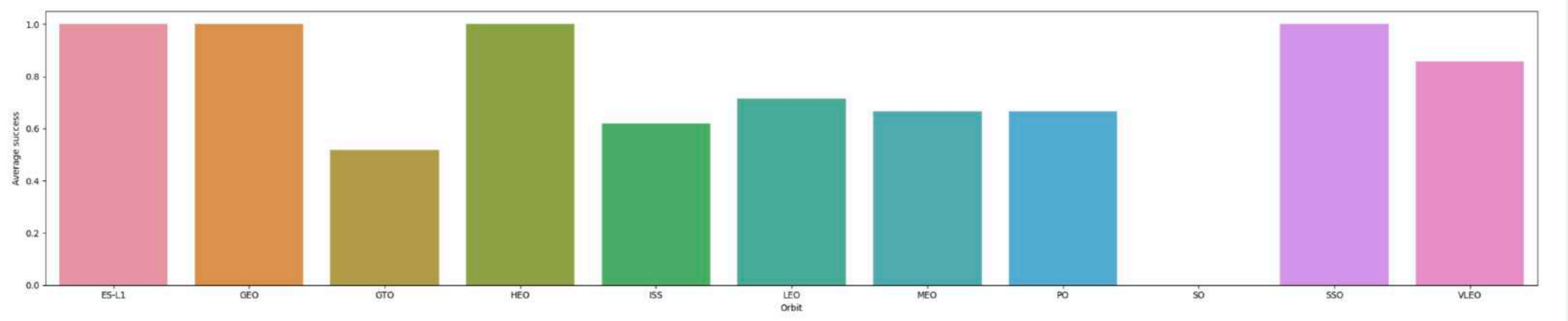
The more amount of flights at a launch site the greater the success rate at a launch site

Payload vs. Launch Site



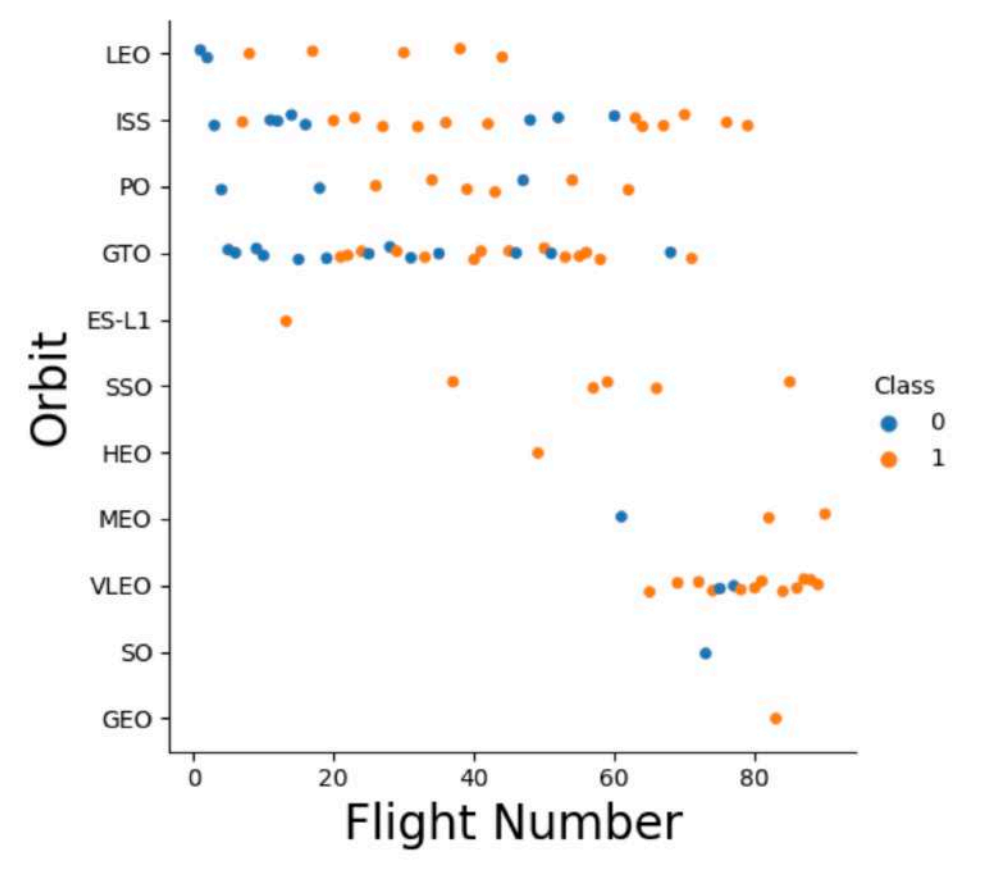
- The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependant on Pay Load Mass for a success launch.

