IBM Data Science Capstone Project- Space X

Swati Narang

23rd June 2022

OUTLINE

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

Executive Summary

- ***Summary of methodologies***
  - Data Collection
  - Data Wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building and interactive map with folium
  - Building a dashboard with plotly Dash
  - Predictive analysis.

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

Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Common problems that needed solving.

What influences if the rocket will land successfully?

- The effect each relationship with certain rocket variables will impact in determining the success rate of a successful landing.
- What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate.

METHODOLOGY

Executive Summary

- Data collection methodology:
  - SpaceX Rest API
  - (Web Scrapping) from Wikipedia
- Perform data wrangling(Transforming data for Machine Learning)
  - one hot Encoding data fields for Machine Learning and dropping irrelevant columns
- Perform exploratory data analysis (EDA) using visualization and SQL

  -Plotting : Scatter Graphs, Bar Graphs to show relationships between variables to show 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

1.

DATA COLLECTION

The following data set was collected by :

1. We worked with SpaceX launch data that is gathered from the Space X REST API.

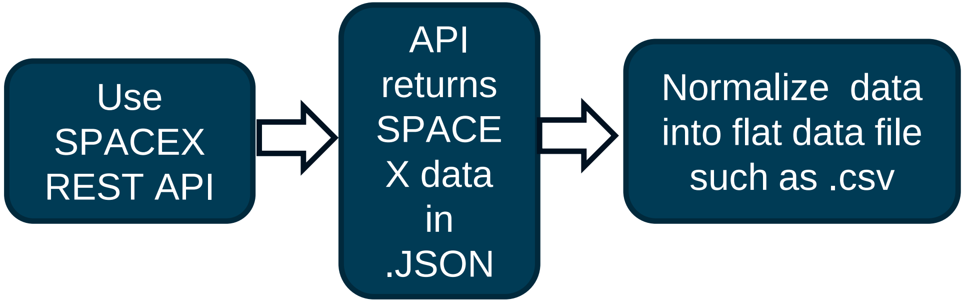
2. This API will give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.

3. Our goal is to use this data to predict whether SpaceX will attempt to land a rocket or not.

4. The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.

5. Another popular data source for obtaining Falcon 9 Launch data is web scraping Wikipedia using BeautifulSoup.

SPACEX API

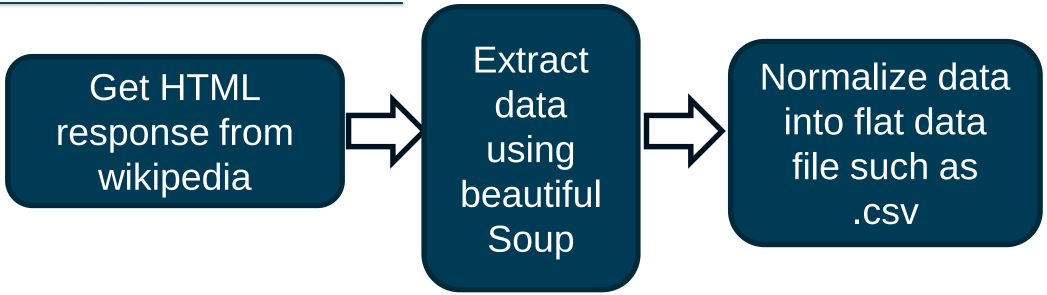


Use SPACEX REST API

API returns SPACE X data in .JSON

Normalize data into flat data file such as .csv

WEB SCRAPPING

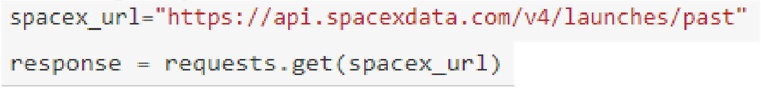


Get HTML response from wikipedia

Extract data using beautiful Soup

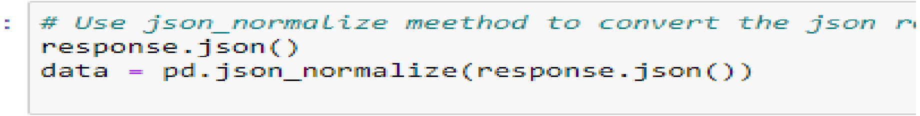
Normalize data into flat data file such as .csv

1 .Getting Response from API



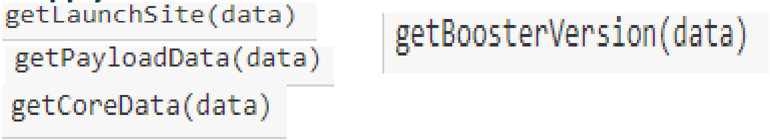
```
spacex_url="https://api.spacexdata.com/v4/launches/past"

response = requests.get(spacex_url)
```

