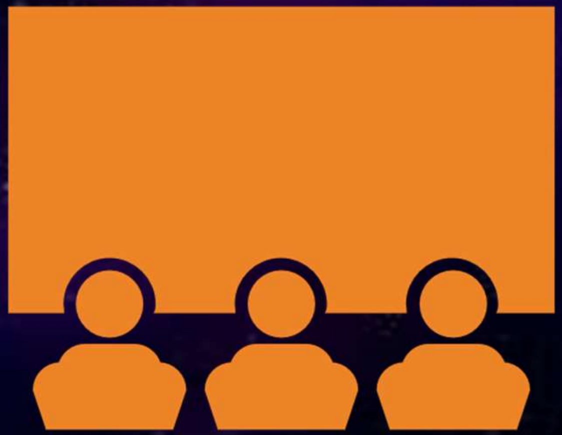
IBM Data Science Project

SPACE X

Carlos Augusto P. Costa

10-06-2022

OUTLINE



- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Conclusion

EXECUTIVE SUMMARY

- Data collection and data wrangling methodology related slides 9-13
- EDA and interactive visual analytics methodology related slides 14 - 15
- Predictive analysis methodology related slides 16-17
- EDA with visualization results slides 19-26
- EDA with SQL results slides 27-30
- Interactive map with Folium results slides -31
- Plotly Dash dashboard results slides 32
- Predictive analysis – Classification results slides 33-34
- Conclusion slide 35

EXECUTIVE SUMMARY

- Methodology
  - Collecting the Data
  - Preparing de Data
  - Analysing the Data
  - Use Machine Learning
- Results
  - Exploratory Results
  - Predictive analysis results

INTRODUCTION

The Objective of this Project is to predict if the Falcon 9 first stage will land successfully.

SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. So, 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.

METHODOLOGY

- Use webscraping and data analysis tools to load a dataset, clean it, and find out interesting insights from it.

➢ Collecting data on the Falcon 9 first-stage landings (using a RESTful API and web scraping), converting the data into a dataframe and then performing some data wrangling.

➢ Data wrangling (Transforming data for Machine Learning) ; One Hot Encoding data fields and dropping irrelevant columns

METHODOLOGY

- Use data visualization skills to visualize the data and extract meaningful patterns to guide the modeling process.

➢ Exploratory data analysis (EDA) using visualization and SQL - Plotting : Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data.

METHODOLOGY

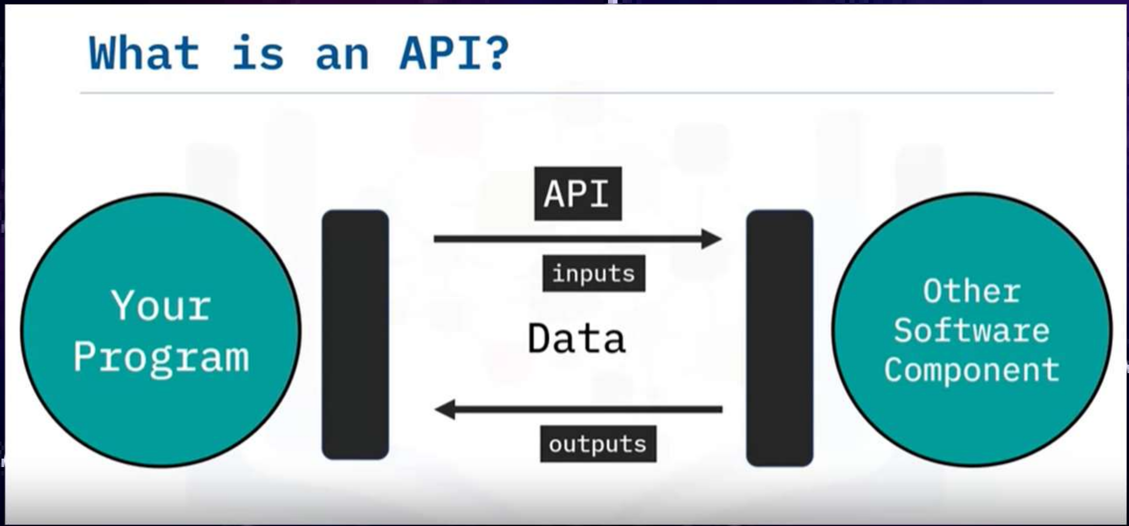
- Build a dashboard to analyze launch records interactively with Plotly Dash and an interactive map to analyze the launch site proximity with Folium.
- Use machine learning to determine if the first stage of Falcon 9 will land successfully.