Success Rate vs. Orbit Type



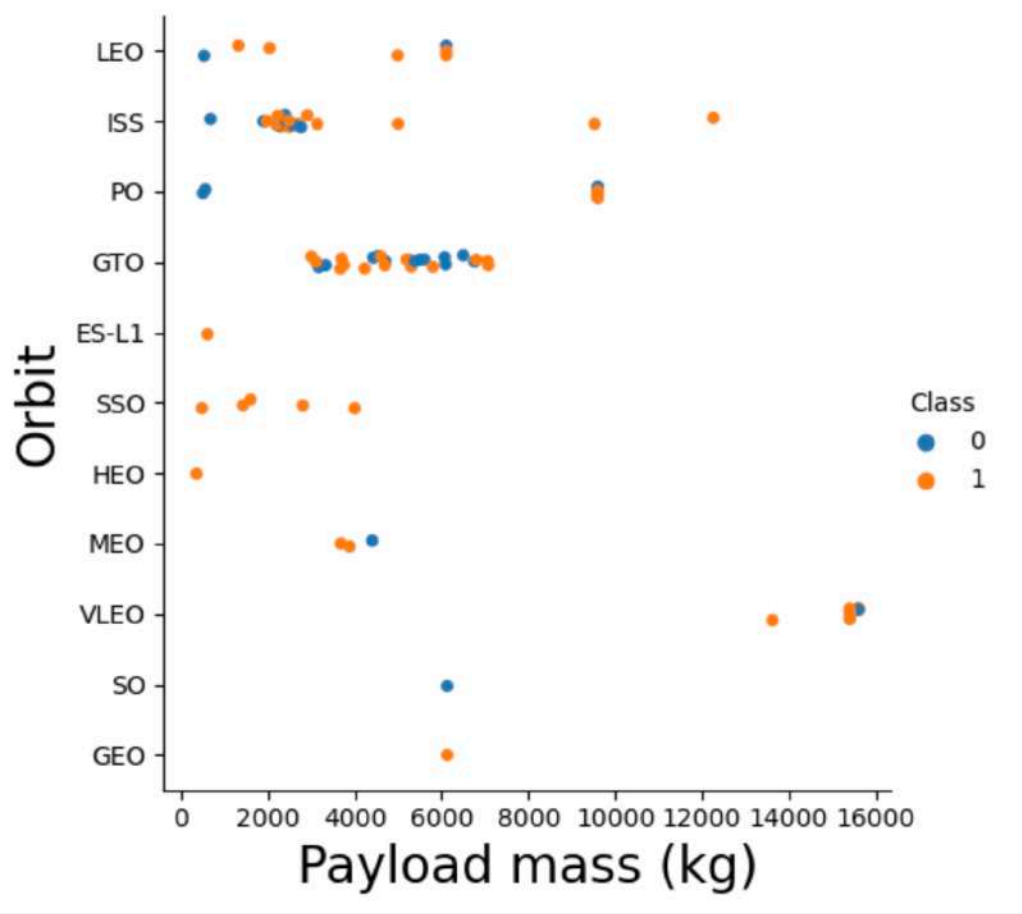
- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

Flight Number vs. Orbit Type



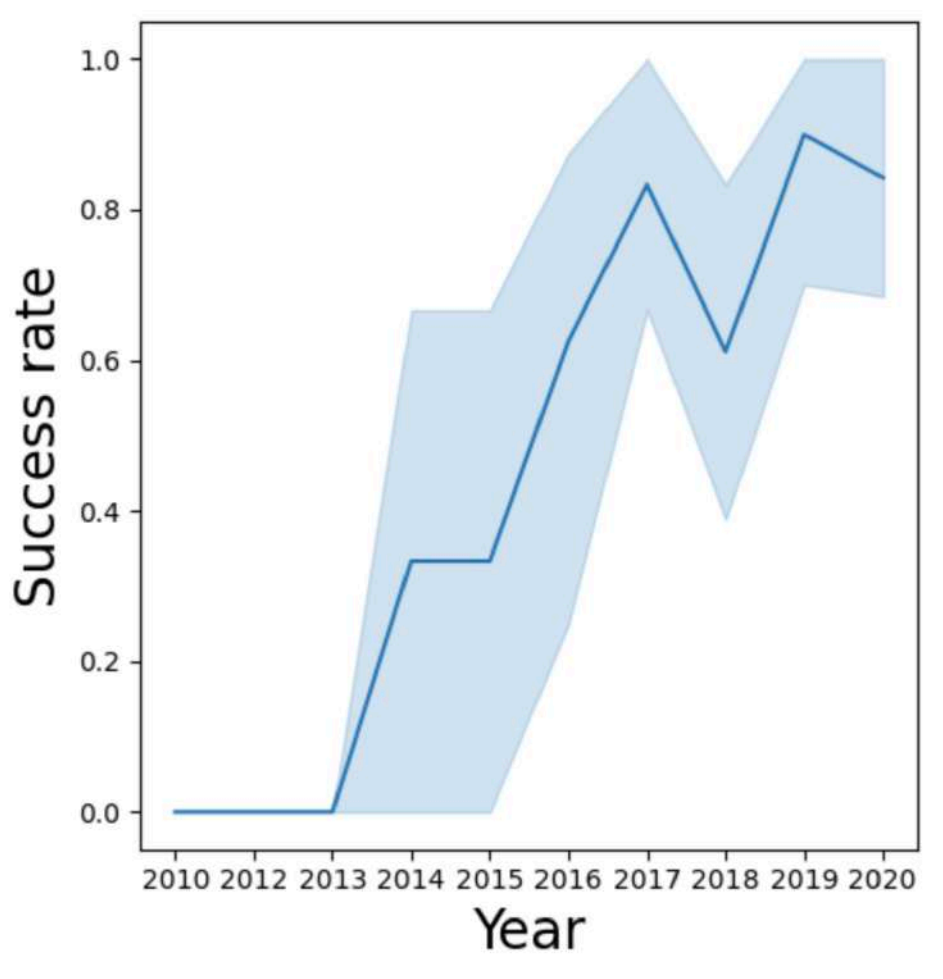
- LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type



- Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits

Launch Success Yearly Trend



- The success rate since 2013 kept increasing till 2020

All Launch Site Names

```
select DISTINCT Launch_Site from SPACEXTBL
```

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

- 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'

```
select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Lan
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Fai
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Fai
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	

Total Payload Mass

