



IBM Developer  
SKILLS NETWORK

# Winning Space Race with Data Science

Rhodri Owen  
06.12.2021



# Outline

---

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

**GitHub repository:** <https://github.com/rhod-dev/IBM-datascience-capstone-spacex>

# 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 and context
  - We predicted if the Falcon 9 first stage would 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 alternative company wants to bid against SpaceX for a rocket launch.
- Common problems
  - What influences rocket landing success?
  - 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 optimal results and ensure the best rocket success landing rate



Section 1

# Methodology

# Methodology

---

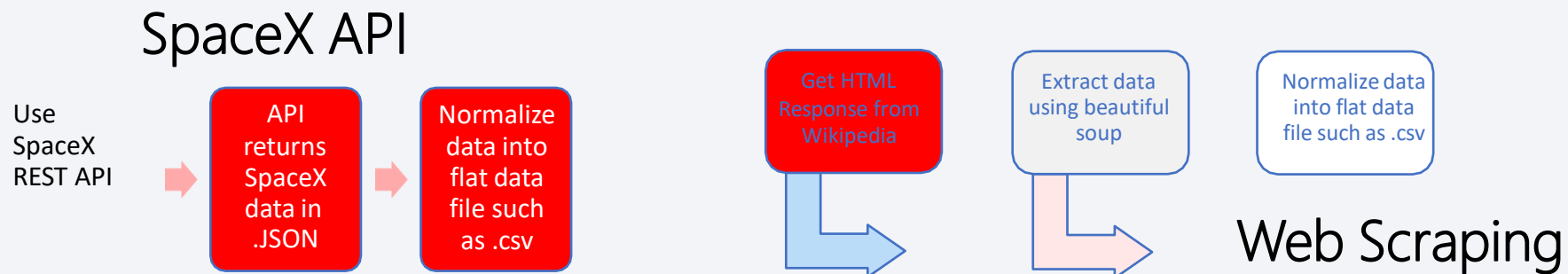
## Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web scraping from Wikipedia
- Perform data wrangling
  - Data was preprocessed and transformed using One-Hot Encoding (from categorical data to 1,0 for us in ML algorithms).
  - Irrelevant data removed (columns).
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Plotted scatter graphs and bar charts to visualize relationship between chosen variables and to reveal patterns.
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

# Data Collection

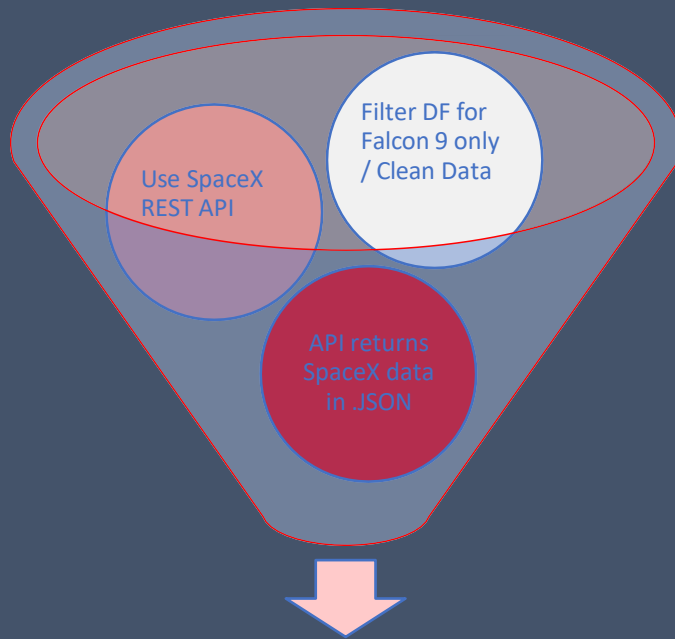
---

- The following datasets were collected:
  - SpaceX launch data was gathered from the SpaceX REST API
  - This will give us data on rocket type, payload, launch specifications, landing specifications, and landing outcome.
  - Our goal is to use this data to predict whether SpaceX will attempt to land a rocket.
  - SpaceX REST API endpoints begin with `api.spacxdata.com/v4/...`
  - Flacon 9 launch data can also be scraped from Wikipedia using the BeautifulSoup library.



[https://github.com/rhod-dev/IBM-datascience\\_capstone-spacex/blob/302be62c9b0253a1ed8d47c43f3bb9a73a71a64e/Data%20collection%20with%20SpaceX%20API](https://github.com/rhod-dev/IBM-datascience_capstone-spacex/blob/302be62c9b0253a1ed8d47c43f3bb9a73a71a64e/Data%20collection%20with%20SpaceX%20API)

## Data collection – SpaceX API



Normalize data into flat data file such as .csv

### 1. Getting Response from API

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

### 2. Converting Response to a .json file

```
static_json_url='https://cf-courses-data.s3.us.cloudfront.net/assets/courses/02-001-01/data.json'
response_content = response.json()
data = pd.json_normalize(response_content)
```

### 3. Apply custom functions to clean data

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

```
getBoosterVersion(data)
```

### 4. Assign list to dictionary then dataframe

```
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}
```

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

### 5. Filter dataframe and export to flat file (.csv)

```
data_falcon9 = df[(df.BoosterVersion == "Falcon 9")]
```

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

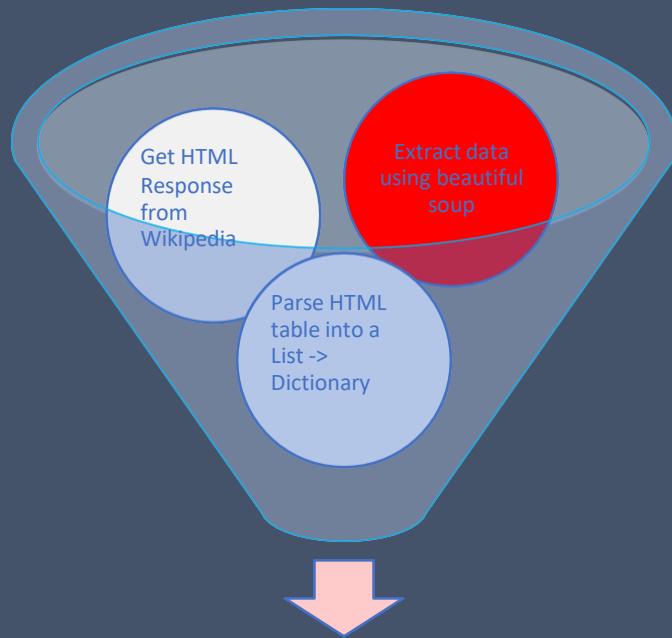
*simplified flow chart*

8

8



# Data collection – Web Scraping



Normalize data into flat data file such as .csv

## 1 .Getting Response from HTML

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

## 2. Creating BeautifulSoup Object

```
soup = BeautifulSoup(response.content, 'html5lib')
```

### 3. Finding tables

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

## 4. Getting column names

```
column_names = []
for i in first_launch_table.find_all('th'):
    column_names.append(extract_column_from_header(i))
```