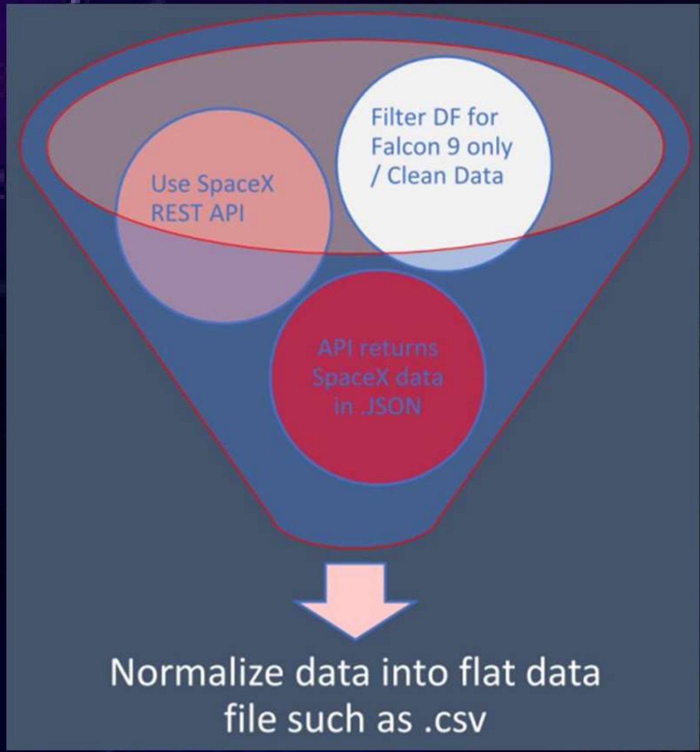
➢ Split the data into training data and test data in different classification methods (SVM, Classification Trees, and Logistic...) then create a grid search to find the best Hyperparameter for each algorithm..

➢ Perform predictive analysis using classification models

METHODOLOGY - Collecting the data

- Collecting the data from a URL using RESTful API





METHODOLOGY - Collecting the data

Now let's start requesting rocket launch data from SpaceX API with the following URL:

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

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

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Skil
```

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
# Use json_normalize meethod to convert the json result into a dataframe
import json
# Load data using Python JSON module
data = json.loads(requests.get(static_json_url).text)
# Flatten data
data = pd.json_normalize(data)
```

- Collecting the data from a URL using RESTful API

1. Get Response from a API
2. Convert response to a json file
3. Apply custom functions to clean data
4. Assign list to dictionary then to the dataframe
5. Filter dataframe and export to csv file

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

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

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

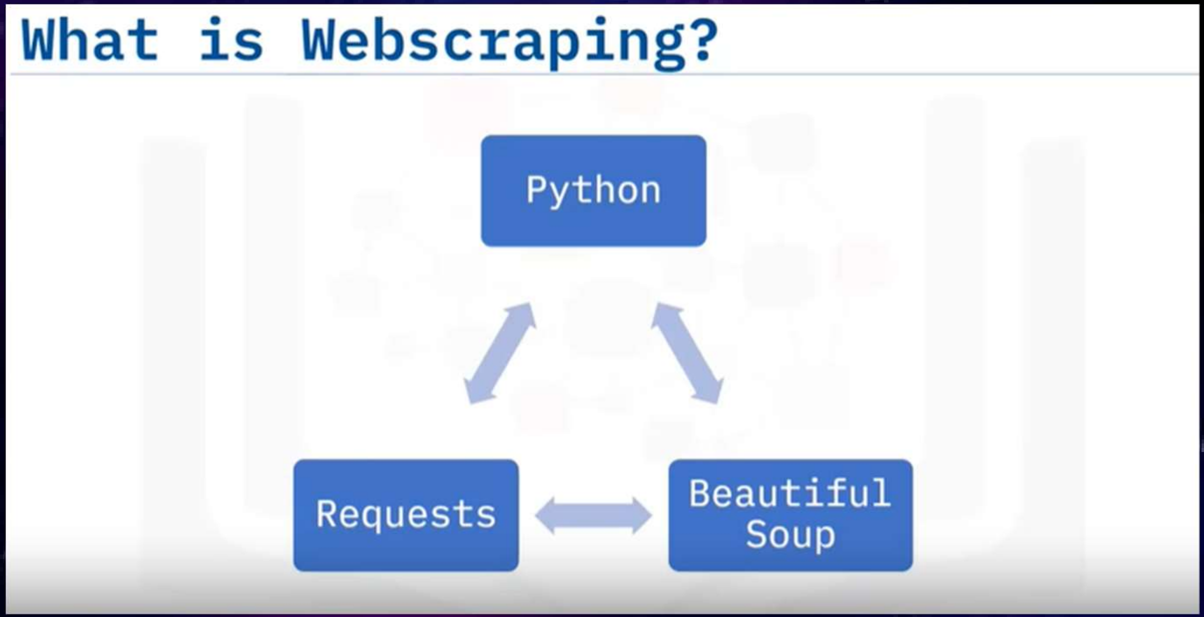
Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

```
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}
```

METHODOLOGY - Collecting the data

- Webscraping – using Beautifull Soup
- Historical launch records from a Wikipedia page

https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches



METHODOLOGY - Collecting the data

- Webscraping – using Beautifull Soup

1. Get response from HTML
2. Create BeautifulSoup Object
3. Find Table
4. Get Column names
5. Create dictionary
6. Append data to Keys
7. Convert dictionary to dataframe
8. Export Dataframe to csv file

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url
# assign the response to a object
response=requests.get(static_url)
```

Create a `BeautifulSoup` object from the HTML `response`

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