```
select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer like '%CRS%'
```

sum(PAYLOAD_MASS__KG_)
48213.0

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

```
select avg(PAYLOAD_MASS__KG_) from SPACEXTBL
where Booster_Version like 'F9 v1.1%'
```

avg(PAYLOAD_MASS__KG_)
2534.6666666666665

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

```
select min(substr(Date, 7,4) || '-' || substr(Date, 4,2) || '-' || substr(Date, 1,2)) as date
from SPACEXTBL where Landing_Outcome like 'Success%'
```

date
2015-12-22

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

```
select Booster_Version, PAYLOAD_MASS__KG_, Landing_Outcome
from SPACEXTBL
where PAYLOAD_MASS__KG_ between 4000 and 6000 and
Landing_Outcome like 'Success%'
```

Booster_Version	PAYLOAD_MASS__KG_	Landing_Outcome
F9 FT B1022	4696.0	Success (drone ship)
F9 FT B1026	4600.0	Success (drone ship)
F9 FT B1021.2	5300.0	Success (drone ship)
F9 FT B1032.1	5300.0	Success (ground pad)
F9 B4 B1040.1	4990.0	Success (ground pad)
F9 FT B1031.2	5200.0	Success (drone ship)
F9 B4 B1043.1	5000.0	Success (ground pad)
F9 B5 B1046.2	5800.0	Success
F9 B5 B1047.2	5300.0	Success
F9 B5 B1046.3	4000.0	Success
F9 B5 B1048.3	4850.0	Success
F9 B5 B1051.2	4200.0	Success
F9 B5B1060.1	4311.0	Success
F9 B5 B1058.2	5500.0	Success
F9 B5B1062.1	4311.0	Success

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

```
select count(*) as total_success from SPACEXTBL
where Mission_Outcome like "Success%"
```

```
select count(*) as total_failure from SPACEXTBL
where Mission_Outcome like 'Fail%'
```

total_success
100

total_failure
1

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

```
select Booster_Version, PAYLOAD_MASS__KG_ from SPACEXTBL
where PAYLOAD_MASS__KG_= (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600.0
F9 B5 B1049.4	15600.0
F9 B5 B1051.3	15600.0
F9 B5 B1056.4	15600.0
F9 B5 B1048.5	15600.0
F9 B5 B1051.4	15600.0
F9 B5 B1049.5	15600.0
F9 B5 B1060.2	15600.0
F9 B5 B1058.3	15600.0
F9 B5 B1051.6	15600.0
F9 B5 B1060.3	15600.0
F9 B5 B1049.7	15600.0

This is subquery selecting the PAYLOAD_MASS_KG_ only for maximum value and finds the corresponding version of Booster

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

```
select Landing_Outcome, (substr(Date, 7,4) || '-' || substr(Date, 4,2) || '-' ||
substr(Date, 1,2)) as FormDate from SPACEXTBL
where Landing_Outcome like 'Success%' and
(substr(Date, 7,4) || '-' || substr(Date, 4,2) || '-' || substr(Date, 1,2)) between
'2010-06-04' and '2017-03-20'
order by FormDate desc
```

Landing_Outcome	FormDate
Success (ground pad)	2017-03-06
Success (ground pad)	2017-02-19
Success (drone ship)	2017-01-14
Success (ground pad)	2017-01-05
Success (drone ship)	2016-08-14
Success (drone ship)	2016-08-04
Success (ground pad)	2016-07-18
Success (drone ship)	2016-06-05
Success (drone ship)	2016-05-27
Success (ground pad)	2015-12-22

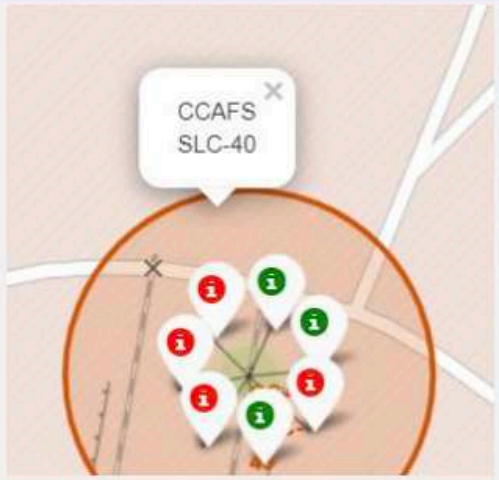
Filters the data by date between '2010-06-04' and ‘2017-03-20'