## 5. Creation of dictionary

```
launch_dict= dict.fromkeys(column_names)
```

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

```
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'] = []
launch_dict['Version Booster']=[ ]
launch_dict['Booster landing']=[ ]
launch_dict['Date']=[ ]
launch_dict['Time']=[ ]
```

## 6. Appending data to keys (refer) to notebook block 12

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

## 7. Converting dictionary to dataframe

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

## 8. Dataframe to .CSV

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

# Data Wrangling

<https://github.com/rhod-dev/IBM-datascience-capstone-spacex/blob/302be62c9b0253a1ed8d47c43f3bb9a73a71a64e/Data%20wrangling%20EDA.ipynb>

## Introduction

In the data set, 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 means the mission outcome was unsuccessfully landed to a ground pad. True ASDS means 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 where 1 means the booster successfully landed and 0 means it was unsuccessful.

## Data wrangling process

Perform Exploratory Data Analysis EDA on dataset

Calculate the number of launches at each site

Calculate the number and occurrence of each orbit

Calculate the number and occurrence of mission outcome per orbit type

Export dataset as .CSV

Create a landing outcome label from Outcome column

Work out success rate for every landing in dataset

# EDA with Data Visualization

---

<https://github.com/rhod-dev/IBM-datascience-capstone-spacex/blob/302be62c9b0253a1ed8d47c43f3bb9a73a71a64e/EDA%20with%20Visualisation.ipynb>

What charts were plotted and why did we use them?

## Scatter Graphs:

- 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 Graphs:

- Mean Vs Orbit

## Line Graphs:

- Success rate Vs Year

# EDA with SQL

<https://github.com/rhod-dev/IBM-datascience-capstone-spacex/blob/302be62c9b0253a1ed8d47c43f3bb9a73a71a64e/EDA%20with%20SQL.ipynb>

---

- SQL queries performed to gather information on 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 achieved.
  - 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

<https://github.com/rhod-dev/IBM-datascience-capstone-spacex/blob/302be62c9b0253a1ed8d47c43f3bb9a73a71a64e/Interactive%20Visual%20Analytics%20with%20Folium.ipynb>

---

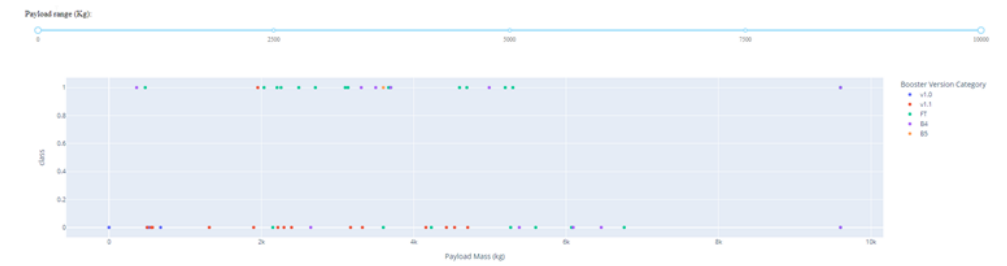
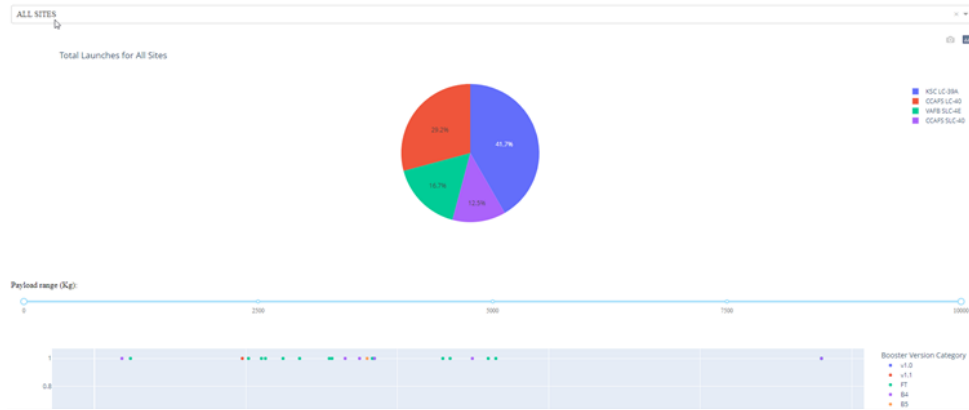
- To visualize the Launch Data into an interactive map we took the Latitude and Longitude Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.
- We assigned the dataframe `launch_outcomes(failures, successes)` to classes 0 and 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

- The dashboard is built with Plotly
- Pie Chart showing the total launches by a certain site/all sites
  - Displayed 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, i.e. maximum and minimum value, can be determined.
  - Observation and reading are straightforward.

## SpaceX Launch Records Dashboard



# 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 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 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



The background of the slide is an abstract composition. It features a solid blue area on the left side, which transitions into a dynamic pattern of diagonal streaks in shades of blue and red on the right. These streaks are layered over a fine, light-colored grid, creating a sense of depth and movement, reminiscent of a digital or data visualization theme.