```
# 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')
```

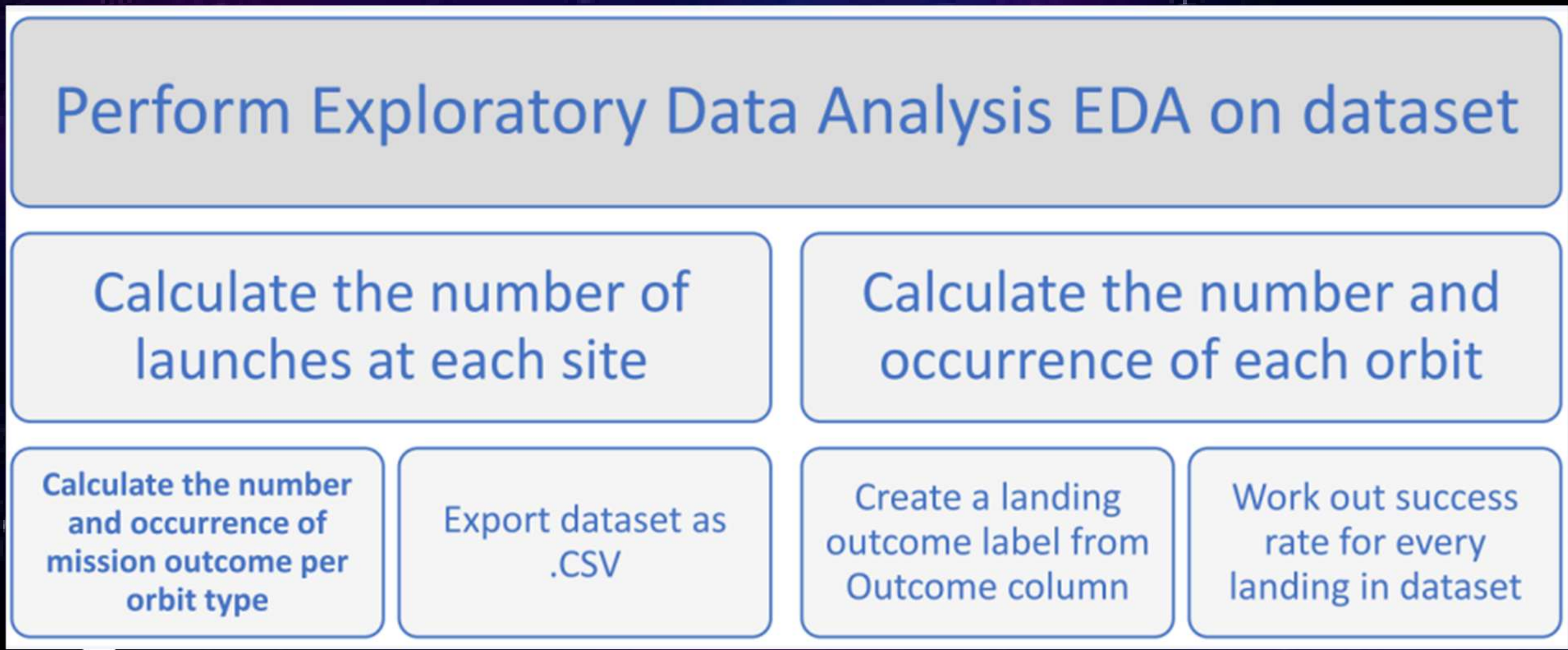
```
column_names = []
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

```
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']=[]
```

```
headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if  ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

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

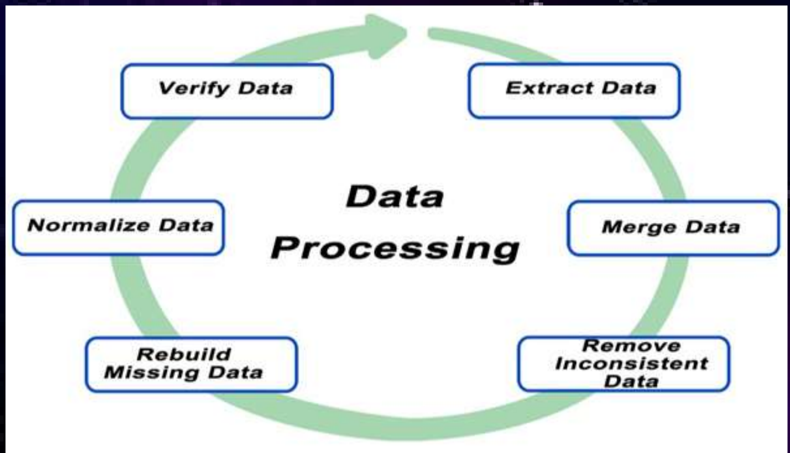
METHODOLOGY – Data Wrangling

- In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident. We mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.



METHODOLOGY – EDA

- Before analising we must clear and prepare the data (check inconsistencies, deal with null values, normalize…).
- The use of graphic visualization is the best way of understanding the data
- SQL queries are used to gather information about the dataset (filtering and grouping)

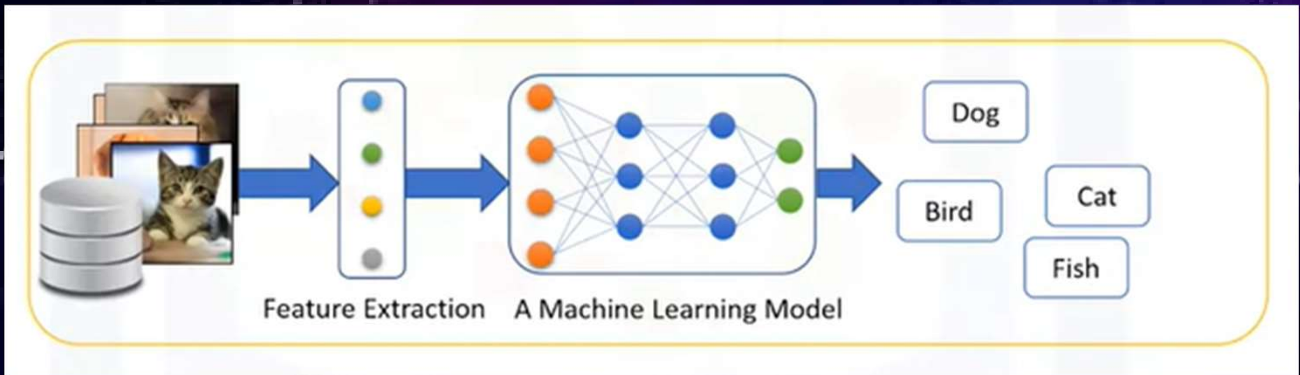


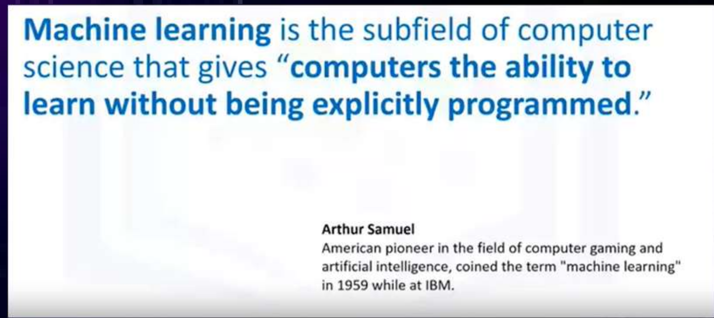
METHODOLOGY – Visualization with Folium

- To visualize the Launch Data into an interactive map. We took the Latitude and Longitude Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.
- We assigned the dataframe launch_outcomes(failures, successes) to classes 0 and 1 with Green and Red markers on the map in a MarkerCluster()
- Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. Lines are drawn on the map to measure distance to landmarks

METHODOLOGY – Machine Learning

- Machine Learning Pipeline





METHODOLOGY – Machine Learning

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

Preparing and understanding the data – EDA Results

- Cleaning, filtering and dealing with missing values

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

In [61]:
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
data_falcon9.head()
```

