SPACEX PROJECT PRESENTATION

Claudio Aceto

15th Dec 2021

OUTLINE

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

EXECUTIVE SUMMARY

- Data Collection – REST API
- Data Collection – Web Scraping
- Data Wrangling
- EDA with data visualization
- EDA with SQL
- Map and distance with Folium
- Dashboard with Plotty
- Machine learning prediction (Best classification score)

INTRODUCTION

- Project background

We want predict that Falcon 9 first stage will land successful. SpaceX advertises Falcon 9 rocket launches, with a cost of 62 million dollars other providers cost of 165 million dollars each, 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 that need to solve
- if the rocket will land successfully?
- Relationship with certain rocket variables will impact in determining the success rate of a successful landing.
    - Conditions does SpaceX have to get the best results and ensure the best rocket success landing rate.

METHODOLOGY

- Data Collection:
- SpaceX Rest API
- Web scraping from Wikipedia
- Data Wrangling:
- Transforming data for Machine Learning and dropping irrelevant features for project
- Exploration with Data Analysis using visualization and SQL
- Interactive visualizations using a map with Folium and Dashboard with Plotty
- Predictive analysis using Classifications methods
  - Build
  - Evaluate model
  - Find the best performance Classification

DATA COLLECTION

The dataset was collecting by:

- We worked with SpaceX launch from 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 use this data to predict SpaceX will attempt to land a rocket or not
- SpaceX REST API URL, id api.spacexdata.com/v4/
- Used data source for obtaining Falcon 9 Launch data is web scraping Wikipedia using BeautifulSoup.

DATA COLLECTION – REST API

DATA COLLECTION – REST API

1. Getting request URL from SpaceX Rest API

```
[n [6]:
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
[n [7]:
response = requests.get(spacex_url)
```

2. Covert reponse to .json file

```
In [14]:
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

3. Use a custom function to clean data

```
In [21]:
# Call getLaunchSite
getLaunchSite(data)
```

```
In [22]:
# Call getPayloadData
getPayloadData(data)
```

```
In [23]:
# Call getCoreData
getCoreData(data)
```

DATA COLLECTION – REST API

4. Construct our dataset using the data we have obtained into a dictionary

In [24]:
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

5. Apply filter to dataset and export it to a .csv file

In [28]:
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

Now that we have removed some values we should reset the FlgihtNumber column

In [29]:
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

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

DATA COLLECTION – WEB SCRAPING

DATA COLLECTION – WEB SCRAPING

1. Getting repons from HTML page

```
# use requests.get() method with the provided static_url
page= requests.get(static_url)
```

2. Create BeatifulSoup Object from HTML reponse

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

3. Finding all tables

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

4. Getting Colums names

DATA COLLECTION – WEB SCRAPING

```
column_names = []

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

5. Create a dictionary

```
launch_dict= dict.fromkeys(column_names)

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

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

DATA COLLECTION – WEB SCRAPING

6. Append data to keys