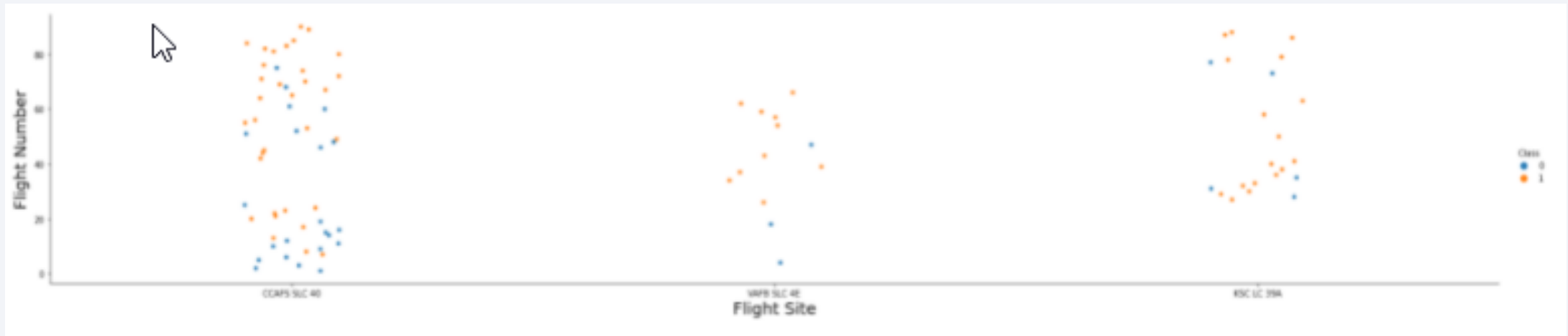
Section 2

# Insights drawn from EDA



# Flight Number vs. Launch Site

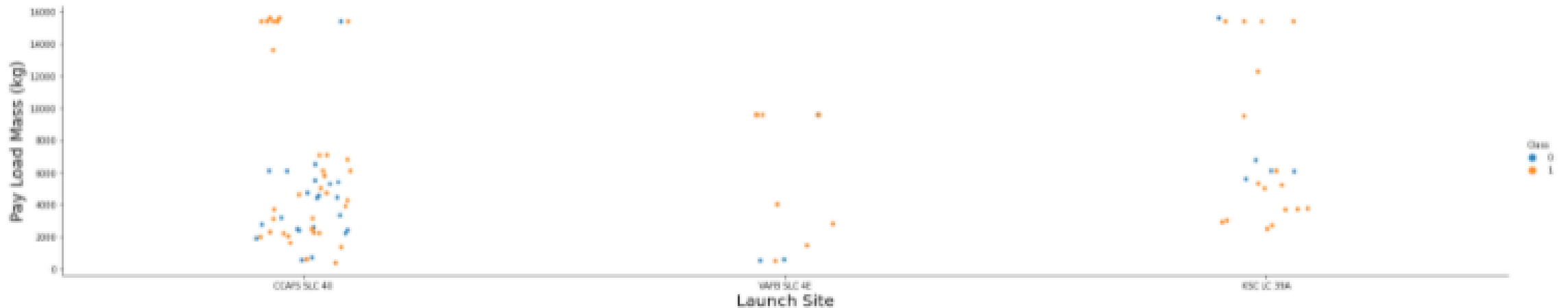
---



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

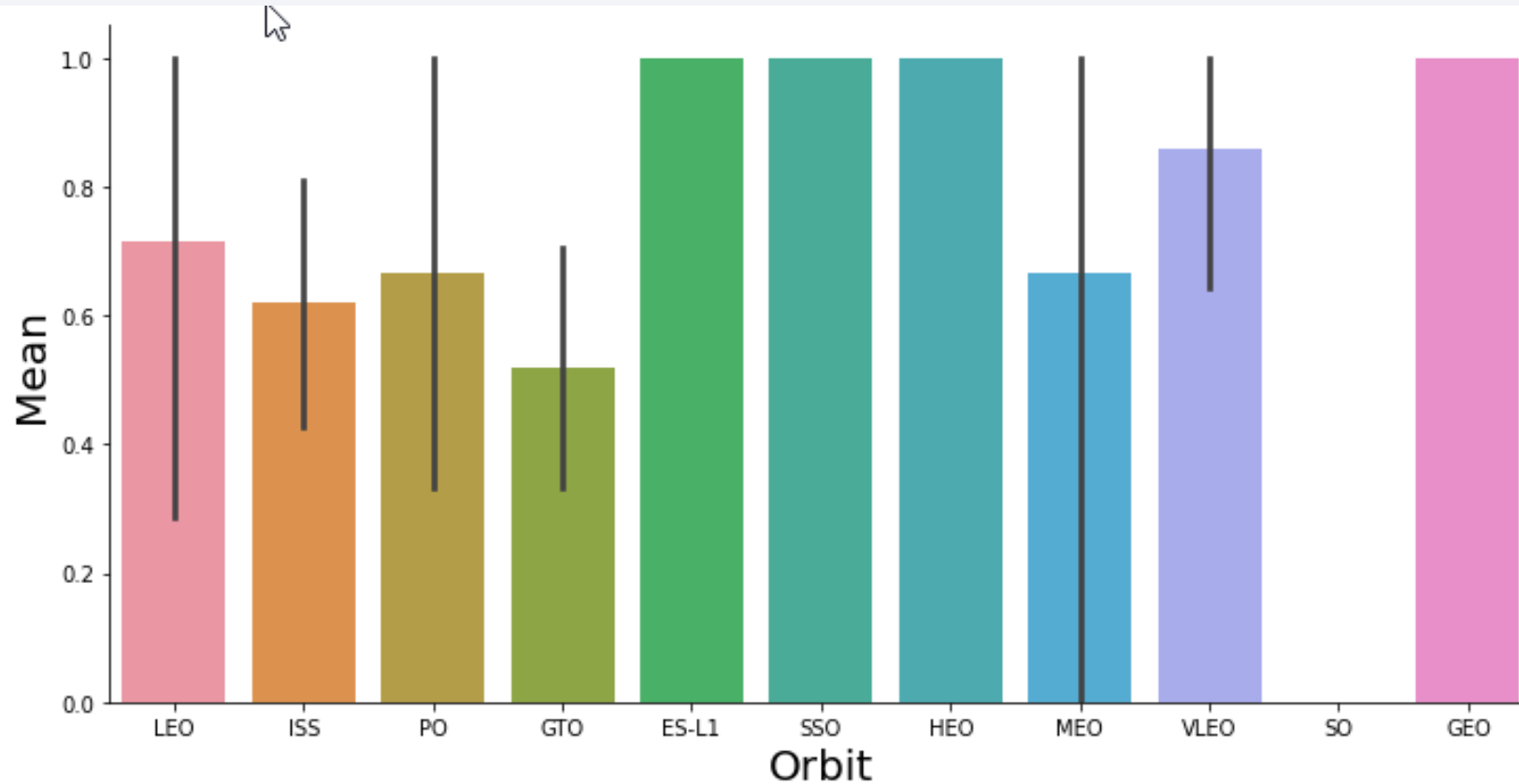
# Payload vs. Launch Site

---