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

In [65]:
```
# Calculate the mean value of PayloadMass column
PLmass_mean = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].fillna(value=PLmass_mean, inplace=True)
data_falcon9.isnull().sum()
```

- Understanding the data – Calculating Launch Sites

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

In [5]:
```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

Out[5]:
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

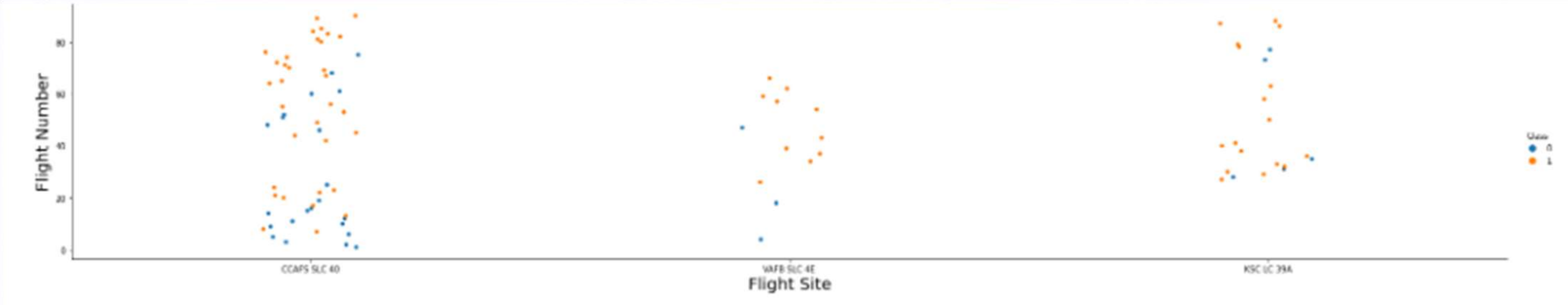
Understanding the data – EDA Results

- Understanding the data - Visualization of the Dataframe

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011

Understanding the data – Graphic Visualization

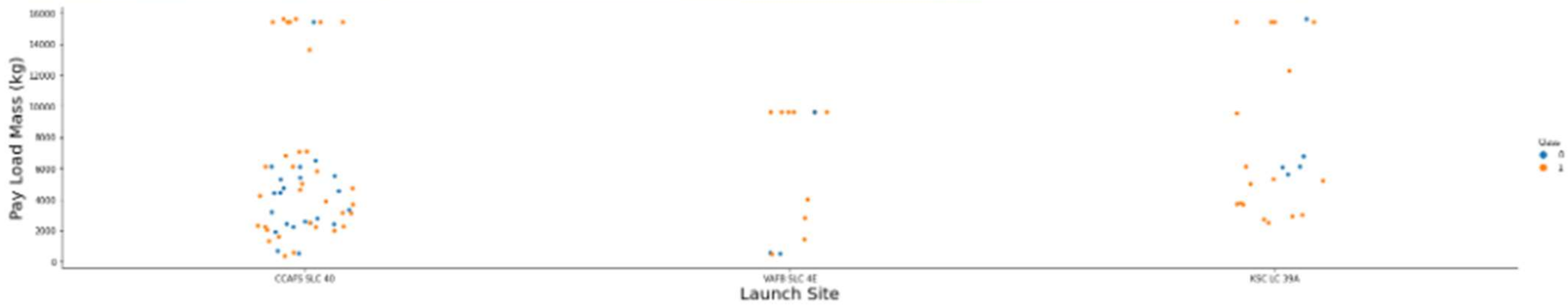
- Relationship between Flight Number and Lauch site



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

Understanding the data – Graphic Visualization

- Relationship between Pay Load and Lauch site

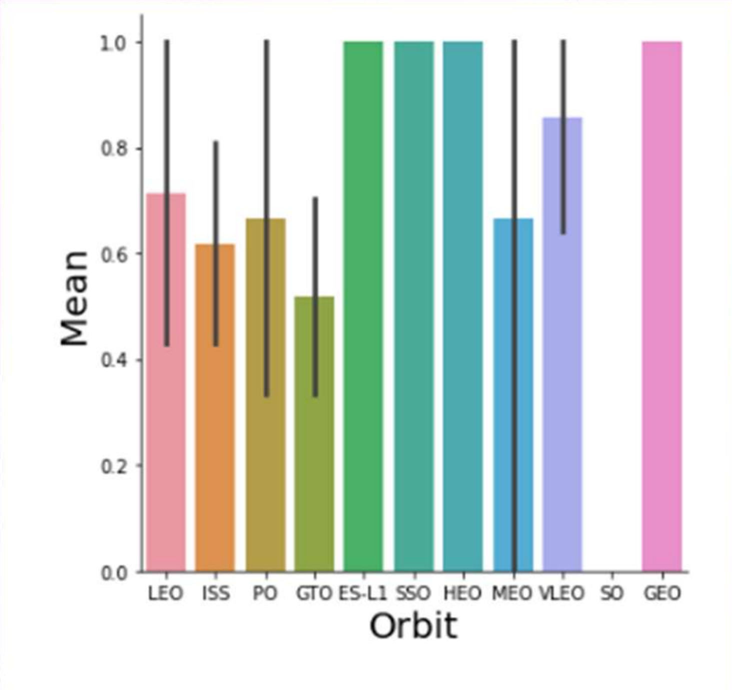


The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket.