```
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()
        else:
            flag=False
```

7. Convert dictionary to dataframe

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

8. Export dataset to .csv

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

DATA WRANGLING

DATA WRANGLING

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due a problem 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 with 1 means the booster successfully landed 0 means it was unsuccessful.

DATA WRANGLING

Steps:

- 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
- Create a landing outcome label from Outcome column
- Work out success rate for every landing in dataset
- Export dataset as .CSV

RESULTS

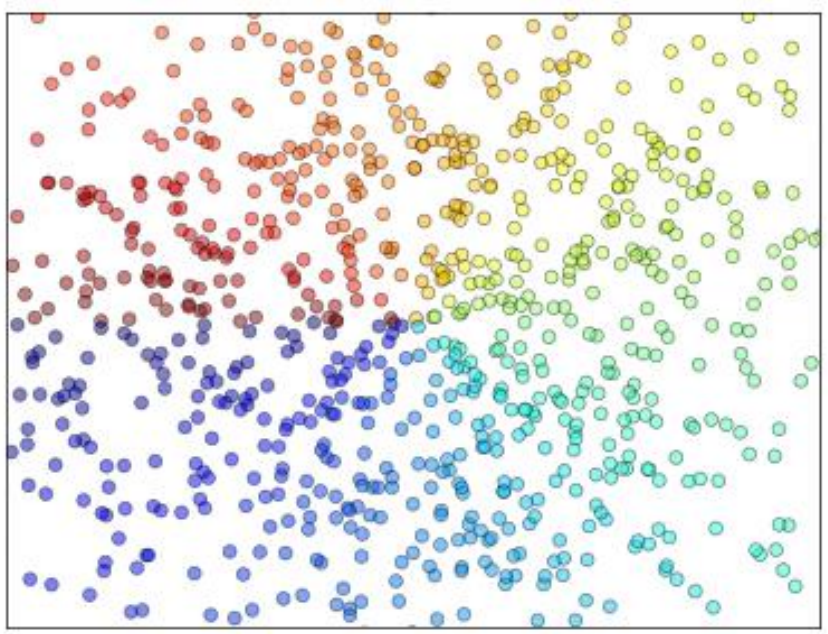
- Exploratory data analysis results
- Interactive analytics screenshots
- Predictive results

EDA WITH DATA VISUALIZATION