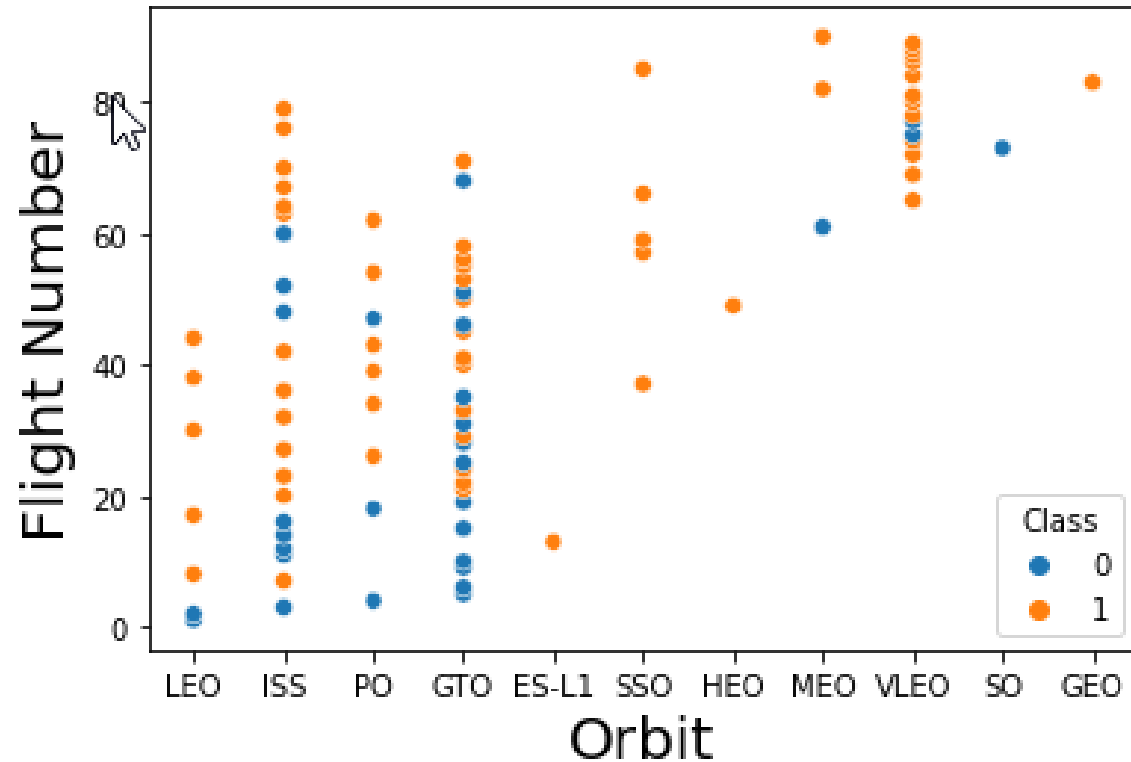
The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependent on Payload Mass for a successful launch.

# Success Rate vs. Orbit Type



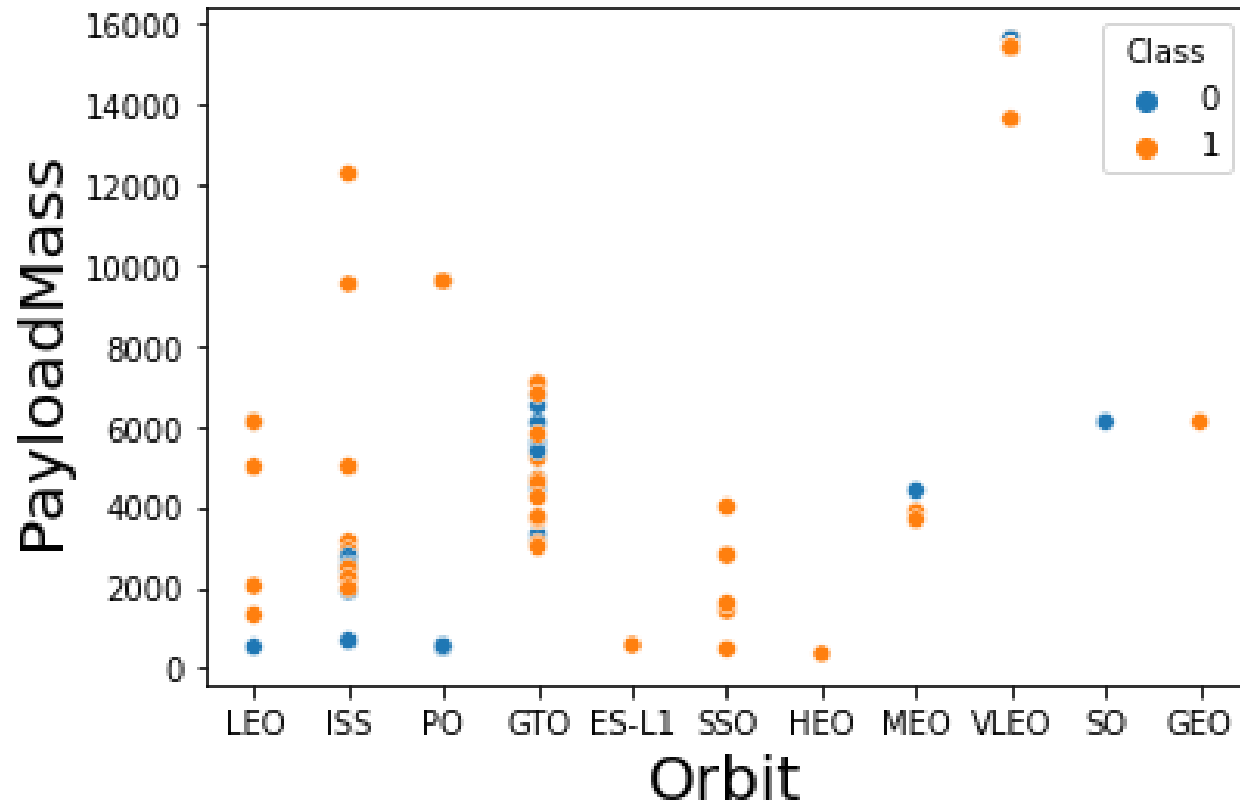
Orbits GEO, HEO, SSO and ES-L1 have the best Success Rate