2. Converting Response to a .JSON file:



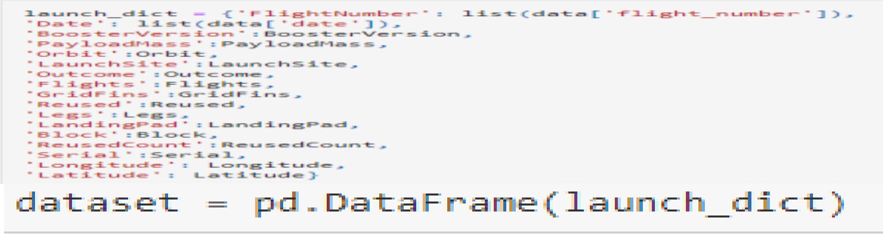
```
# Use json_normalize meethod to convert the json r
response.json()
data = pd.json_normalize(response.json())
```

3. Apply Custom function to clean data:



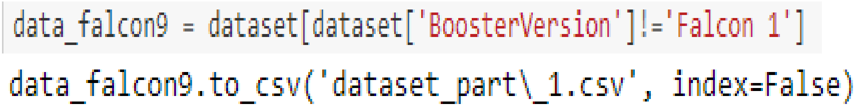
```
getLaunchSite(data)

getPayloadData(data)

getCoreData(data)

getBoosterVersion(data)
```

4. Assign list to dictionary then DataFrames:



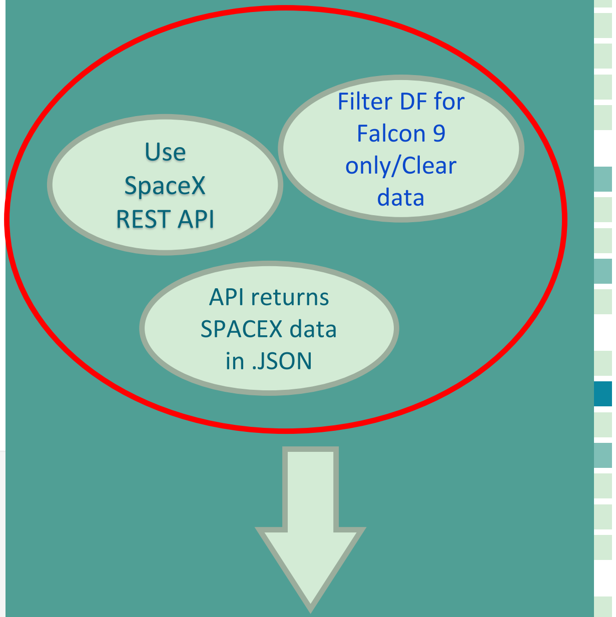
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,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

dataset = pd.DataFrame(launch_dict)
```

5. Filter Dataframe and export to flat file(.csv):



```
data_falcon9 = dataset[dataset['BoosterVersion']!='Falcon 1']

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

DATA COLLECTION –SPACEX API



Use SpaceX REST API

API returns SPACEX data in .JSON

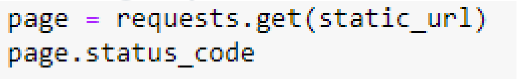
Filter DF for Falcon 9 only/Clear data

Normalize data into flat data file such as .csv

Github URL to Notebook

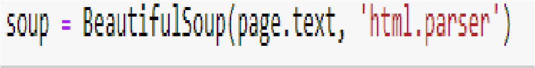
DATA COLLECTION –WEB SCRAPPING

1 .Getting Response from HTML



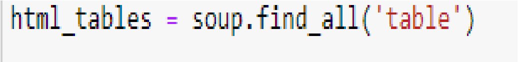
```
page = requests.get(static_url)
page.status_code
```

2. Creating Beautiful Soup Object:



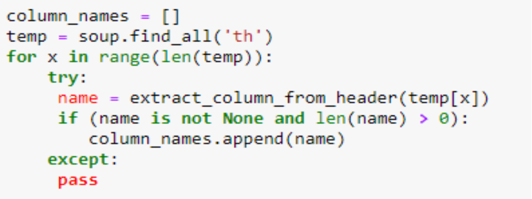
```
soup = BeautifulSoup(page.text, 'html.parser')
```