EDA WITH DATA VISUALIZATION

Scatter plot that has been drawn:

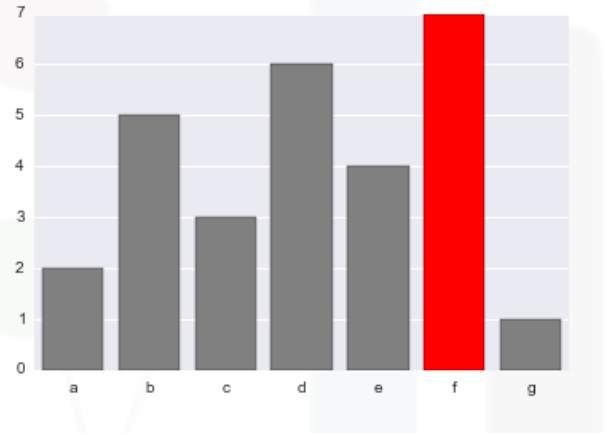
- Flight Number VS. Payload Mass
- Flight Number VS. Launch Site
- Payload VS. Launch Site
- Orbit VS. Flight Number
- Payload VS. Orbit Type
- Orbit VS. Payload Mass



EDA WITH DATA VISULIZATION

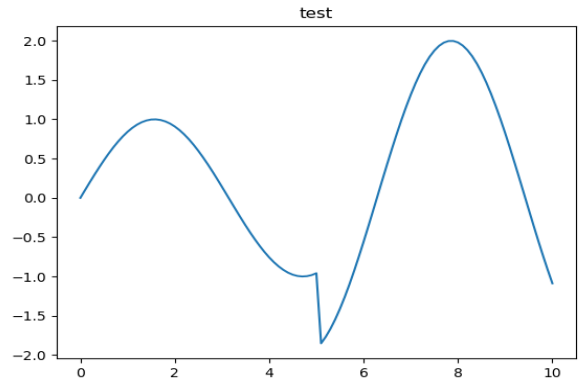
Bar plot that has been drawn:

- Mean VS. Orbit



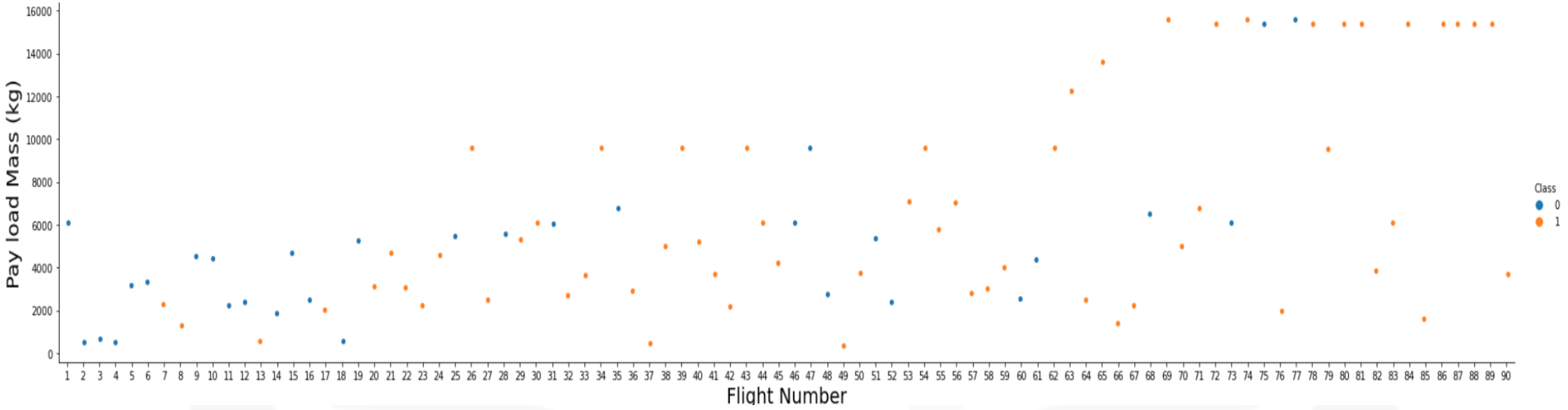
Line plot that has been drawn:

- Success Rate VS. Year



EDA WITH DATA VISUALIZATION

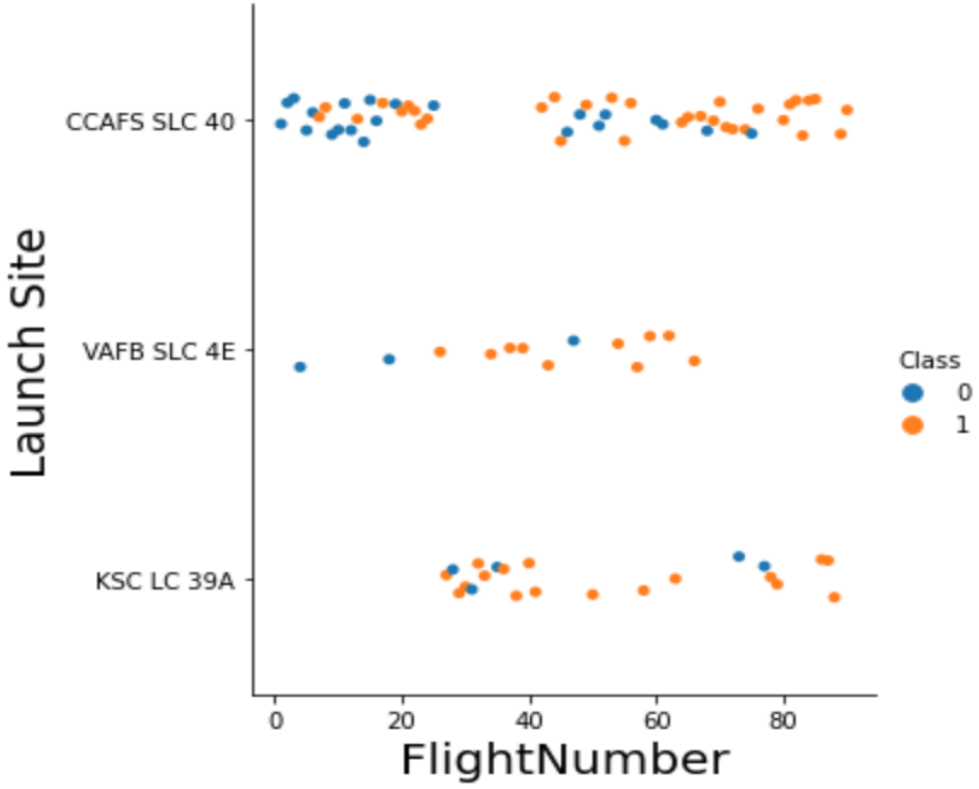
Flight Number vs. Pay Load Mass(kg)



We see that different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%

EDA WITH DATA VISUALIZATION

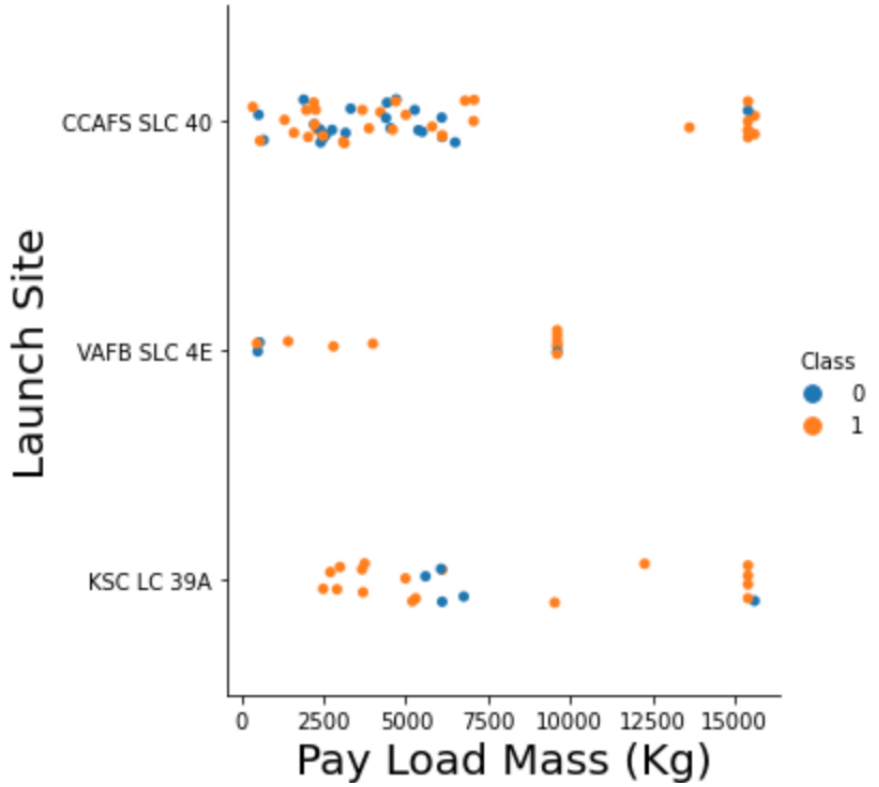
Flight Number vs. Launch Site



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

EDA WITH DATA VISUALIZATION

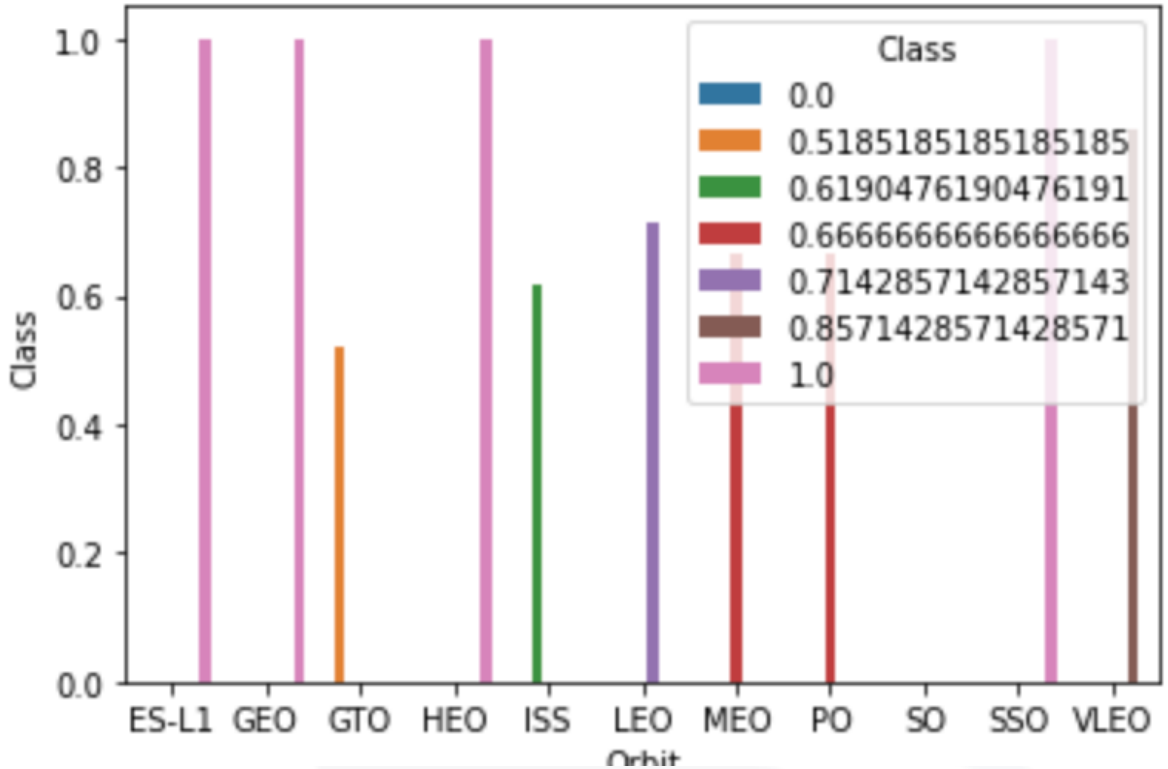