# 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.

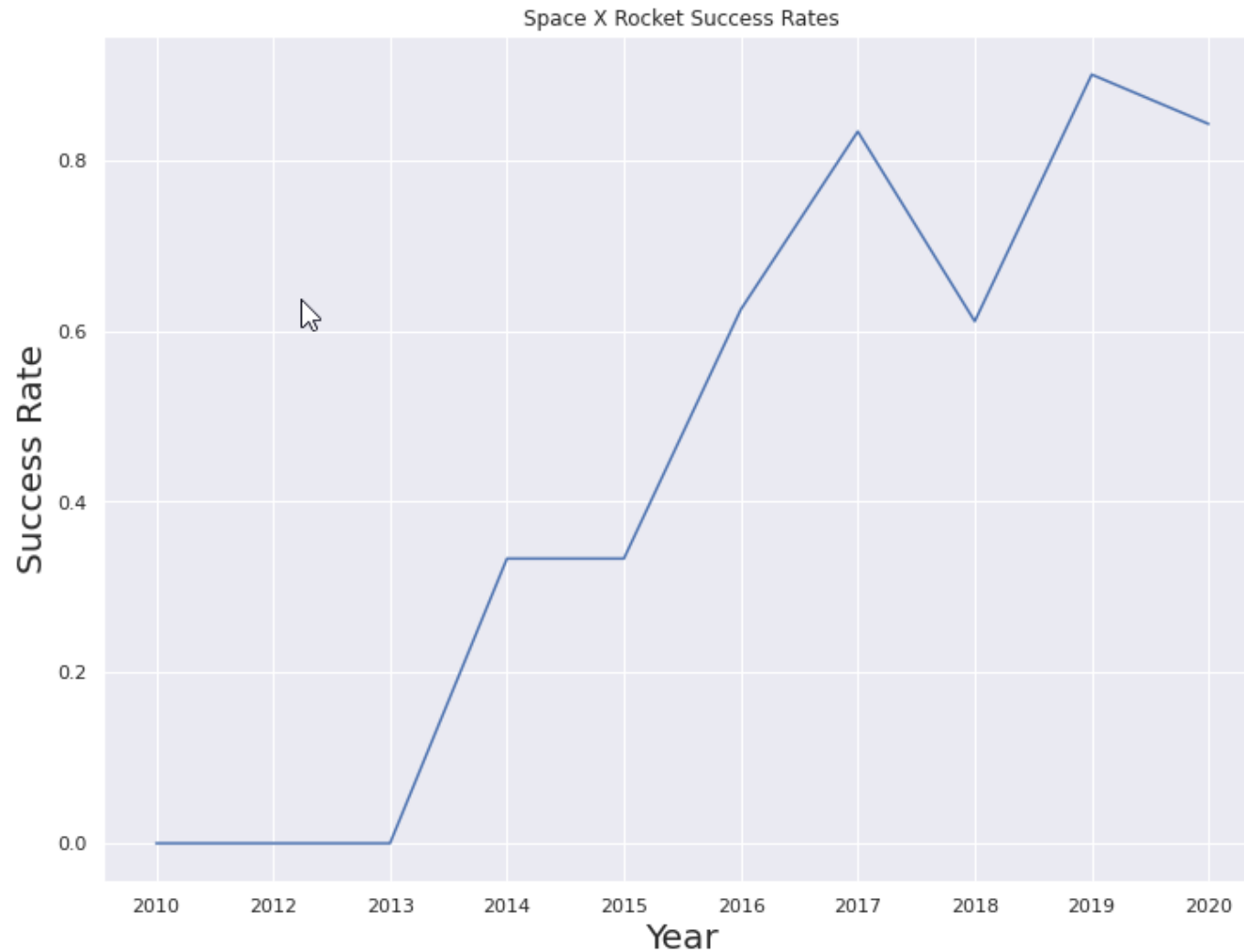
# Payload vs. Orbit Type



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



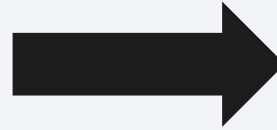
You can observe that the success rate since 2013 kept increasing till 2020

# All Launch Site Names

---

## SQL QUERY

```
%sql select distinct(LAUNCH_SITE) from QTM64119.SPACEXDATASET
```



Unique Launch Sites
CCAFS LC-40
CCAFS SLC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

## QUERY EXPLANATION

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 KSC'

## SQL QUERY

```
%sql select * from QTM64119.SPACEXDATASET where LAUNCH_SITE like 'KSC%' limit 5
```



## QUERY EXPLANATION

Using the word **TOP 5** in the query means that it will only show 5 records from **the table** and **LIKE** keyword has a wild card with the words '**KSC%**' the percentage in the end suggests that the Launch\_Site name must start with KSC.

DATE	time_utc_	booster_version	launch_site	payload	payload_mass_kg_	orbit	customer	mission_outcome	landing_outc
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (gr
2017-03-16	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No att
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (d
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (gr
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No att

# Total Payload Mass

---

## SQL QUERY

```
%sql select sum(PAYLOAD_MASS__KG_) from QTM64119.SPACEXDATASET where CUSTOMER = 'NASA (CRS)'
```



Total Payload Mass	
0	45596

## QUERY EXPLANATION

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_) from QTM64119.SPACEXDATASET where BOOSTER_VERSION = 'F9 v1.1'
```



2928

## QUERY EXPLANATION

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 min(DATE) from QTM64119.SPACEXDATASET where Landing__Outcome = 'Success (drone ship)'
```



2016-04-08

# Successful Drone Ship Landing with Payload between 4000 and 6000

---

## SQL QUERY

```
%sql select BOOSTER_VERSION from QTM64119.SPACEXDATASET where Landing_Outcome = 'Success (ground pad)' and PAYLOAD_MASS_KG_ > 4000 and PAYLOAD_MASS_KG_ < 6000
```



booster_version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

### QUERY EXPLANATION

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 count(MISSION_OUTCOME) from QTM64119.SPACEXDATASET where MISSION_OUTCOME = 'Success' or MISSION_OUTCOME = 'Failure (in flight)'
```



100

## QUERY EXPLANATION

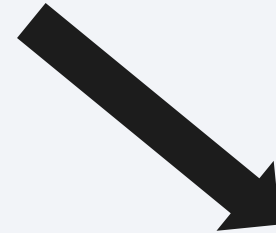
A much harder query, we used subqueries here to produce the results.

# Boosters Carried Maximum Payload

---

## SQL QUERY

```
%sql select BOOSTER_VERSION from QTM64119.SPACEXDATASET where PAYLOAD_MASS_KG_ = (select max(PAYLOAD_MASS_KG_) from QTM64119.SPACEXDATASET)
```



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

# 2017 Launch Records

---

## SQL QUERY

```
%sql select DISTINCT BOOSTER_VERSION, MAX(PAYLOAD_MASS_KG_) as [Maximum Payload Mass] from QTM64119.SPACEXDATASET GROUP BY Booster_Version ORDER BY [Ma
```

## QUERY EXPLANATION

Failed to accurately execute this query

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

## SQL QUERY