3. Finding tables:



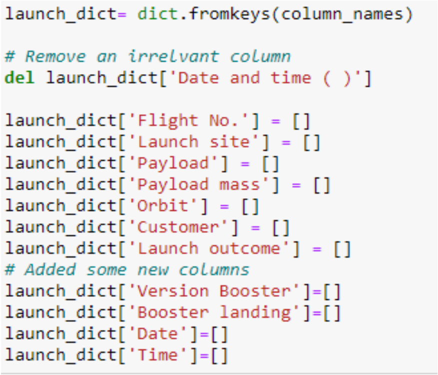
```
html_tables = soup.find_all('table')
```

4. Getting Column name:



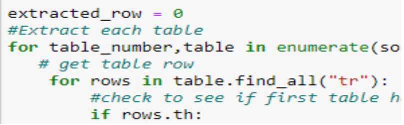
```
column_names = []
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

5 . Creation of Dictionary:



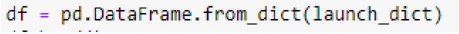
```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant 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'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

5. Appending data to keys(refer) to notebook block 13



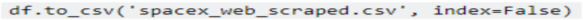
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(so
   # get table row
    for rows in table.find_all("tr"):
       #check to see if first table h
       if rows.th:
```

6.Converting dict to dataframes:

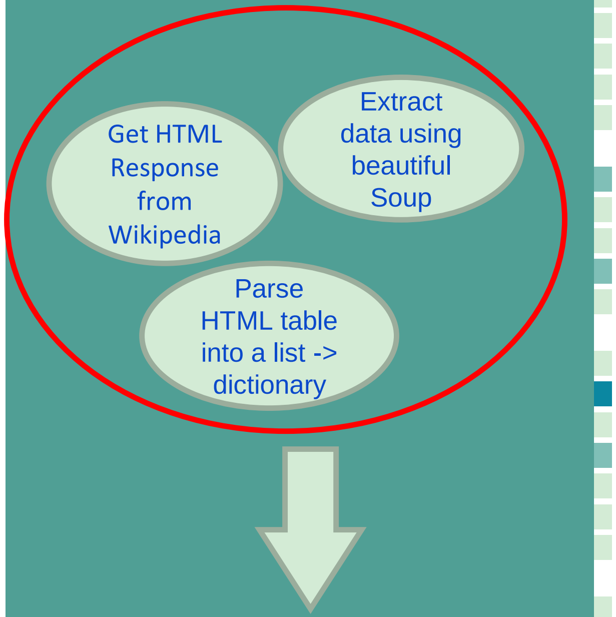