Payload Mass vs. Launch Site



if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

EDA WITH DATA VISUALIZATION

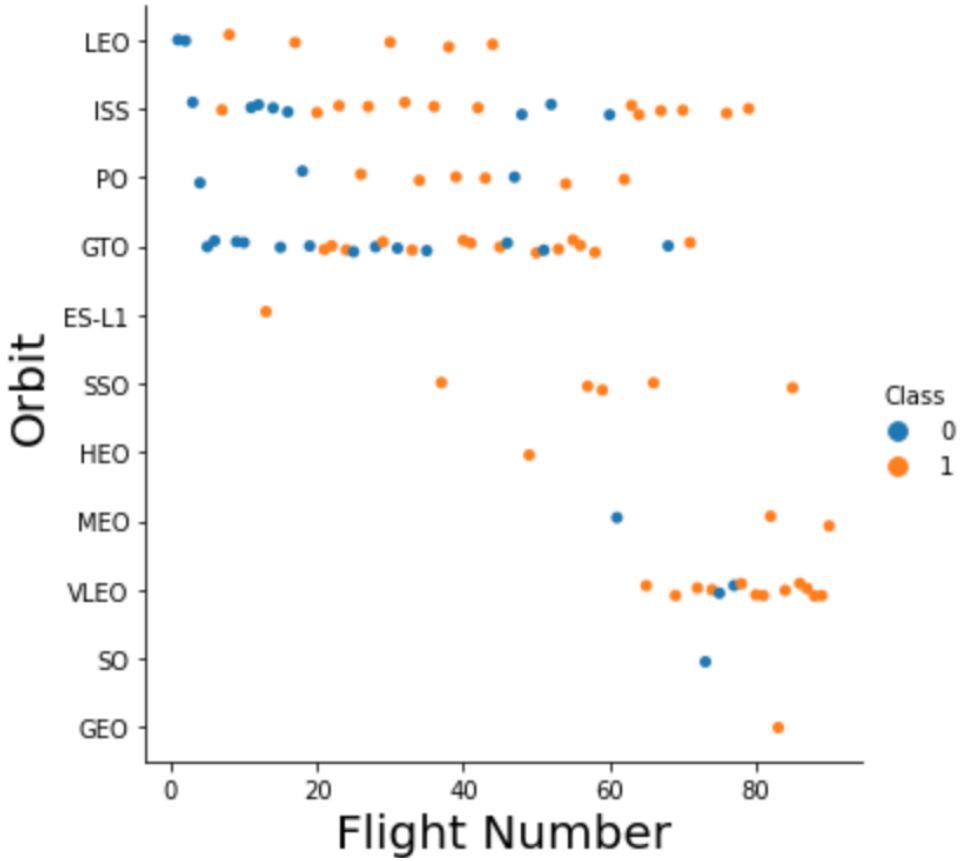
Success rate vs. Orbit type



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

EDA WITH DATA VISUALIZATION

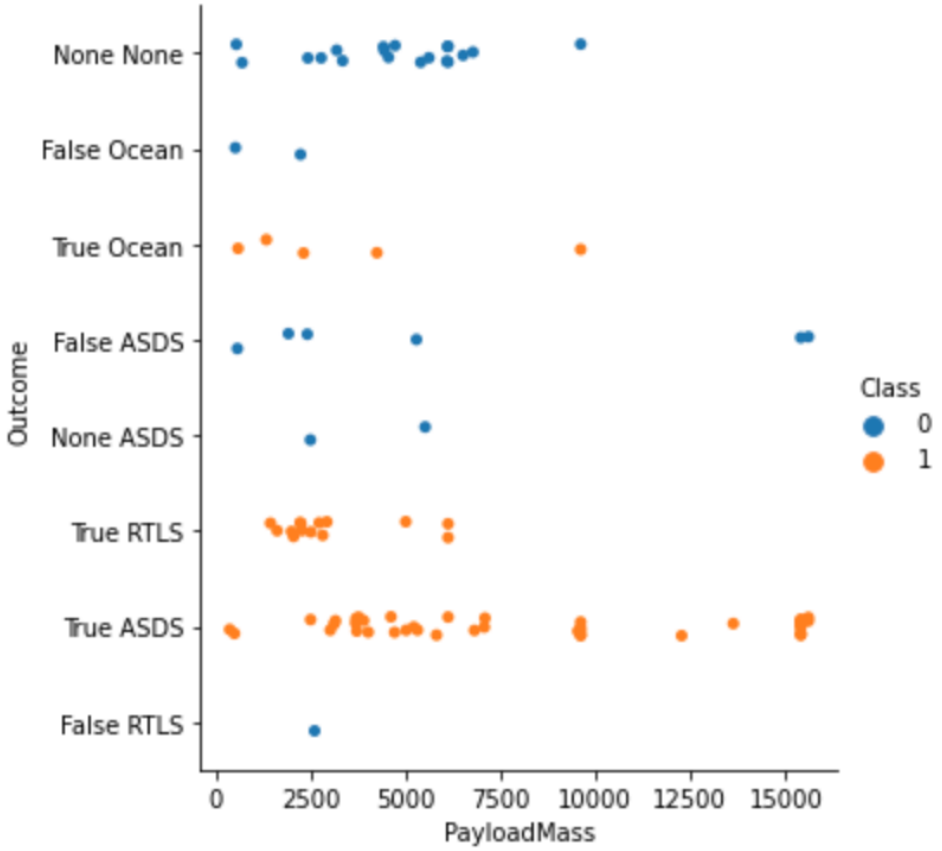
Flight Number vs. Orbit type



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.

EDA WITH DATA VISUALIZATION

Payload vs. Orbit type

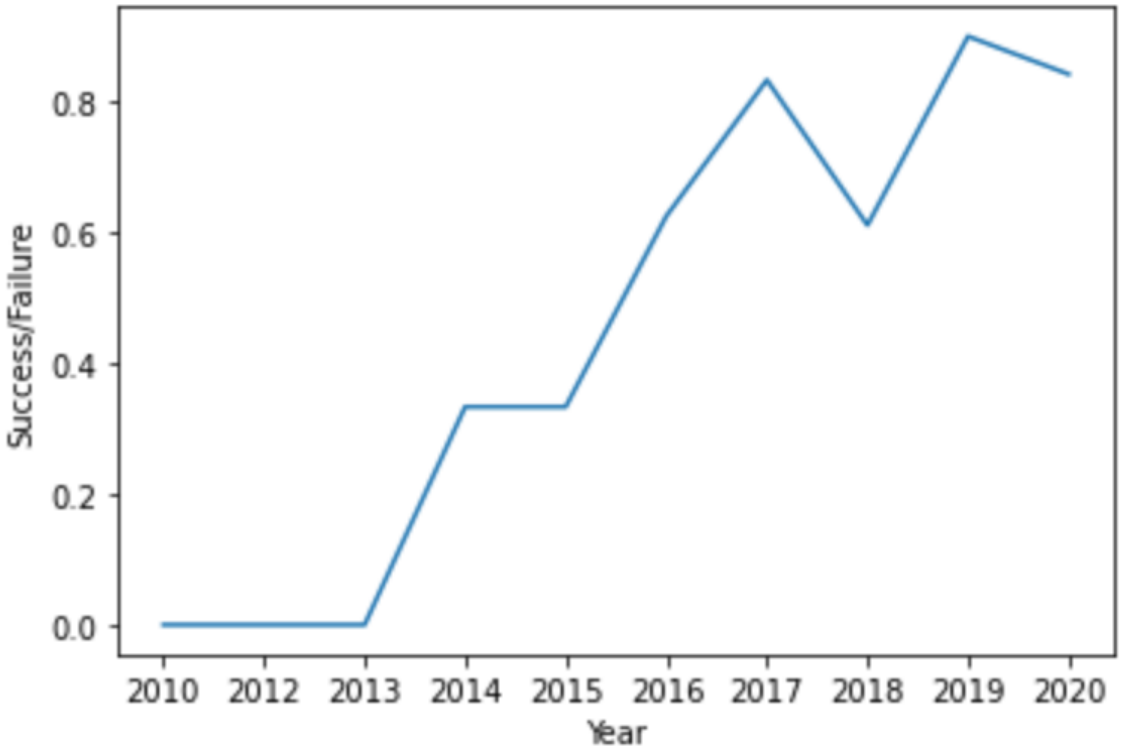


With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

EDA WITH DATA VISUALIZATION

Launch success yearly trend



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

EDA WITH SQL

EDA WITH SQL

For 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 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

EDA WITH SQL

Unique Launch Sites

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