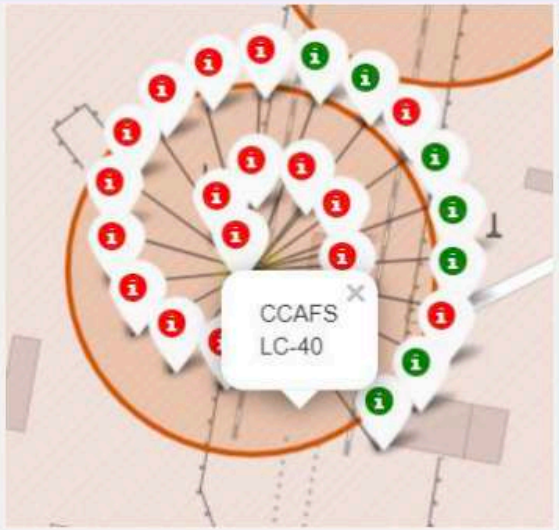

Section 3
Launch Sites
Proximities Analysis

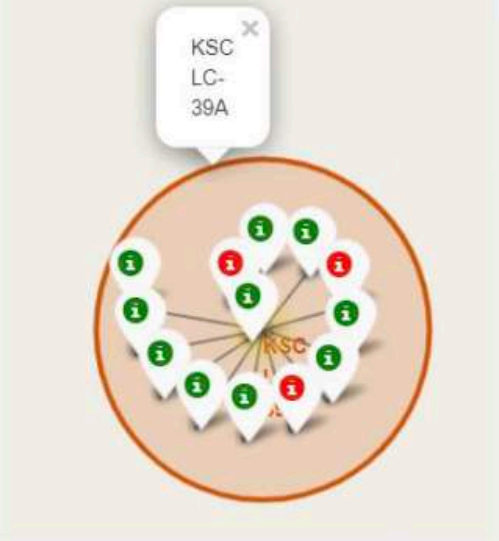
All launch sites global map markers

We can see that the SpaceX launch sites are in the USA coasts - Florida and California

Colour Labelled Markers

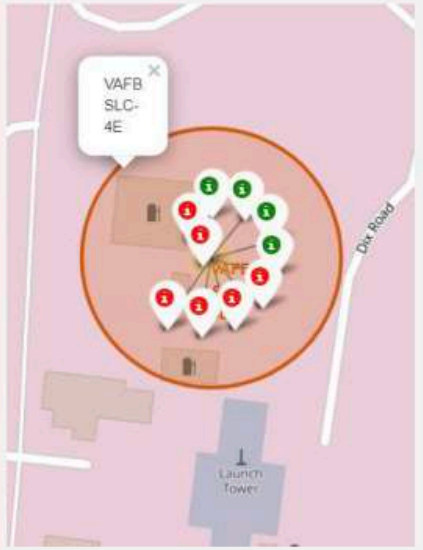






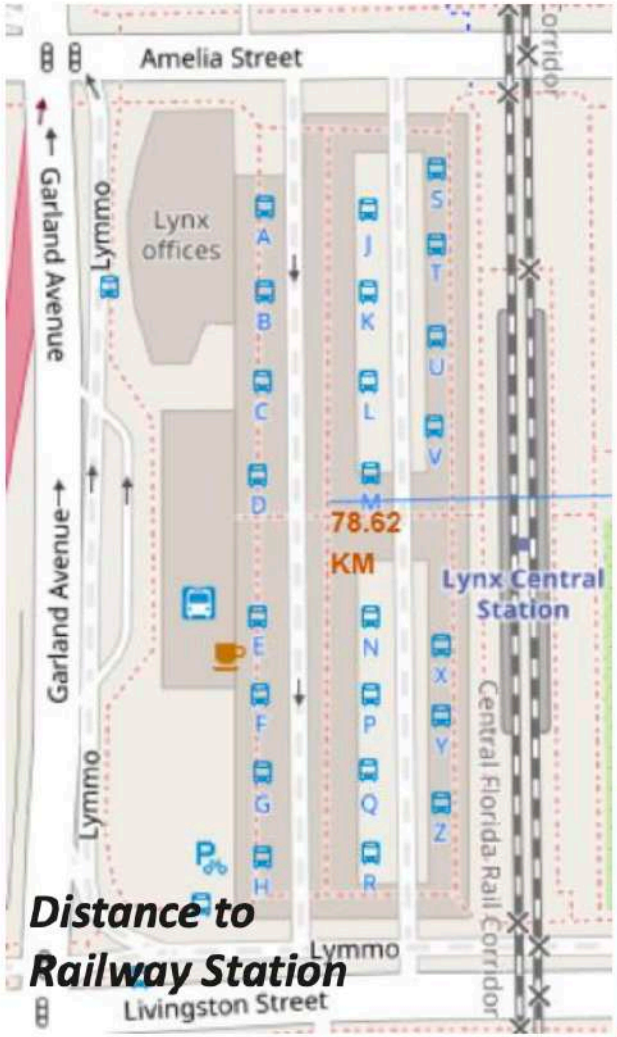
Florida Launch Sites

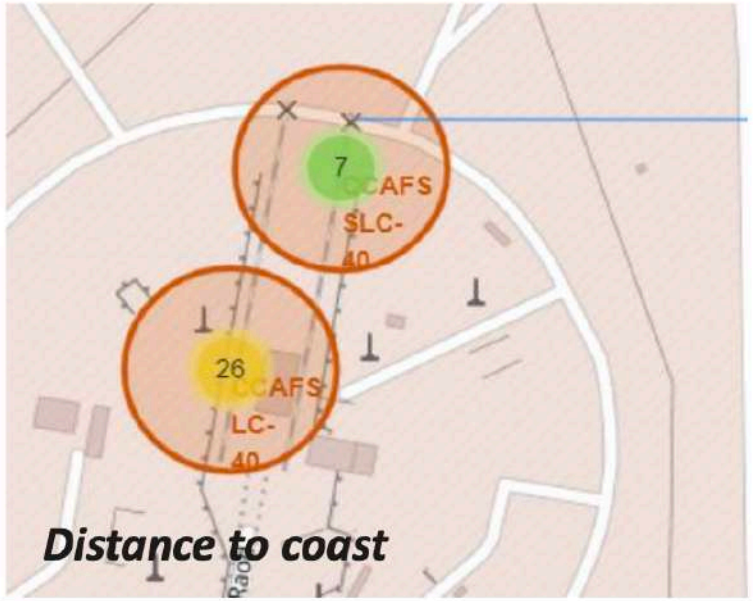
Green Marker shows successful Launches and Red Marker shows Failures