```
df = pd.DataFrame.from_dict(launch_dict)
```

7.Dataframe to .cssv



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



Get HTML Response from Wikipedia

Extract data using beautiful Soup

Parse HTML table into a list -> dictionary

Normalize data into flat data file such as .csv

Github URL to Notebook

2.

DATA WRANGLING

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 with 1 means the booster successfully landed 0 means it was successful.

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

[Github URL to Notebook]

Each launch aims to an dedicate orbit and here are some common orbit types.

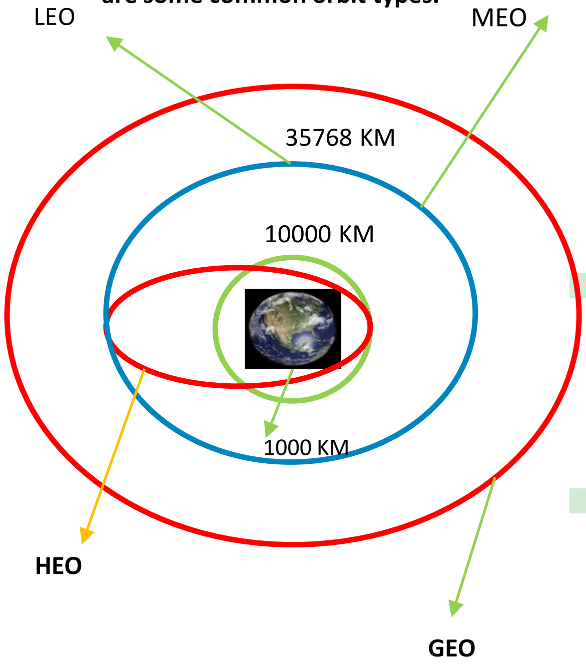


Diagram showing common orbit types SpaceX uses.

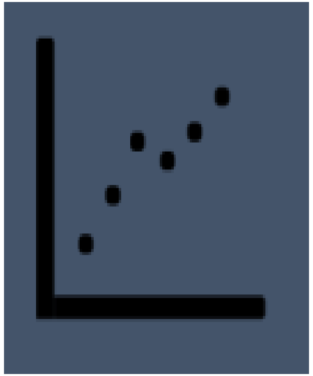
3.

EDA with Data Visualization

Scatter Graph 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

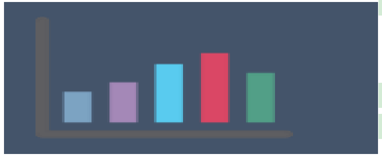


Scatter plot show how much one variable is affected by another. The relationship between two variables is called their correlation . Scatter plots usually consists of a large body of data.