```
%sql select Unique(LAUNCH_SITE) from SPACEXTBL
```

```
 * ibm_db_sa://fpn26770:***@55fbc997-9266-4331-
Done.
```

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

EDA WITH SQL

Launch site names begin with `CCA`

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:

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

```
 * ibm_db_sa://fpn26770:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1o
Done.
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

EDA WITH SQL

Total Payload Mass by Customer NASA (CRS)

Using the function ***SUM*** summates the total in the column ***PAYLOAD_MASS_KG_ as Payload***

```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

```
 * ibm_db_sa://fpn26770:***@55fbc997-9266-4331-afd3-888b05e734c0.b
Done.
```

payloadmass
619967

EDA WITH SQL

Average Payload Mass carried by booster version F9 v1.1

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

```
%sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

```
 * ibm_db_sa://fpn26770:***@55fbc997-9266-4331-afd3-888b05e734c0.bs
Done.
```

payloadmass
6138

EDA WITH SQL

The date where the successful landing outcome in First successful ground landing date drone ship was achieved

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

```
%sql select min(DATE) from SPACEXTBL;
```

```
 * ibm_db_sa://fpn26770:***@55fbc997-92
Done.
```

1
2010-06-04

EDA WITH SQL

Successful drone ship landing with payload between 4000 and 6000

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*

```
%sql select BOOSTER_VERSION from SPACEXTBL where LANDING__OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
```

```
 * ibm_db_sa://fpn26770:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb
Done.
```

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

EDA WITH SQL

Total Number of Successful and Failure Mission Outcomes

GROUP BY statement **groups** rows that have the same values into summary rows in **MISSION_OUTCOME,** and **COUNT** returns the number of rows that matches in **missionoutcomes**

```
%sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://fpn26770:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.
Done.
```

missionoutcomes
1
99
1

EDA WITH SQL

Boosters carried maximum payload

```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

```
 * ibm_db_sa://fpn26770:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb
Done.
```

boosterversion
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

Used **WHERE** as filter to extract the **maxium** payload from **SPACETBL**

EDA WITH SQL

2015 Launch Records

WHERE clause filters ***Year*** to be 2015, for extract record of that year after select **month** and **date**

```
%sql SELECT MONTH(DATE),MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where EXTRACT(YEAR FROM DATE)='2015';
```

```
 * ibm_db_sa://fpn26770:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb
Done.
```

1	mission_outcome	booster_version	launch_site
1	Success	F9 v1.1 B1012	CCAFS LC-40
2	Success	F9 v1.1 B1013	CCAFS LC-40
3	Success	F9 v1.1 B1014	CCAFS LC-40
4	Success	F9 v1.1 B1015	CCAFS LC-40
4	Success	F9 v1.1 B1016	CCAFS LC-40
6	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
12	Success	F9 FT B1019	CCAFS LC-40

EDA WITH SQL

Rank success count between 2010-06-04 and 2017-03-20

WHERE filter Date between 2010-06-04 and 2017-03-20 **ORDER BY** the date in description ***DESC***
means its arranging the dataset into descending order