```
%sql select * from QTM64119.SPACEXDATASET where Landing__Outcome like 'Success%' and (DATE between '2010-06-04' and '2017-03-20') order by date desc
```

### QUERY EXPLANATION

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

**LIKE (wildcard)**  
**AND (conditions)**  
**AND (conditions)**




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

Section 4

# Launch Sites Proximities Analysis





VAFB  
SLC-  
4E

KSC  
FS  
SCC-  
40A

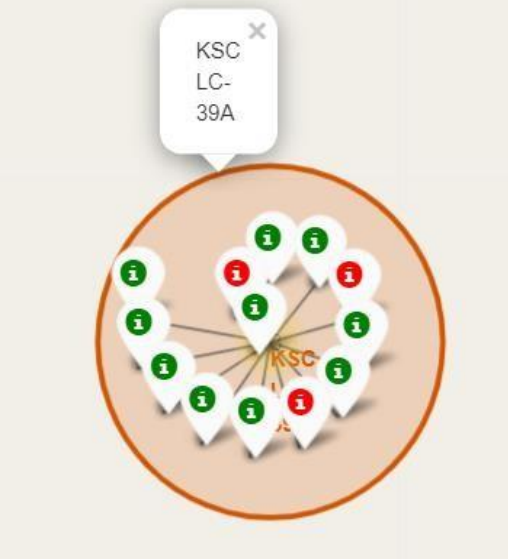
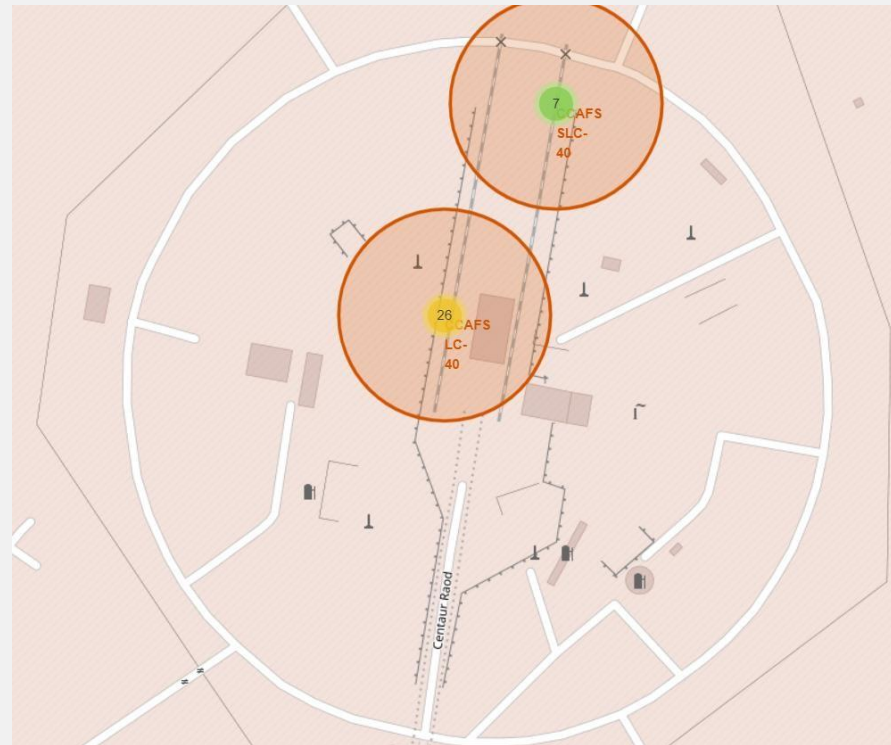
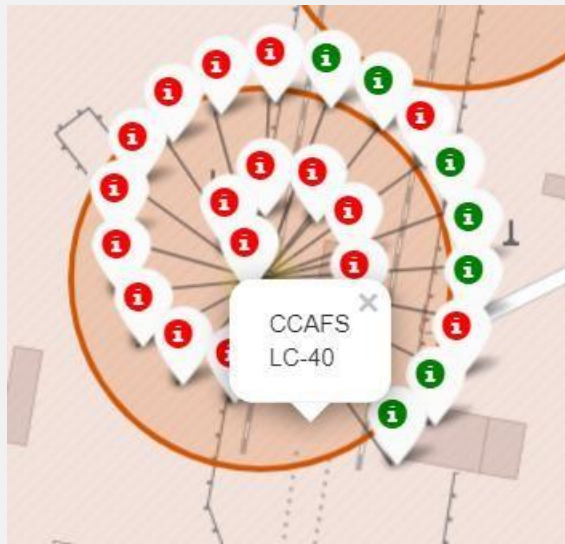
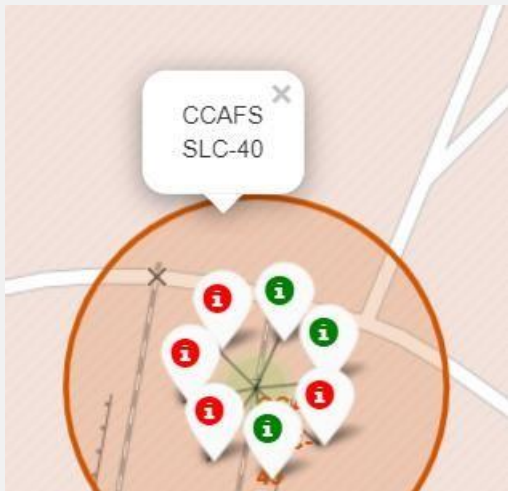
# All launch sites global map markers

---

*We can see that the SpaceX launch sites are in the United States of America coasts.  
Florida and California*

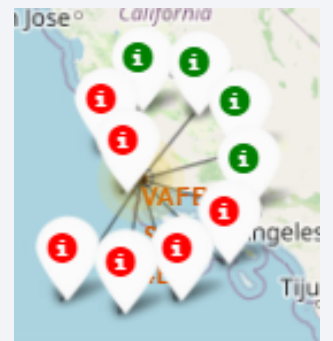