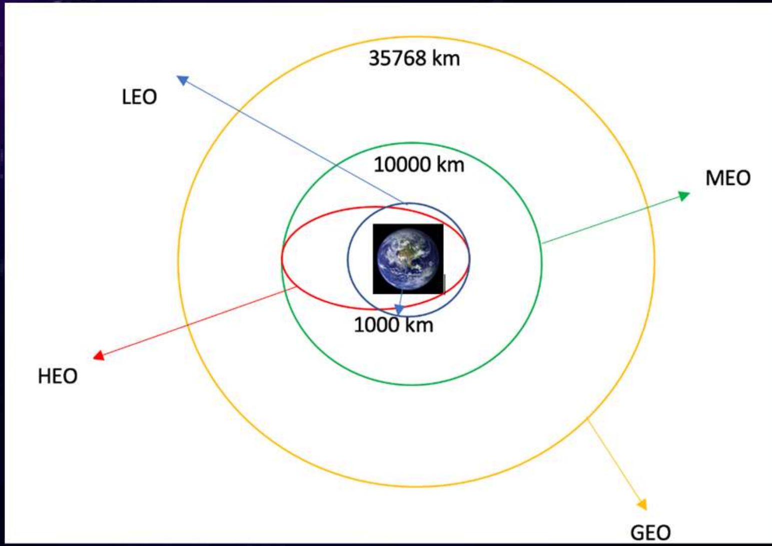
There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependant on Pay Load Mass for a success launch.

Understanding the data – Graphic Visualization

- Relationship between Success rate for each Orbit

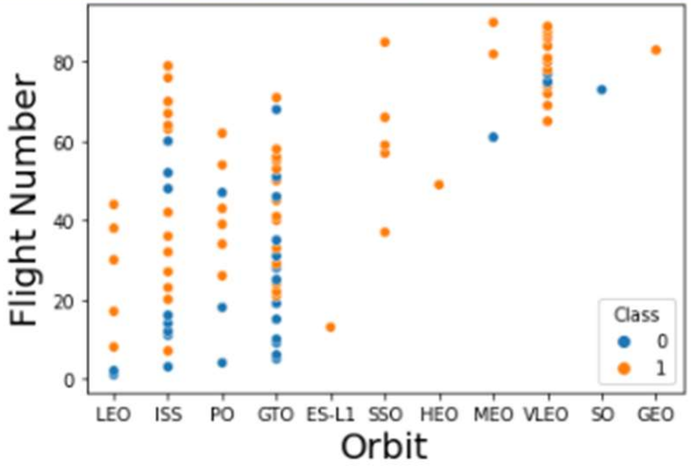


Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate



Understanding the data – Graphic Visualization

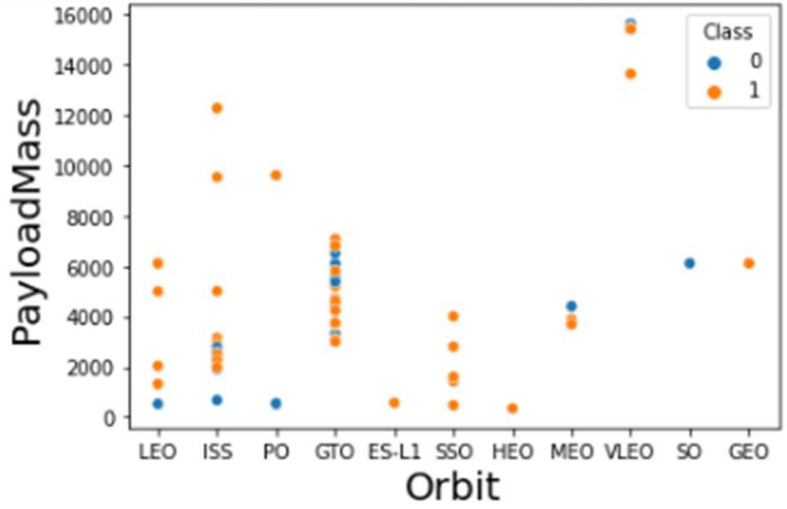
- Relationship between Flight number and Orbits



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.

Understanding the data – Graphic Visualization

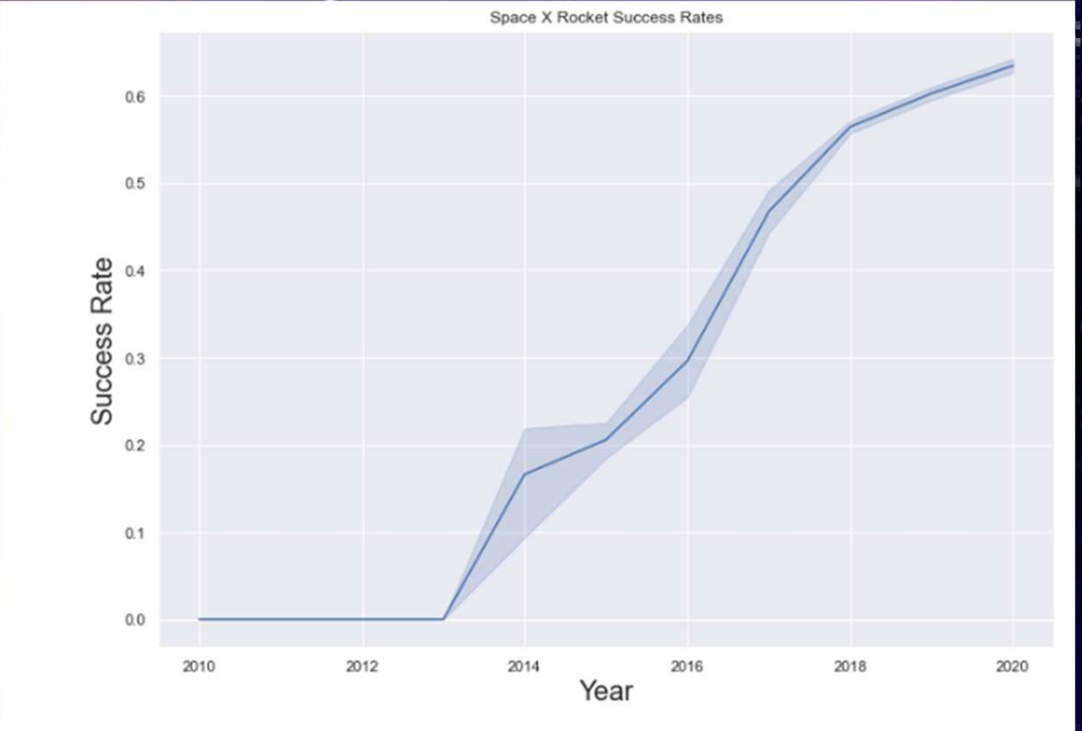
- Relationship between Pay Load and Orbits



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

Understanding the data – Graphic Visualization

- Lauch success Yearly trend



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

Understanding the data - EDA/SQL Results

Display the names of the unique launch sites in the space mission

In [7]:
```
%sql select DISTINCT(Launch_Site) from SPACEXTBL
```