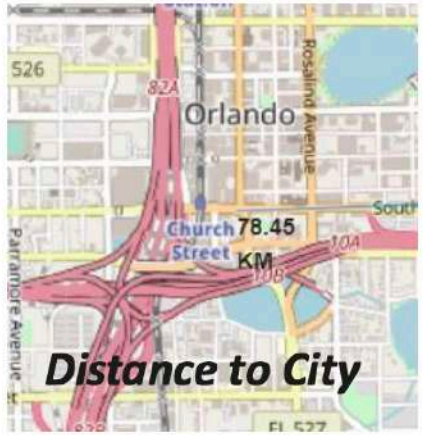


California Launch Site

Launch Sites distance to landmarks



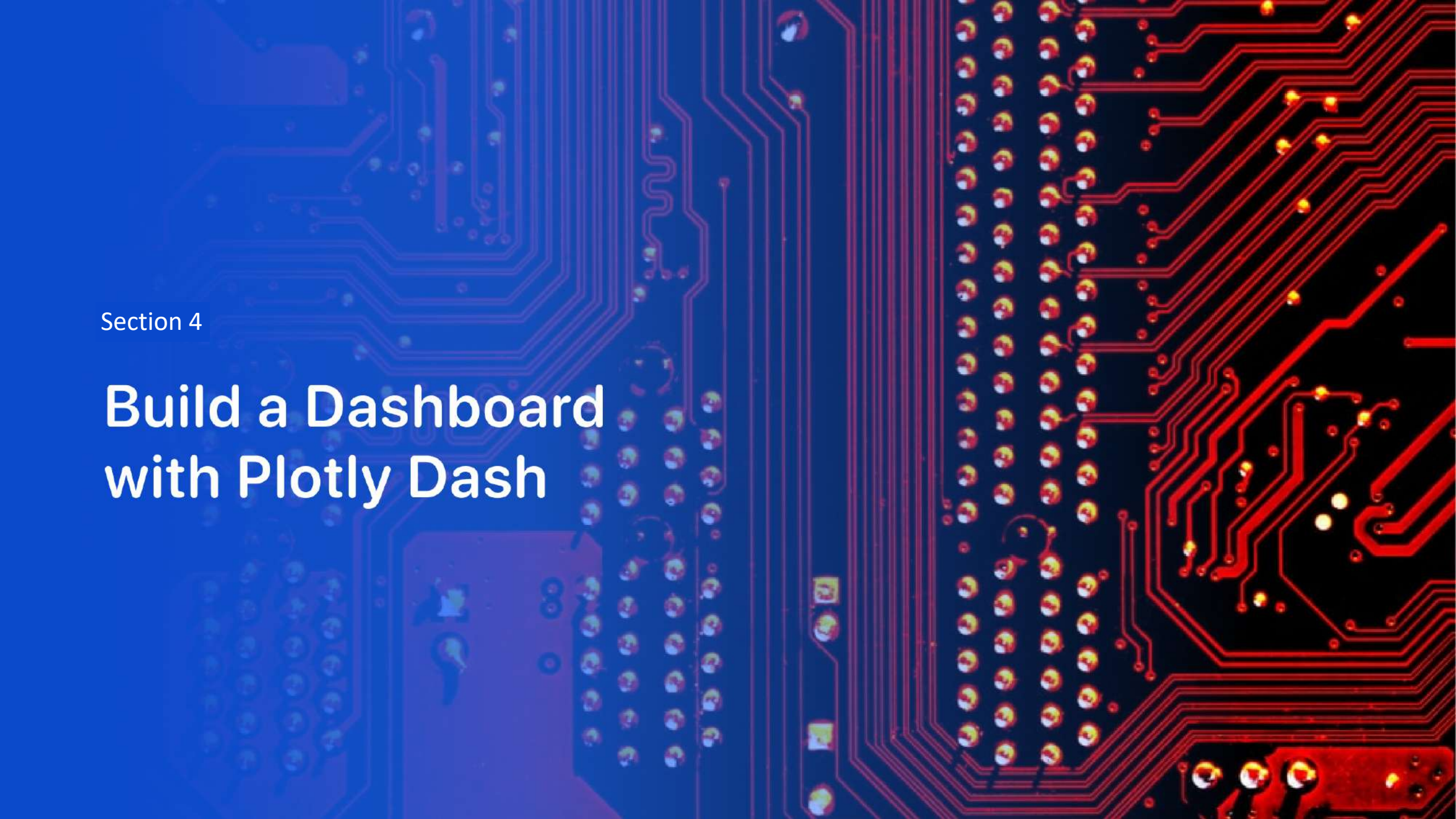




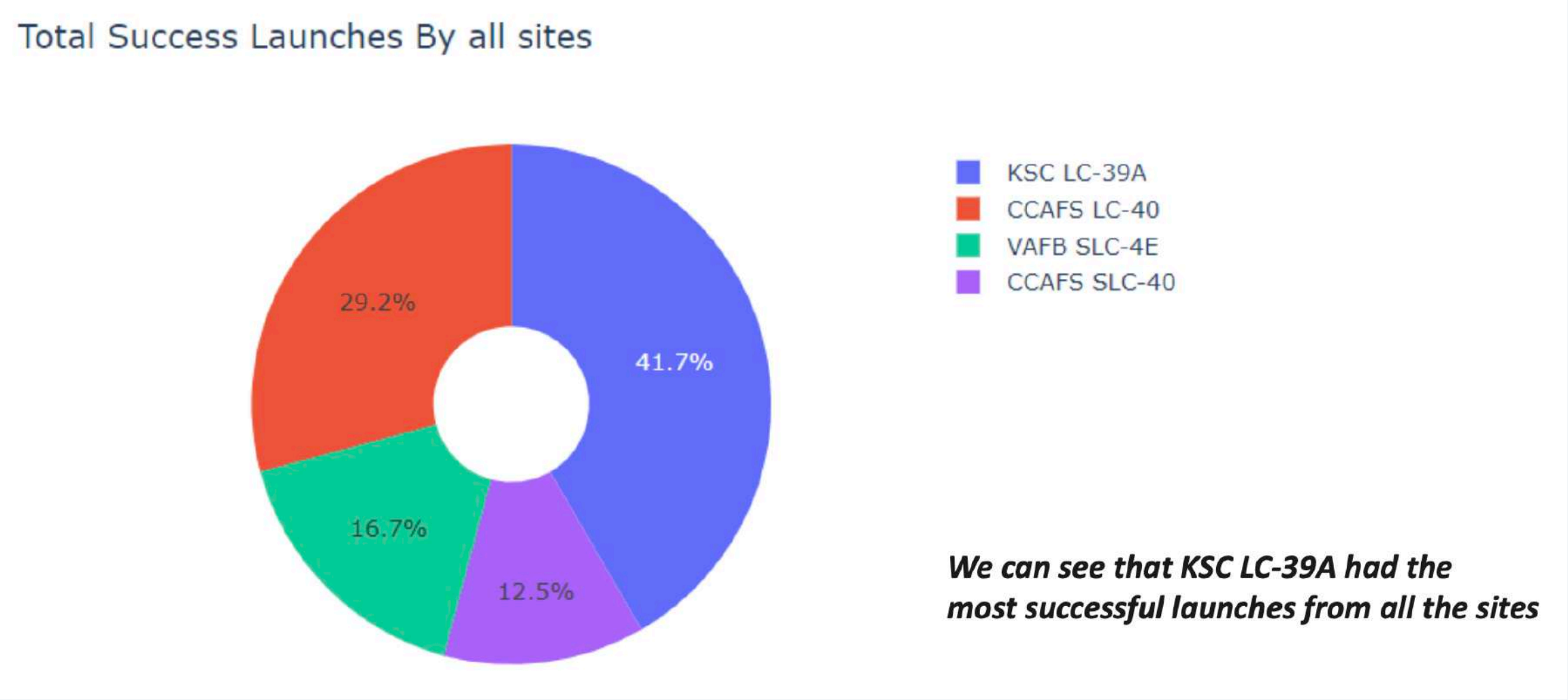
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