Github URL to Notebook

Bar Graph being drawn :

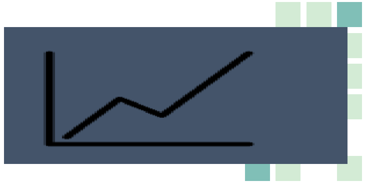
Mean VS. Orbit



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

Line Graph being Drawn:

Success Rate VS. Year



Line graph 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

4.

EDA with SQL

Performed SQL queries to gather information about the dataset.

For example of some questions we were asked about the data we needed information about. Which we are using SQL queries to get the answers in the dataset :

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC’
- Displaying 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.

Github URL to Notebook

5.

Interactive Map 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

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

Github URL to Notebook

6.

Dashboard with Plotly Dash.

Used Python Anywhere to host the website live 24/7 so your can play around with the data and view the data

-The dashboard is built with Dash web framework.

- ***Graphs***
- - ***Pie Chart showing the total launches by a certain site/all sites*** URL LINK TO WEB SITE
-
- - display relative proportions of multiple classes of data.
-
- - size of the circle can be made proportional to the total quantity it represents.

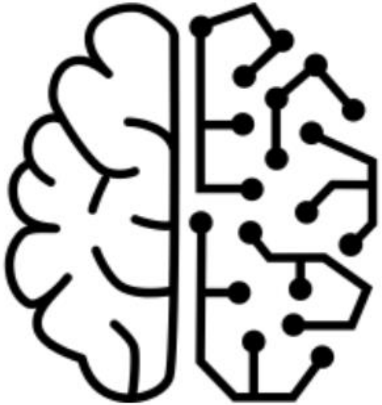
- ***Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions***
- - It shows the relationship between two variables.
- - It is the best method to show you a non-linear pattern.
- - The range of data flow, i.e. maximum and minimum value, can be determined.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

7.

Predictive Analysis.

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.

GITHUB URL to Notebook

RESULTS

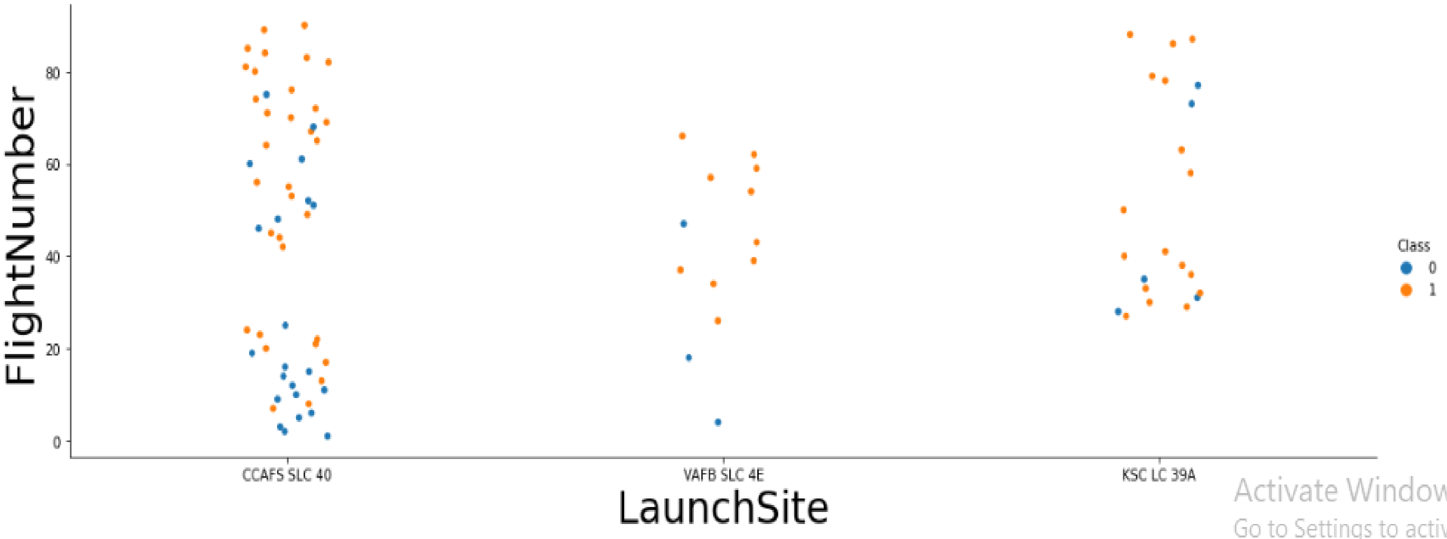
1. Exploratory data analysis results

2.Interactive analytics demo in screenshots.

3. Predictive analysis results

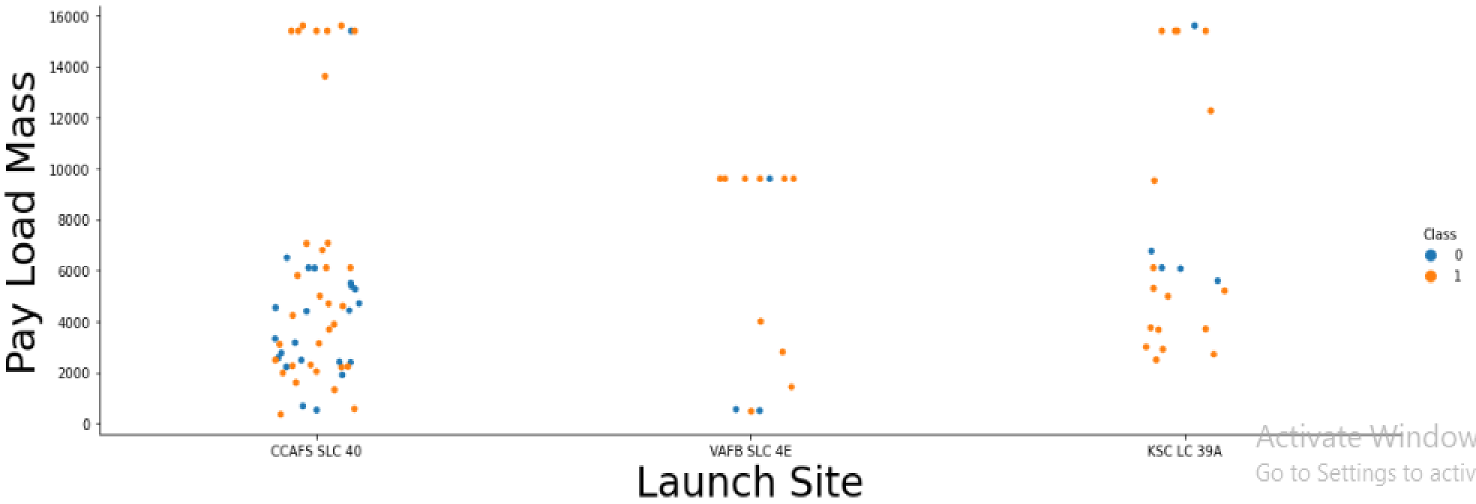
Insights drawn from EDA

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

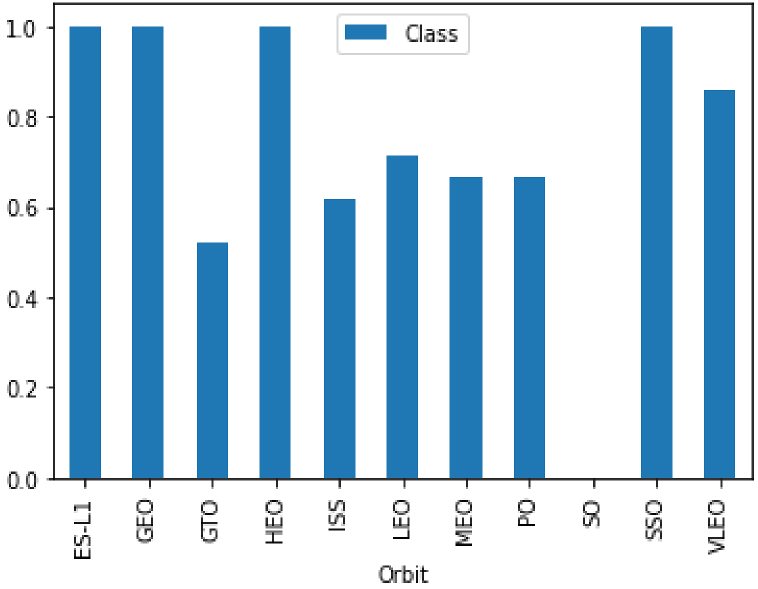