```
%sql SELECT LANDING__OUTCOME FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' ORDER BY DATE DESC;
```

```
 * ibm_db_sa://fpn26770:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb
Done.
```

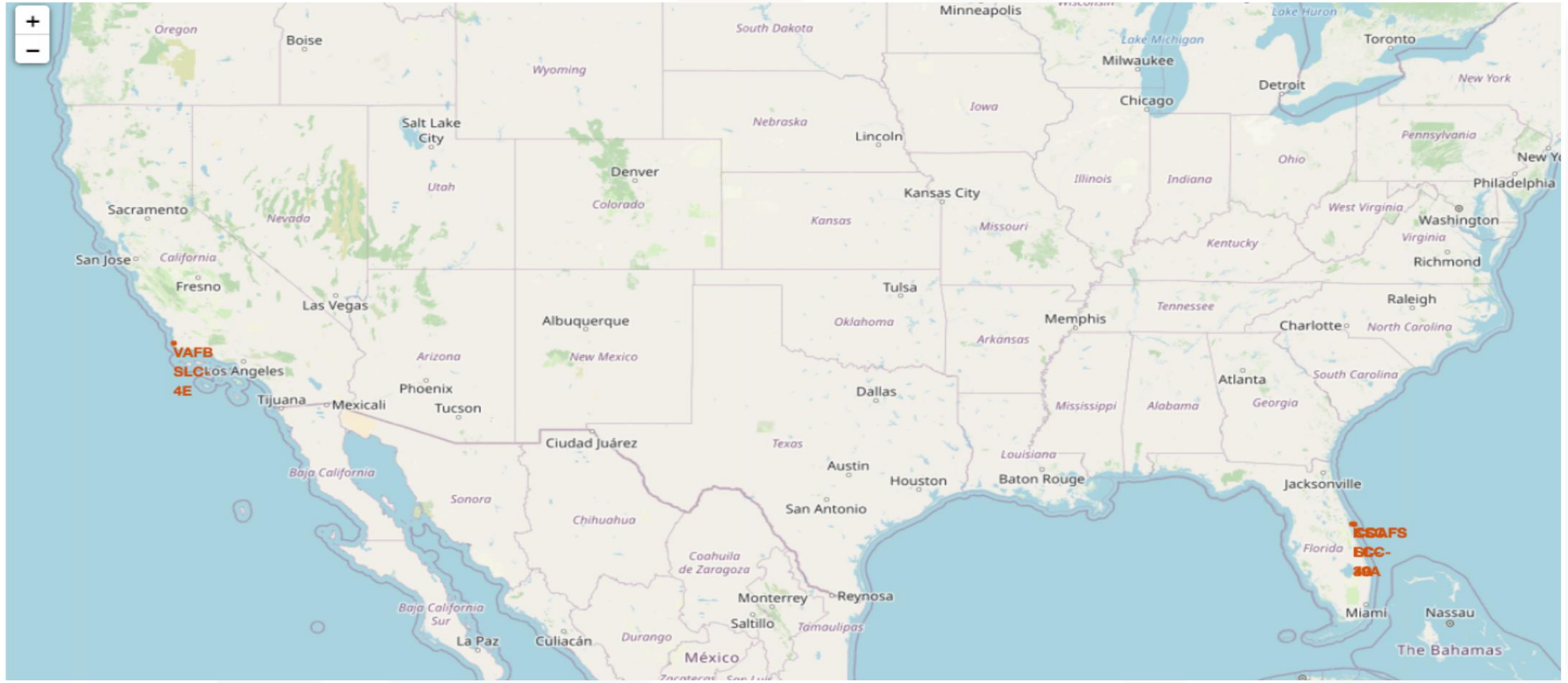
landing__outcome
No attempt
Success (ground pad)
Success (drone ship)
Success (drone ship)
Success (ground pad)
Failure (drone ship)
Success (drone ship)
Success (drone ship)
Success (drone ship)
Failure (drone ship)
Failure (drone ship)
Success (ground pad)
Precluded (drone ship)
No attempt
Failure (drone ship)
No attempt
Controlled (ocean)
Failure (drone ship)
Uncontrolled (ocean)
No attempt
No attempt
Controlled (ocean)
No attempt
No attempt
Uncontrolled (ocean)
No attempt
No attempt
No attempt
Failure (parachute)
Failure (parachute)

MAP AND DISTANCE WITH FOLIUM

MAP AND DISTANCE WITH FOLIUM

- **To visualize the Launch Data into a 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

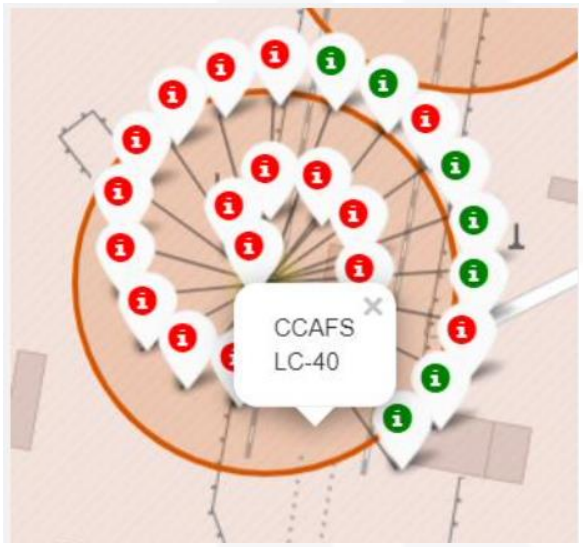
MAP AND DISTANCE WITH FOLIUM

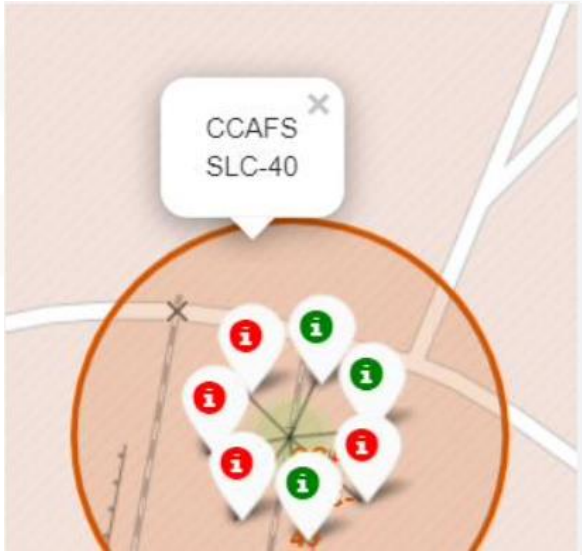


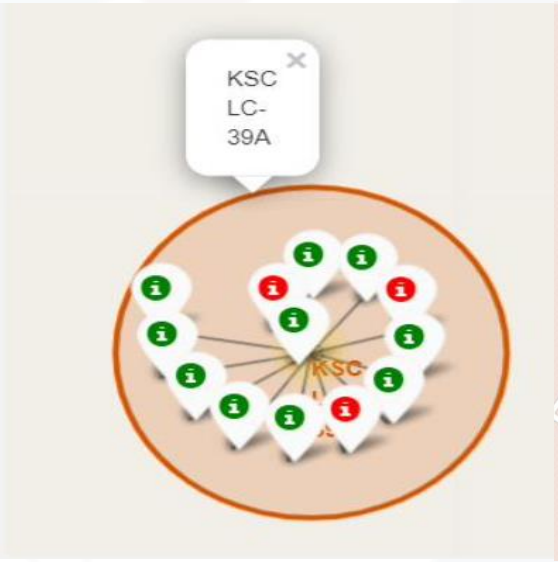
As you can see, *the SpaceX launch sites are in the* ***United States of America*** *states* ***Florida*** *and* ***California***

MAP AND DISTANCE WITH FOLIUM

- **Florida Launch site:**







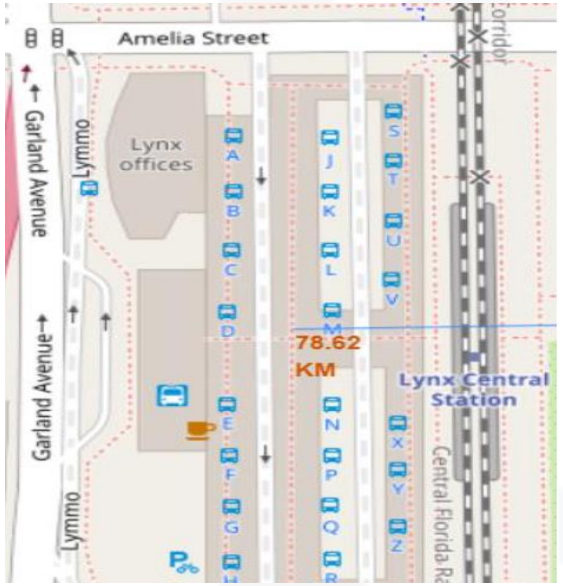
MAP AND DISTANCE WITH FOLIUM

- ***California Launch Site***

Green Marker: Succesfull launcher, Red Market: Unsuccesfull launcher

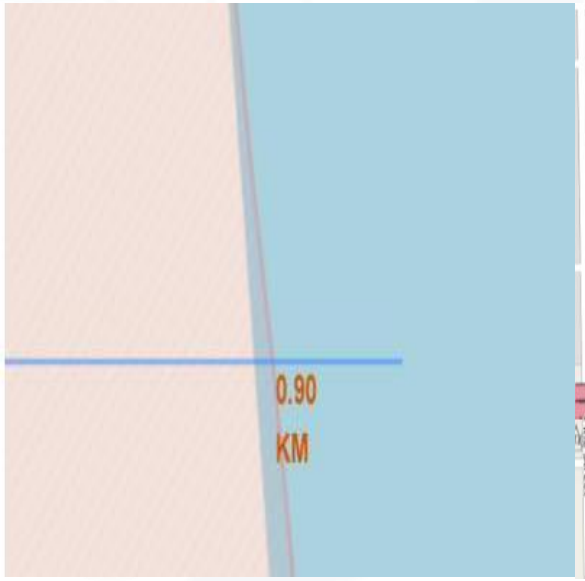
MAP AND DISTANCE WITH FOLIUM

- Launch Sites distance to landmarks to find trends with Haversine formula using CCAFS-SLC-40 as a reference



Distance to Railway Station