Build a Dashboard with Plotly Dash

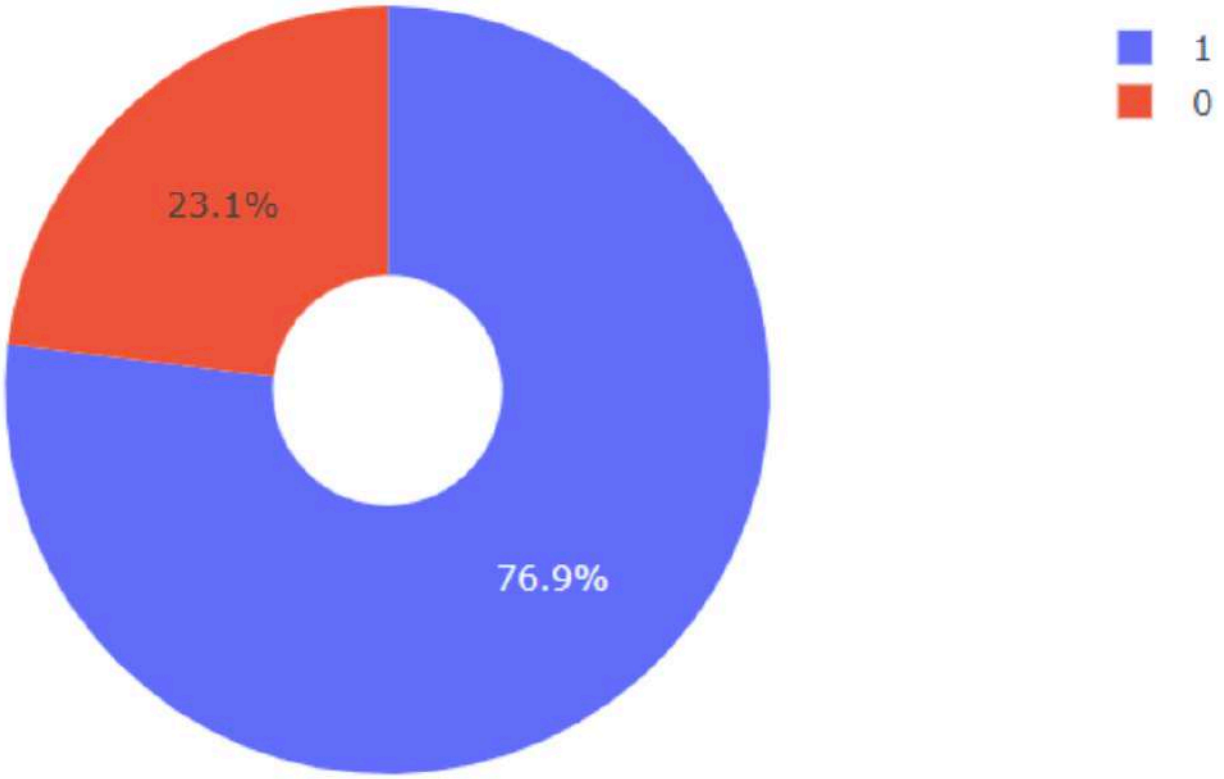


Pie chart showing the success percentage by each launch site



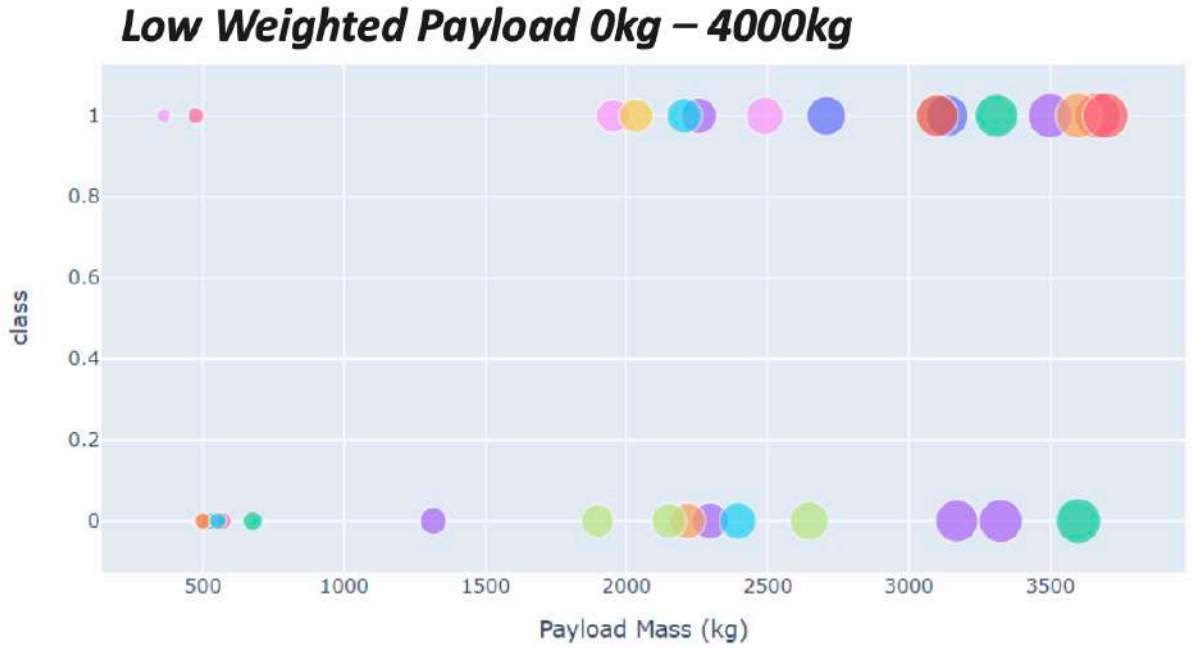