# Colour Labelled Markers



**Florida Launch Sites**

*Green Marker shows successful Launches and Red Marker shows Failures*



**California Launch Site**



Section 5

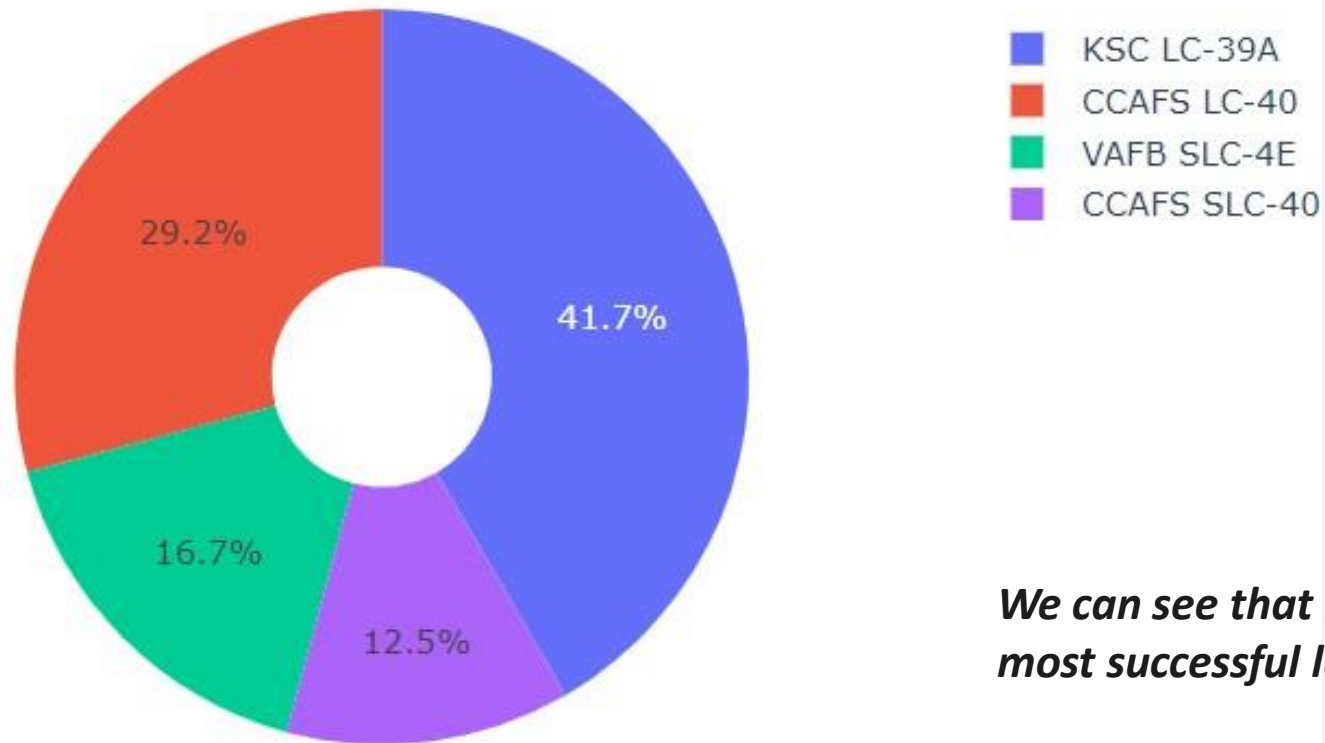
# Build a Dashboard with Plotly Dash



## Pie chart showing success percentage achieve by each launch site

---

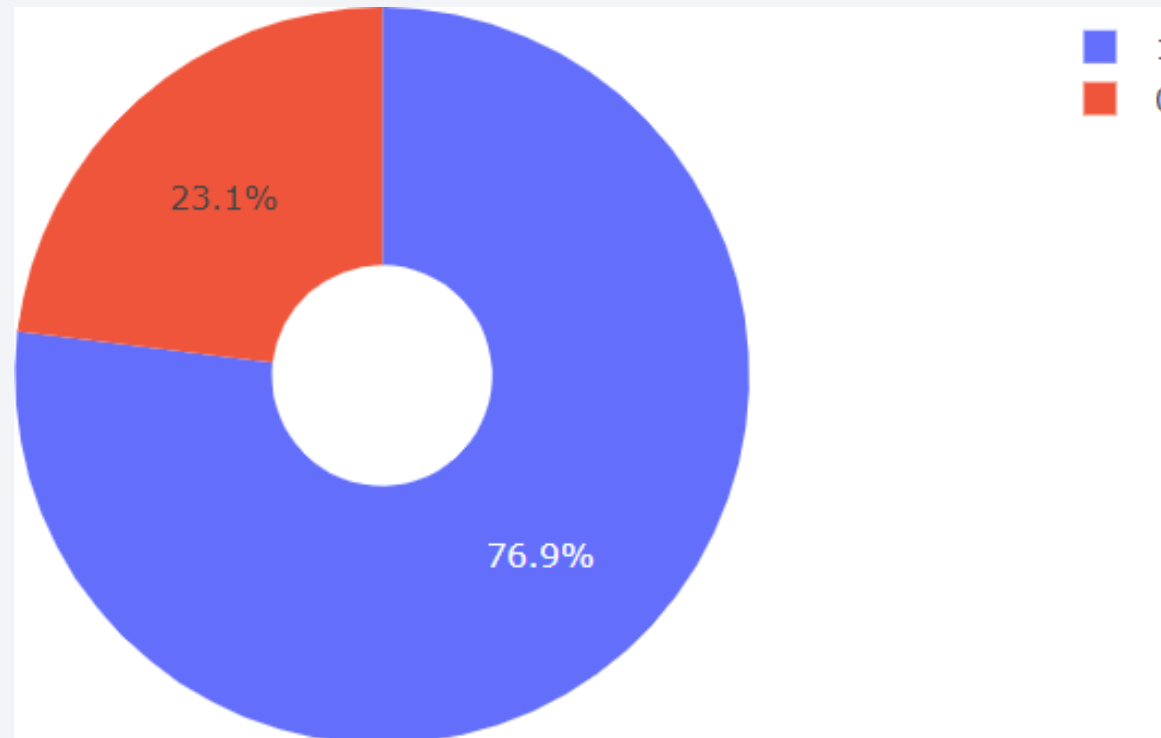
Total Success Launches By all sites



***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

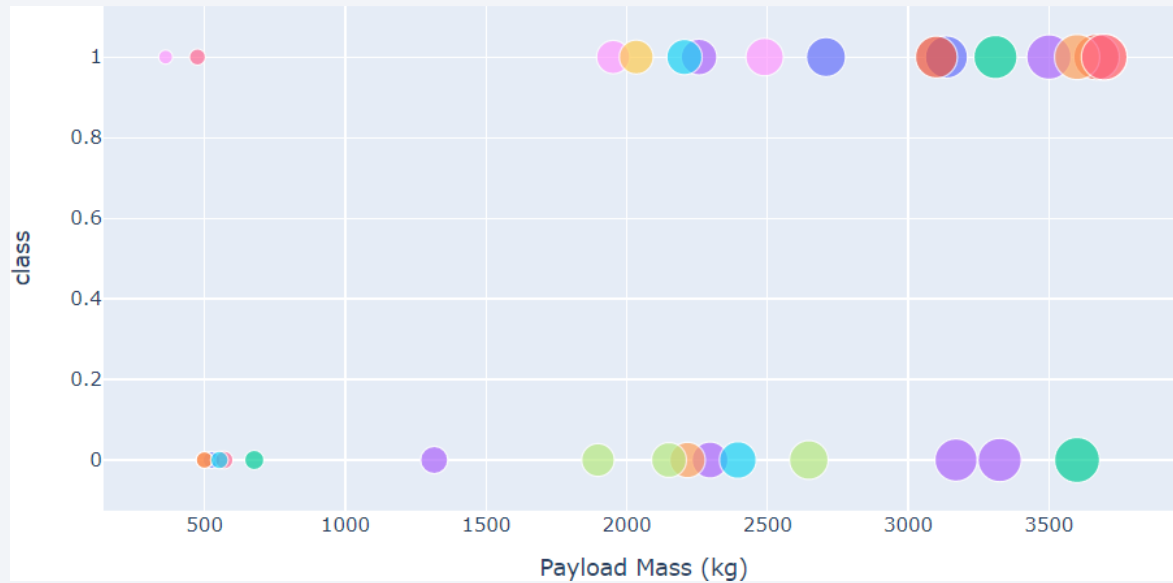
---



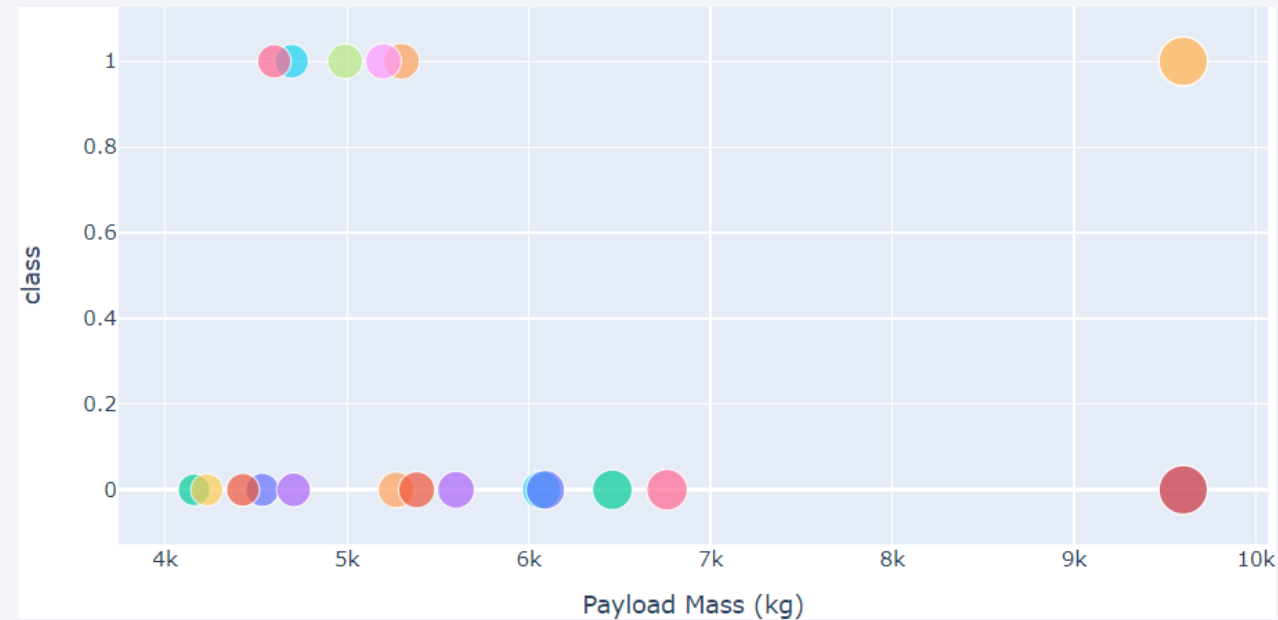
***KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate***

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

**Low Weighted Payload 0kg – 4000kg**



**Heavy Weighted Payload 4000kg – 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

---