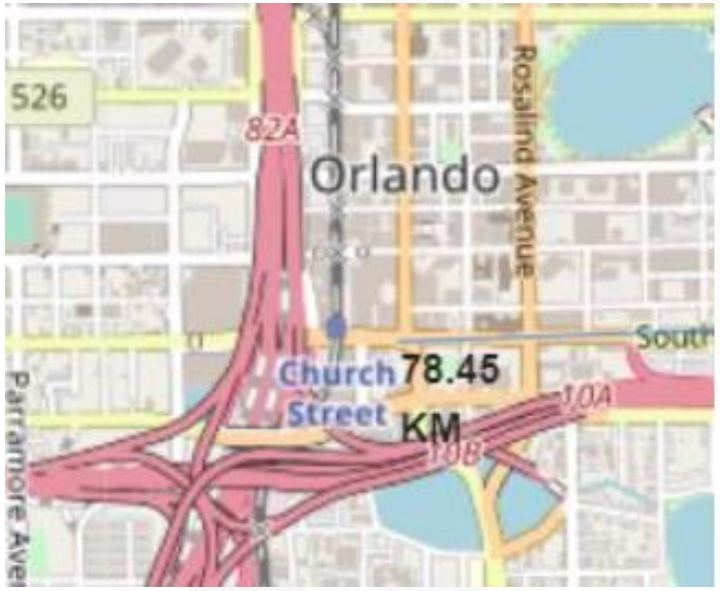
Distance to coast



Distance to Coastline

Distance to closest Highway

MAP AND DISTANCE WITH FOLIUM



Distance to City

- 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

DASHBOARD WITH PLOTTY

DASHBOARD WITH PLOTTY

The dashboard is built with Dash web framework

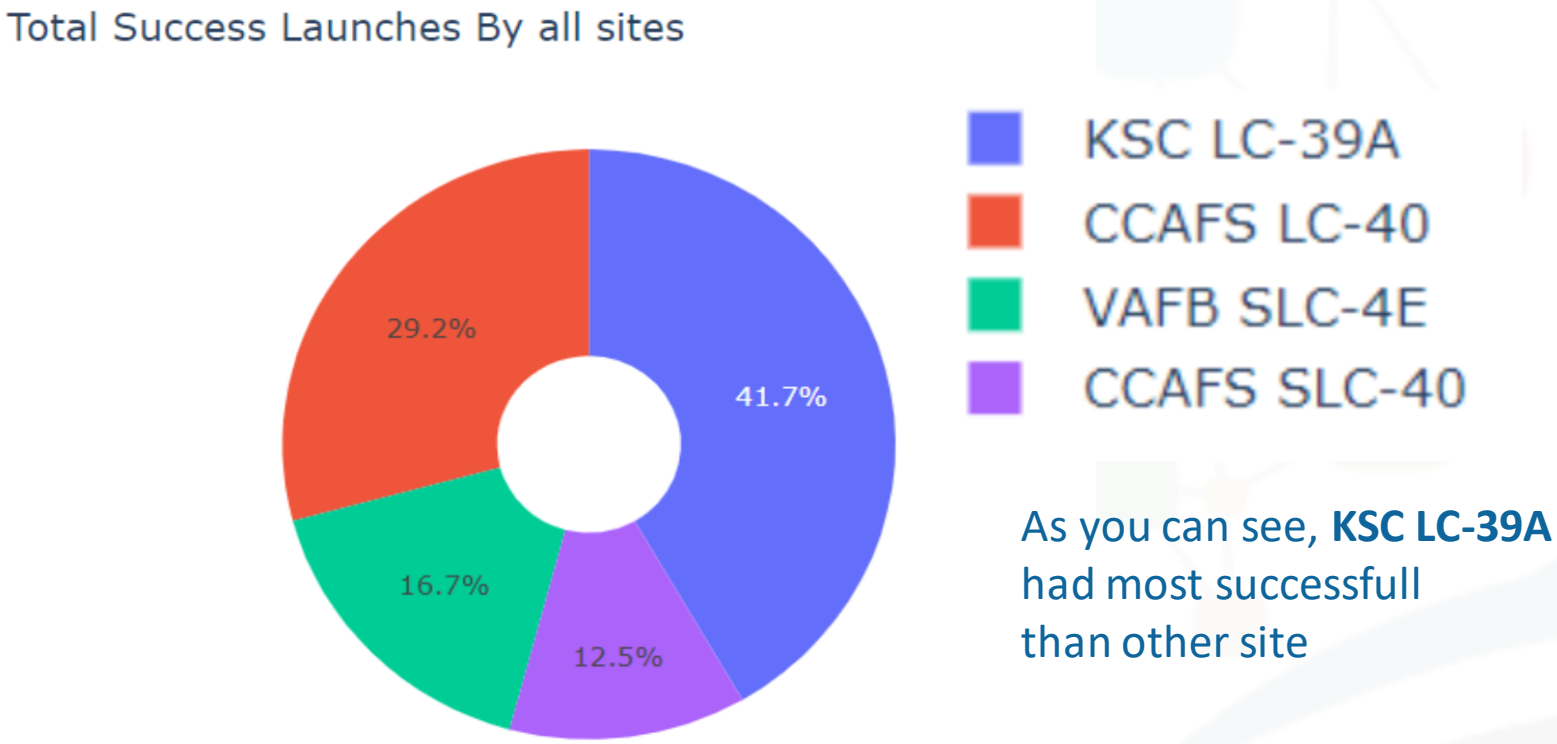
Graphs used:

Pie Chart
- showing the total launches by a certain site/all sites
- *size of the circle can be made proportional to the total quantity*

Scatter Graph:
- showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions
- shows the relationship between two variables.
- It is the best method to show you a non-linear pattern.
  The range of data flow, maximum and minimum value, can be determined.
- Observation and reading are straightforward.

DASHBOARD WITH PLOTTY

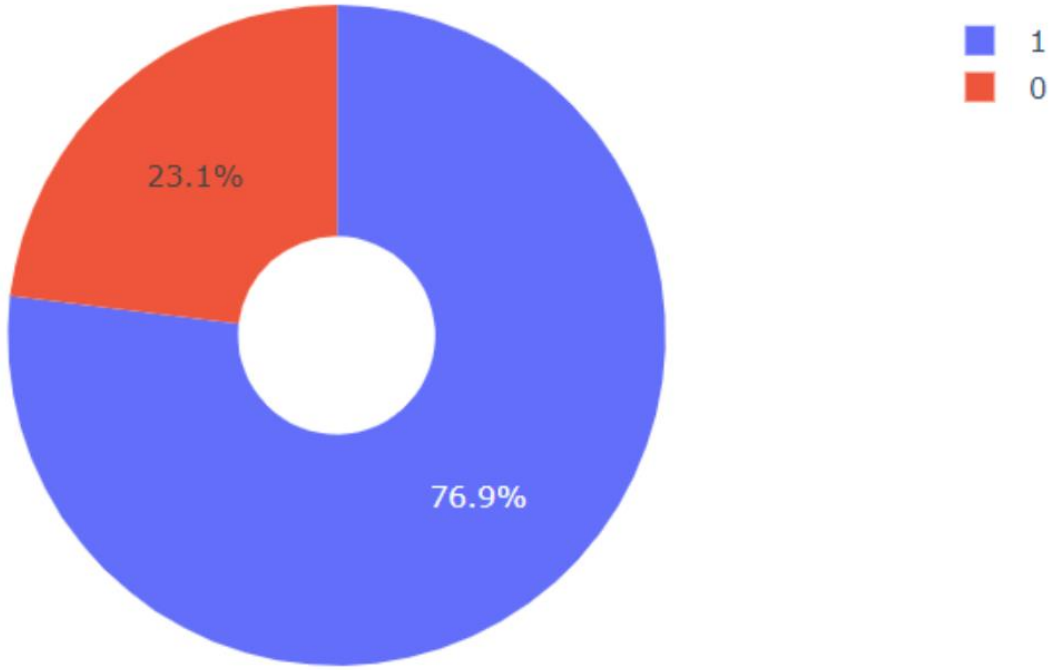
Pie chart that show the success percentage of each launch site:



As you can see, **KSC LC-39A** had most successfull than other site

DASHBOARD WITH PLOTTY

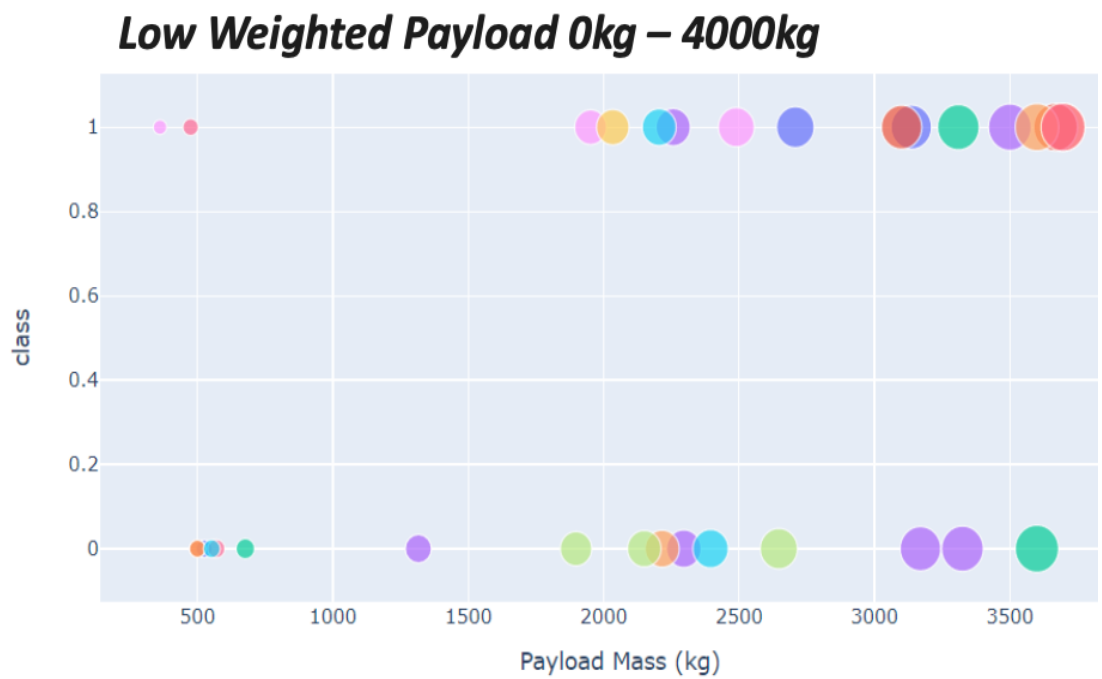
Pie chart for the launch site with highest launch success ratio:

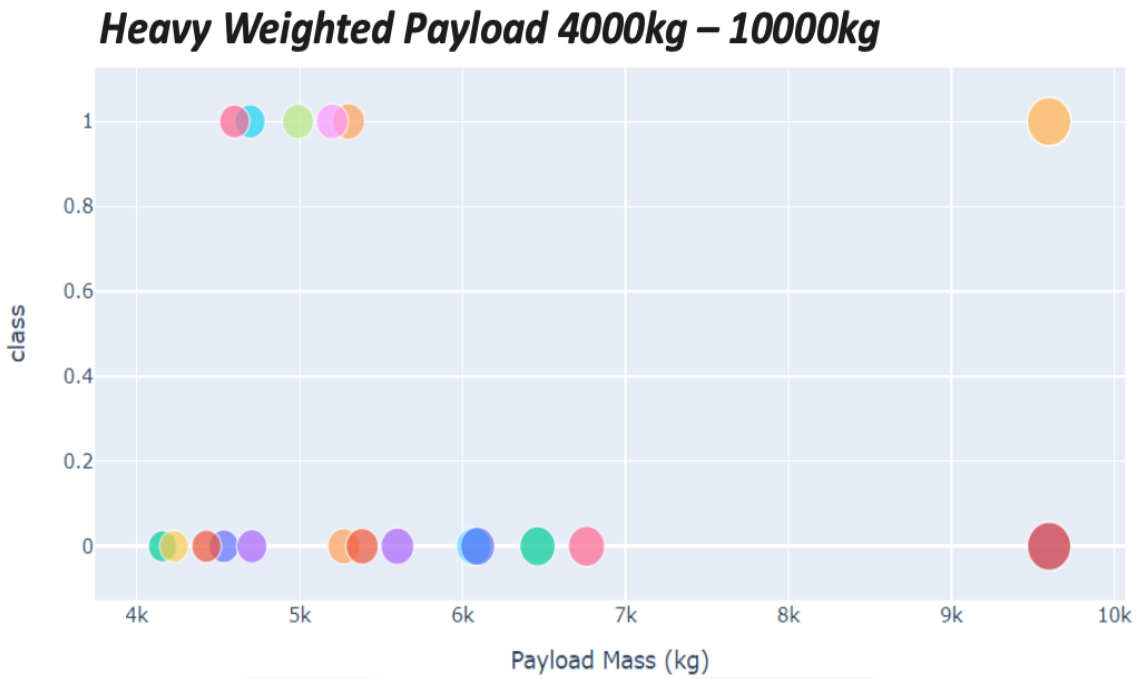


KSC LC-39A had a 76.9% success rate while getting a 23.1% unsuccess rate

DASHBOARD WITH PLOTTY

Payload vs. Launch Outcome scatter plot for all sites





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

MACHINE LEARNING PREDICTION

MACHINE LEARNING PREDICTION

Build a 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.

MACHINE LEARNING PREDICTION

Evaluate Model:

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

Find the best performance Classification Model:

- The model with the best accuracy score is the best performing model
- There is a final score of algorithms Classification in the end of machine learing analysis

MACHINE LEARNING PREDICTION

Classification Accuracy using training data

LogisticRegression accuracy**:**
(best parameters) {'C': 0.01, 'solver': 'lbfgs'}**.**
accuracy : 0.846428571428571**3**
SVM accuracy**:**
(best parameters) {'C': 1.0, 'gamma': 0.0316227766016837**9**, 'kernel': 'sigmoid'}
accuracy : 0.848214285714285**6**

DecisionTreeClassifier accuracy**:**
(best parameters) {'criterion': 'gini', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 1,
'min_samples_split': 10, 'splitter': 'random'}
accuracy : 0.889285714285714**2**

The winner algorithm using training data is: **TreeClassifier** with score **ab. 89%**

MACHINE LEARNING PREDICTION

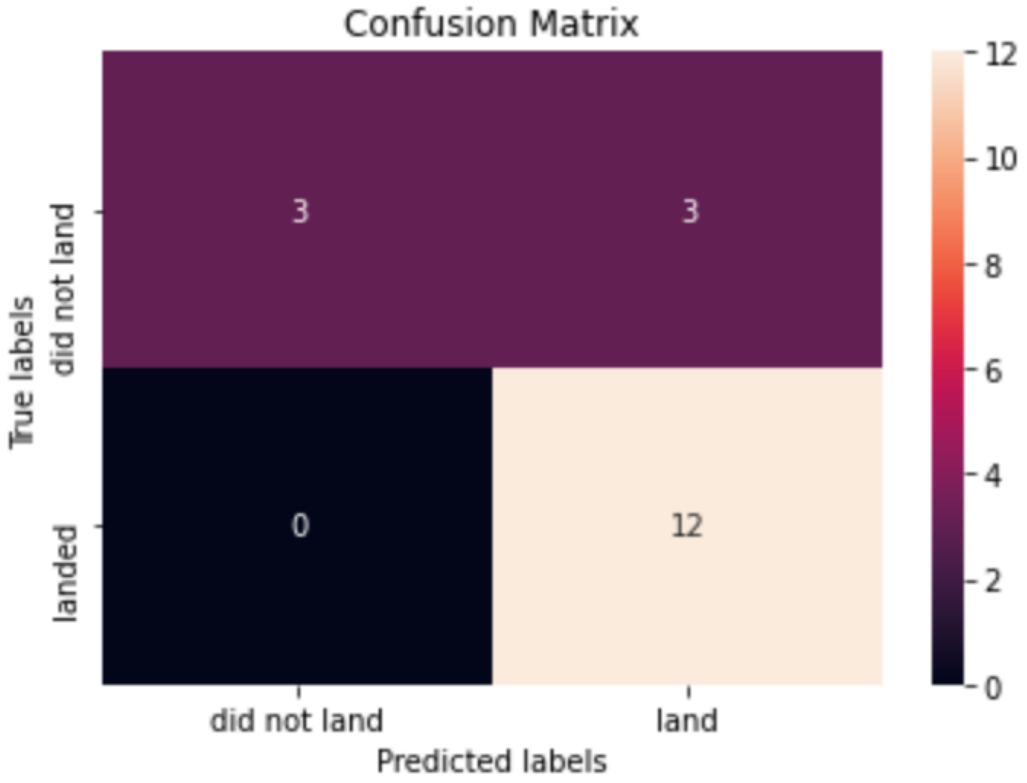
Classiflication Accuracy using Test data:

```
LinearRegression: 0.8333333333333334
SVM: 0.8333333333333334
DecisionTree: 0.9444444444444444
KNN: 0.8333333333333334
```

As you can see the algorithm that perfrom well is **TreeClassifier** with an accuracy of **94%**

MACHINE LEARNING PREDICTION

Confusion Matrix



We see that **Tree** can distinguish between the different classes. We see that the major problem is **false positives.**

CONCLUSION

- The Tree Classifier Algorithm is the best for Machine Learning in this dataset
- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional time in years
- 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

APPENDIX

- Software used during this analyis: IBM Watson studio from IBM Cloud
- Database used for SQL: IBM DB from IBM Cloud
- ADGGoogleMaps Module(Used with Folium Maps)