We can see that KSC LC-39A had the most successful launches from all the sites

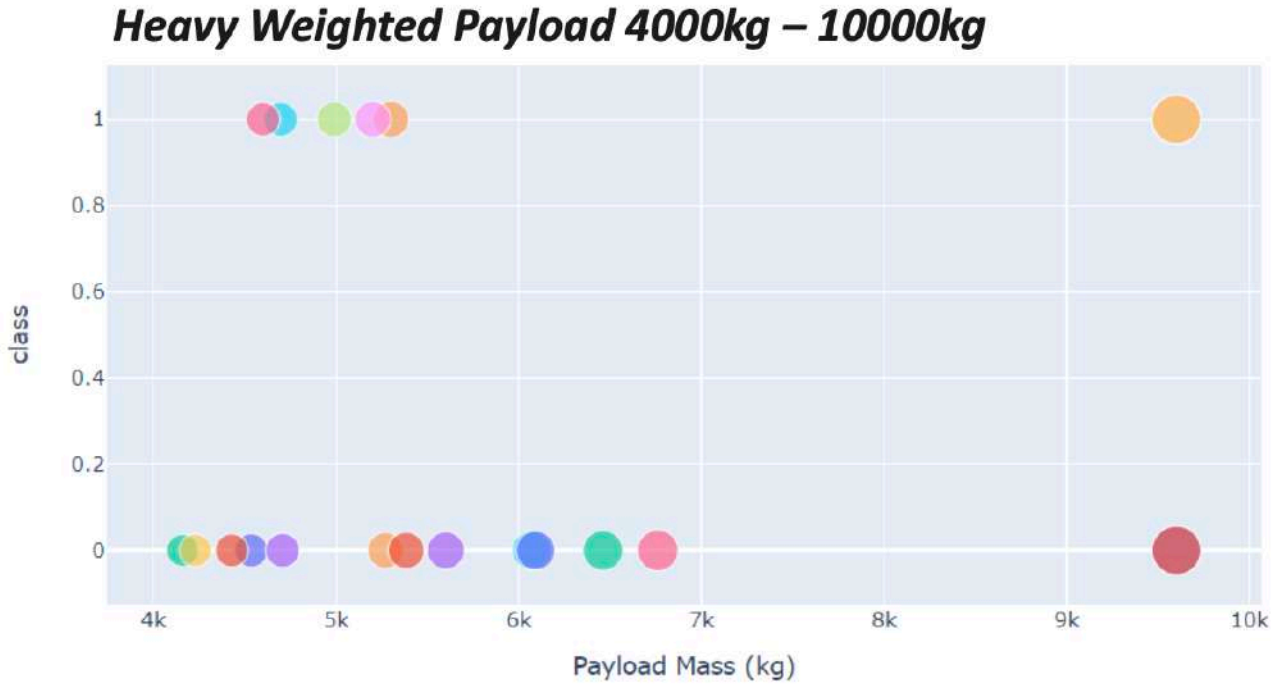
Pie chart for the launch site with highest launch success ratio