```
 * sqlite:///my_data1.db
Done.
```

Out[7]:

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

Display 5 records where launch sites begin with the string 'CCA'

In [8]:
```
%sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

```
 * sqlite:///my_data1.db
Done.
```

Out[8]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Understanding the data - EDA/SQL Results

Display the total payload mass carried by boosters launched by NASA (CRS)

In [9]:
```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer=='NASA (CRS)'
```

```
 * sqlite:///my_data1.db
Done.
```

Out[9]:

sum(PAYLOAD_MASS__KG_)
45596

Display average payload mass carried by booster version F9 v1.1

[15]:
```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version='F9 v1.1'
```

```
 * sqlite:///my_data1.db
Done.
```

[15]:

avg(PAYLOAD_MASS__KG_)
2928.4

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

[14]:
```
%sql select min(Date) from SPACEXTBL where [Landing _Outcome] ='Success (ground pad)'
```

```
 * sqlite:///my_data1.db
Done.
```

[14]:

min(Date)
01-05-2017

Understanding the data - EDA/SQL Results

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
[12]: %sql select Booster_Version from SPACEXTBL where payload_mass__kg_ = (select Max(payload_mass__kg_) from SPACEXTBL)
 * sqlite:///my_data1.db
Done.
```

[12]:

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

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
In [10]: %sql select Booster_Version from SPACEXTBL where [Landing _Outcome]=='Success (drone ship)' and PAYLOAD_MASS
 * sqlite:///my_data1.db
Done.
```

Out[10]:

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

List the total number of successful and failure mission outcomes

```
[20]: %sql select  [Mission_Outcome],count (mission_outcome)  from SPACEXTBL group by [Mission_Outcome]
 * sqlite:///my_data1.db
Done.
```

[20]:

Mission_Outcome	count (mission_outcome)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Understanding the data - EDA/SQL Results

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

[21]:
```
%sql select  Booster_Version, Launch_Site, [Landing _Outcome],substr(Date, 4, 2) from SPACEXTBL where [Landing _Outcome]=='Failure (drone ship)' and sub
```

```
 * sqlite:///my_data1.db
Done.
```

[21]:

Booster_Version	Launch_Site	Landing _Outcome	substr(Date, 4, 2)
F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)	01
F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)	04

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

[24]:
```
%sql select  [Landing _outcome], Date, Launch_site, count([Landing _outcome]) as quant from SPACEXTBL group by [Lan
```

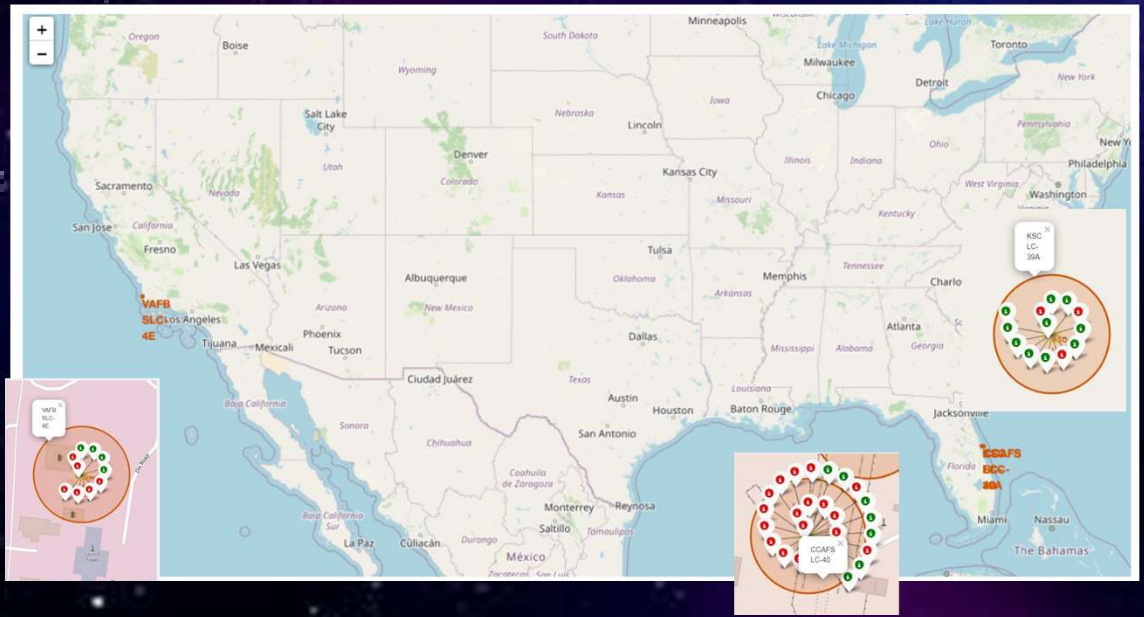
```
 * sqlite:///my_data1.db
Done.
```

[24]:

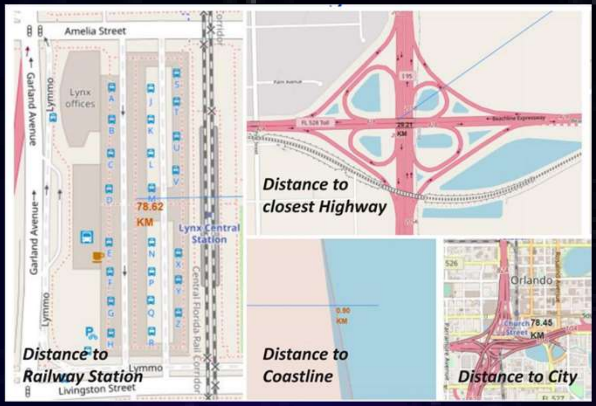
Landing _Outcome	Date	Launch_Site	quant
Success	22-07-2018	CCAFS SLC-40	38
No attempt	22-05-2012	CCAFS LC-40	21
Success (drone ship)	08-04-2016	CCAFS LC-40	14
Success (ground pad)	22-12-2015	CCAFS LC-40	9
Failure (drone ship)	10-01-2015	CCAFS LC-40	5
Controlled (ocean)	18-04-2014	CCAFS LC-40	5
Failure	05-12-2018	CCAFS SLC-40	3
Uncontrolled (ocean)	29-09-2013	VAFB SLC-4E	2
Failure (parachute)	04-06-2010	CCAFS LC-40	2
Precluded (drone ship)	28-06-2015	CCAFS LC-40	1
No attempt	06-08-2019	CCAFS SLC-40	1

Understanding the data – Folium Results

- Marking Lauch sites – Using the Latitude and Longitude Coordinates at each launch site, a Circle Marker was added around each with a label of the name and their success rate