Load Mass for The greater the payload mass for Launch Site CCAFS SLC 40 the higher success rate for the Rocket. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependent on Pay a success launch.

Success Rate vs. Orbit Type

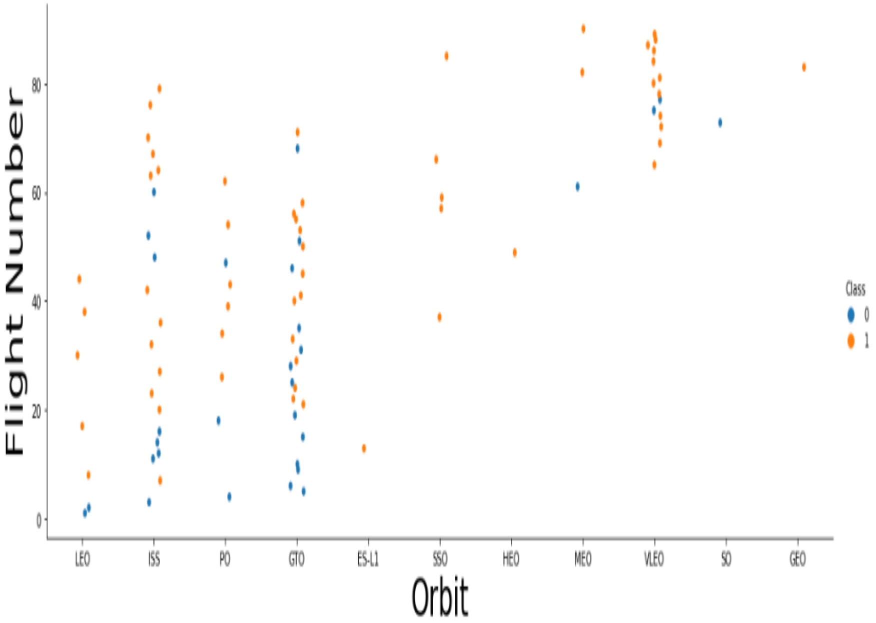
Orbit GEO,HEO,SSO,ES-L1 has the best

Success rate



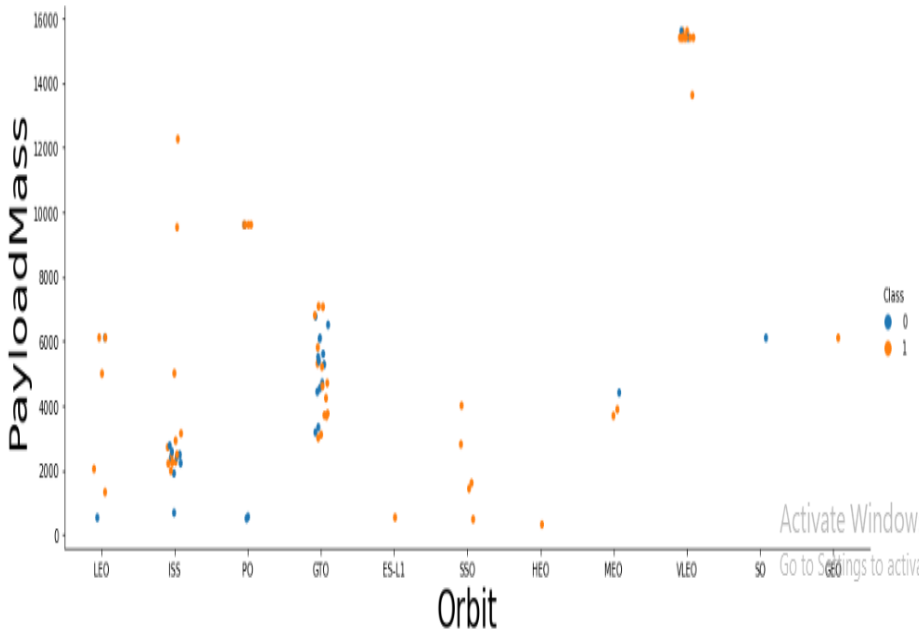
Flight Number vs. Orbit Type

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.



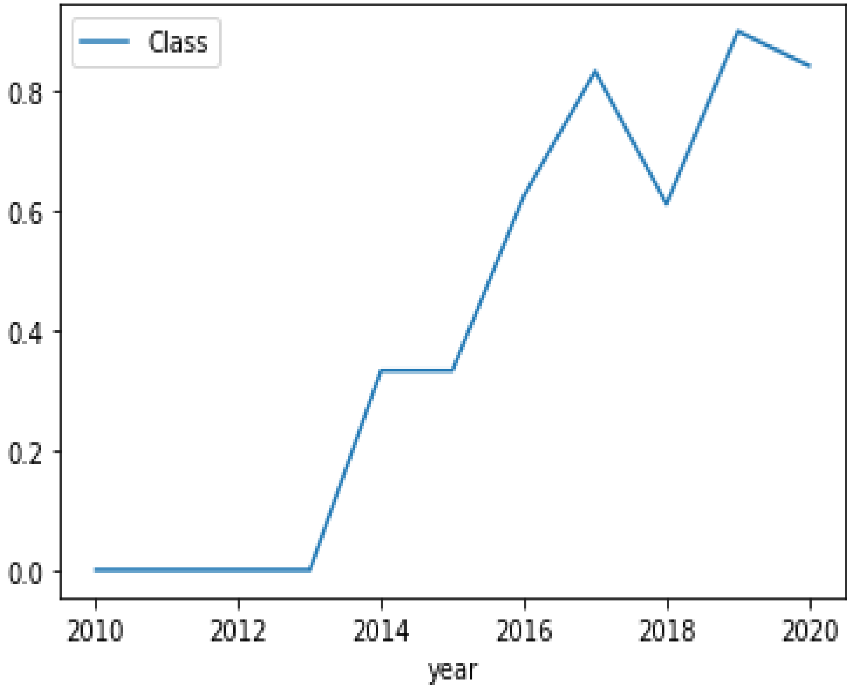
Payload vs. Orbit Type

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.

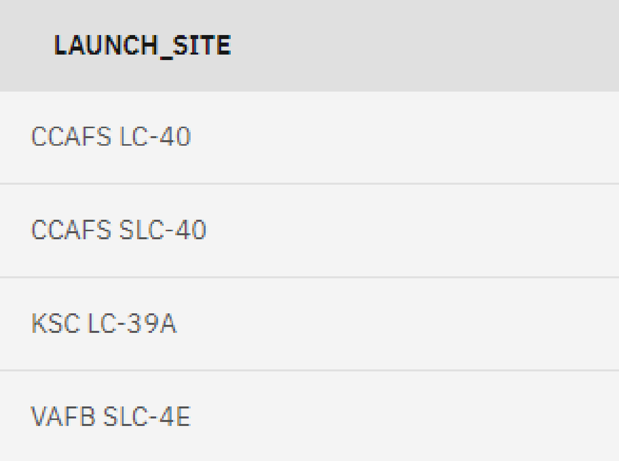


All Launch Site Names

SQL QUERY

Select Distinct launch_site from

SPACEXTBL



LAUNCH_SITE
CCAFS LC-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 ***SPACEXTBL***

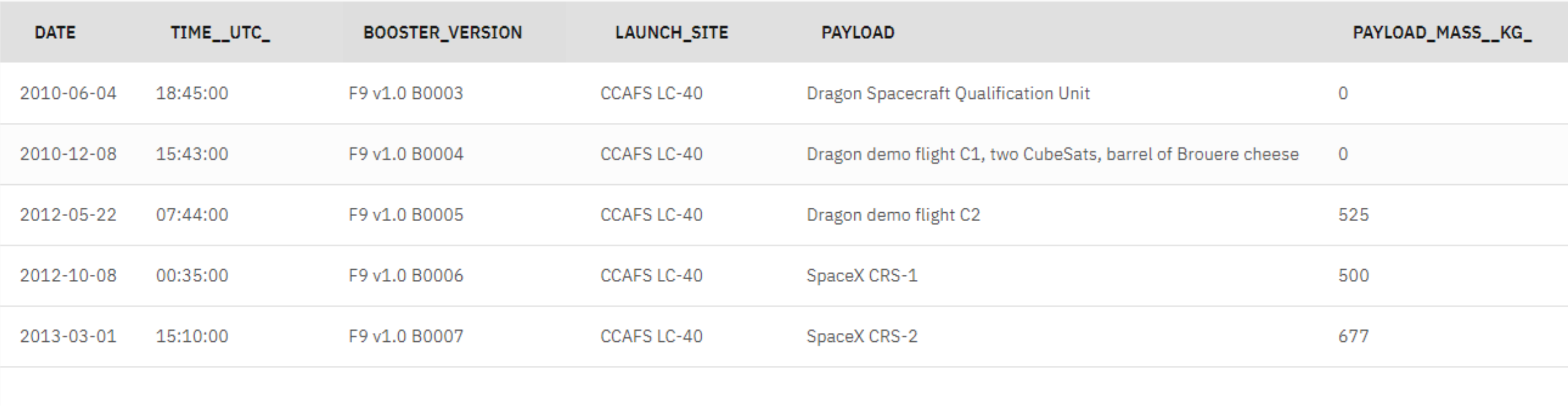
Launch Site Names Begin with 'CCA'

SQL QUERY

Select * from SPACEXTBL where Launch_site like ‘CCA%’ LIMIT 5

QUERY EXPLANATION

Using the word **LIMIT** 5 in the query means that it will show 5 records from **SPACEXTBL** and **LIKE** keyword has wild card with the words ‘CCA%’ the percentage in the end suggest that LAUNCH_SITE name must start with CCA.