```
algorithms = {'KNN':knn_cv.best_score_, 'Tree':tree_cv.best_score_, 'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

*The tree algorithm wins*

```
Best Algorithm is Tree with a score of 0.8892857142857142
Best Params is : {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 4,
'min_samples_split': 5, 'splitter': 'random'}
```

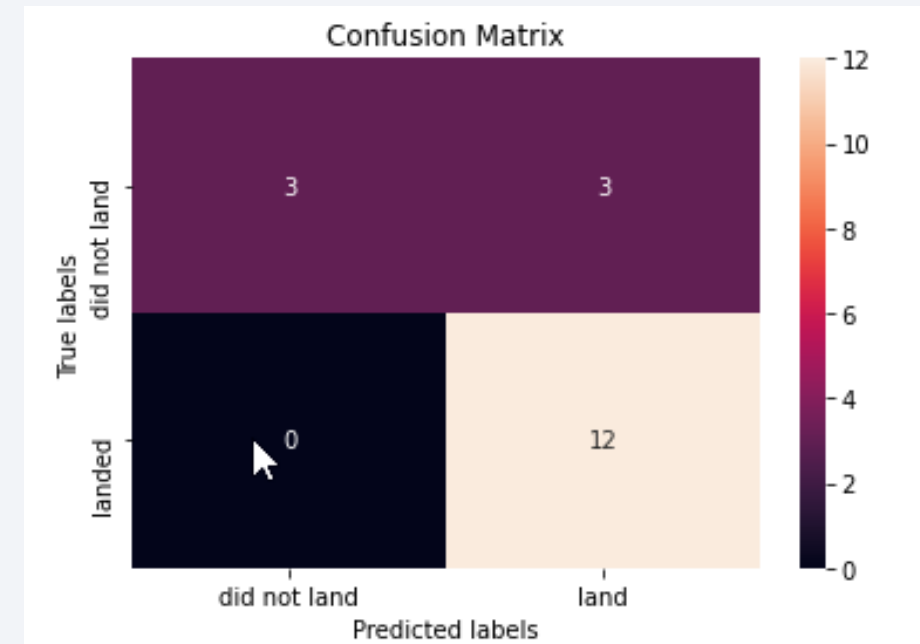
After selecting the best hyperparameters for the decision tree classifier using the validation data, we achieved 83.33% accuracy on the test data.



# 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.

		Predicted Values	
		Negative	Positive
Actual Values	Negative	TN	FP
	Positive	FN	TP



# 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!