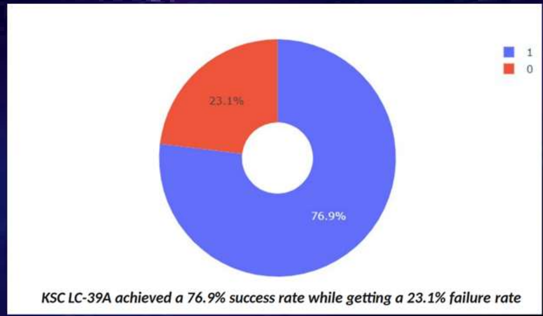


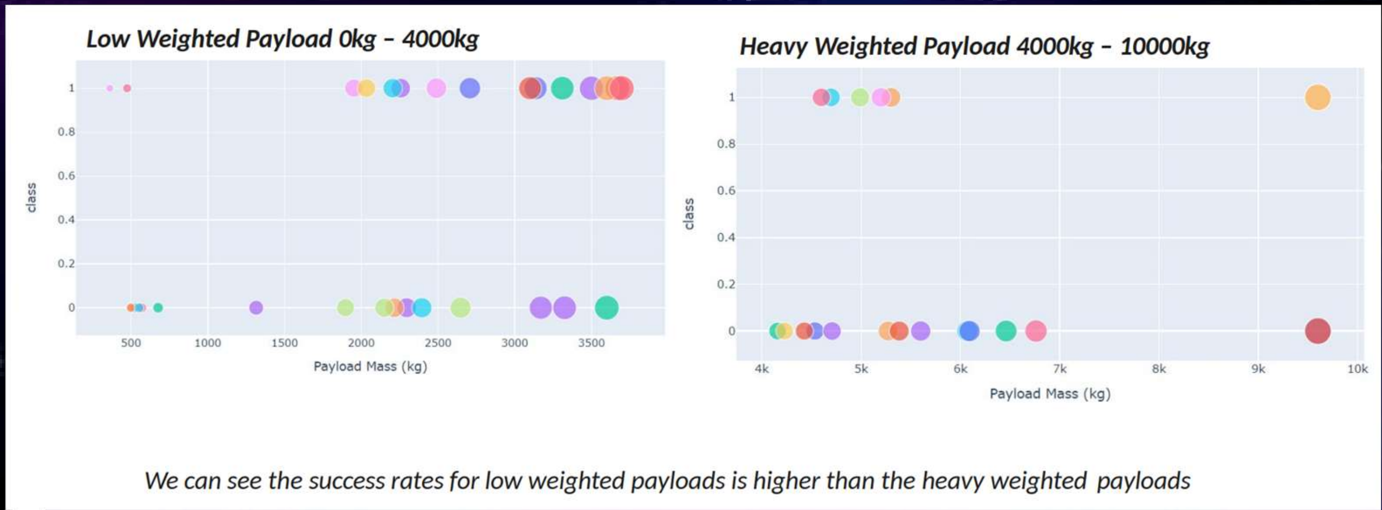
From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates.



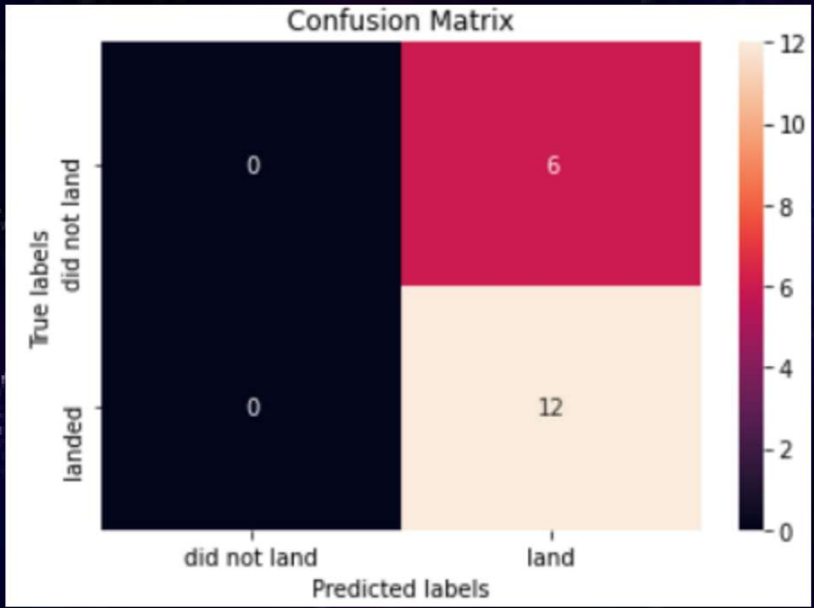
DASHBOARD

<[GitHub - Caugusto75/Applied_Data_Capstone: Final Project](#).>





Machine Learning Results



- The Tree can distinguish between different classes. The major problem is false positives.

- We had multiple runs with the train data for 03 algorithms (Logistic Regression., Tree Decision and KNN) to find the best hyperparameters. Their accuracy was then compared.

```
accuracy loreg: 0.8472222222222222
accuracy knn: 0.8472222222222222
accuracy tree: 0.8888888888888888
```

- After selecting the best classifier, we use the test data and also found the accuracy

```
0.8333333333333334
```

OVERALL FINDINGS & IMPLICATIONS

Findings

- CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.
- The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket.
- 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.
- Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.
- The success rate since 2013 kept increasing till 2020
- The tree Classifier Algorithm had the best predicti[illegible] result for this Data set

Implications

- There is not quite a clear pattern to be found using this visualization to decide if the Launch Site is dependent on Pay Load Mass for a success launch.
- The more amount of flights at a launch site the greater the success rate at a launch site.

CONCLUSION

- The best candidate Launch site would be KSC LC-19A as it has the most successful launches
- Orbits ES-L1, GEO, HEO, SSO and VLEO da the most success rate