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

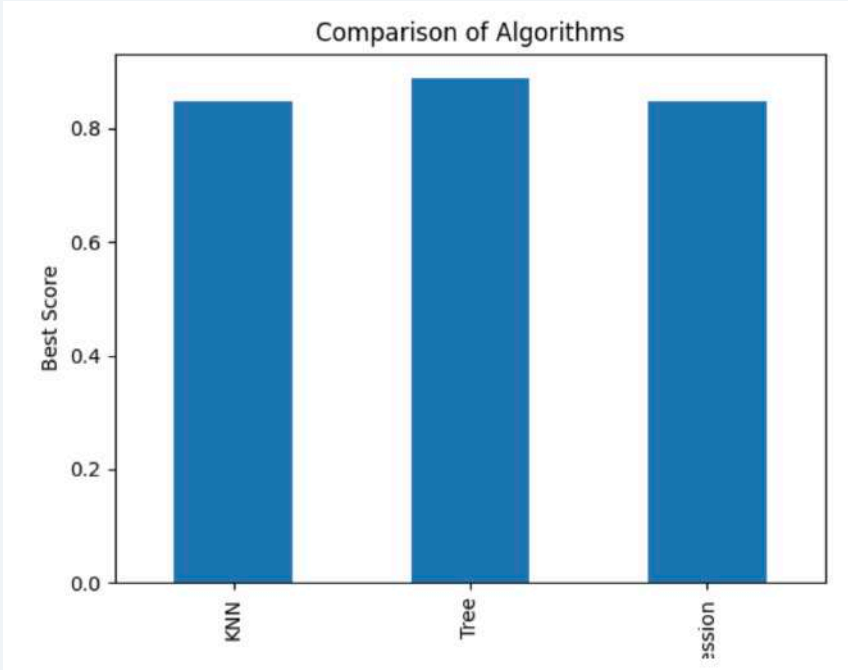
Predictive Analysis (Classification)

Classification Accuracy

- The tree algorithm wins!!

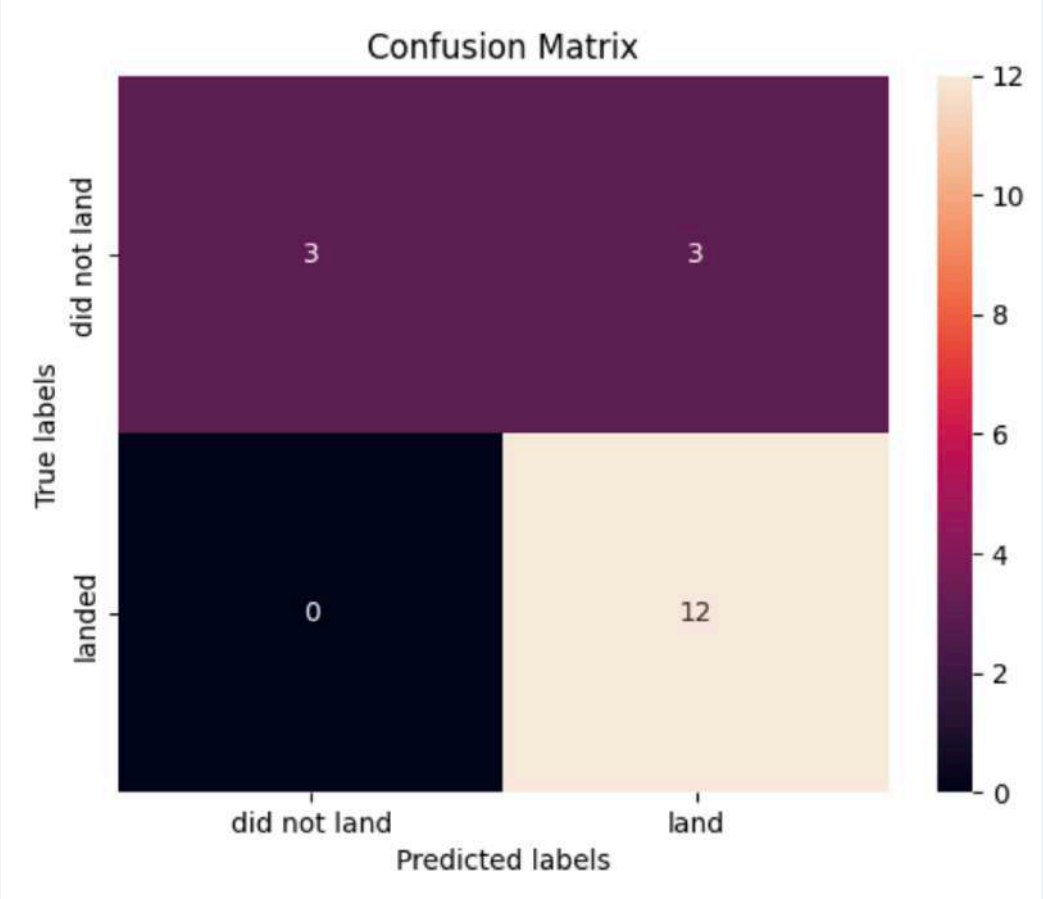
	Best Score
KNN	0.848214
Tree	0.887500
LogisticRegression	0.846429

```
Best Algorithm is Tree with a score of 0.8875
Best Params is : {'criterion': 'entropy', 'max_depth': 8,
'max_features': 'auto', 'min_samples_leaf': 1,
'min_samples_split': 2, 'splitter': ‘best'}
```



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

- 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!