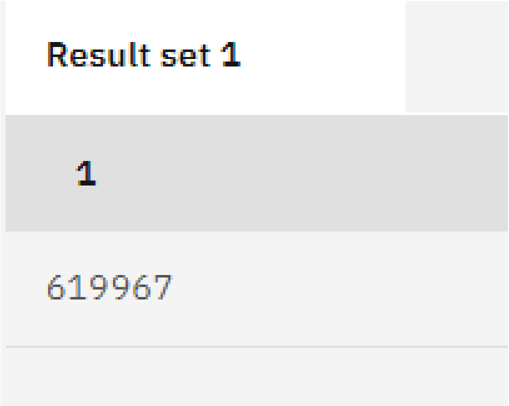


DATE	TIME__UTC_	BOOSTER_VERSION	LAUNCH_SITE	PAYLOAD	PAYLOAD_MASS__KG_
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677

Total Payload Mass

SQL QUERY

Select sum(PAYLOAD_MASS__KG) from SPACEXTBL



Result set 1

1
619967

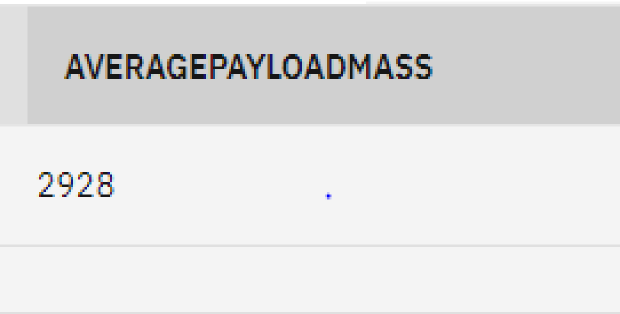
QUERY EXPLANATION

Using the function ***SUM*** summates the total in the column ***PAYLOAD_MASS__KG***.

Average Payload Mass by F9 v1.1

SQL QUERY

Select avg(PAYLOAD_MASS__KG_) as averagepayloadmass from SPACEXTBL where booster_version = ‘F9 v1.1’



AVERAGEPAYLOADMASS
2928

QUERY EXPLANATION

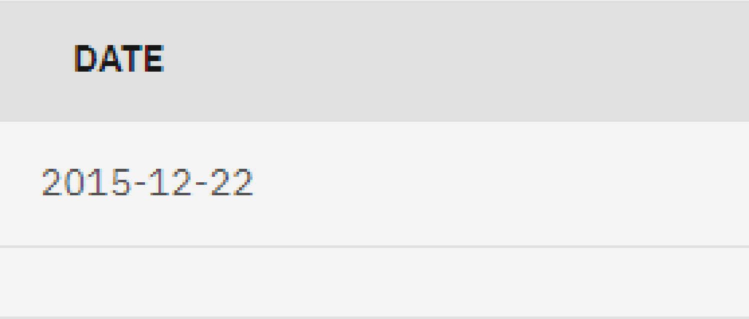
Using the function ***AVG*** works out the average in the column ***PAYLOAD_MASS__KG***.

The ***WHERE*** clause filters the data set to only perform calculations on ***Booster_version F9 v1.1***

First Successful Ground Landing Date

SQL QUERY

Select MIN(Date) as date from SPACEXTBL where landing__outcome like ‘%ground%’



DATE
2015-12-22

QUERY EXPLANATION

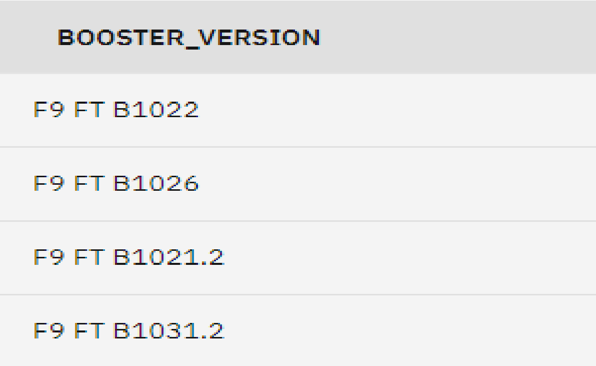
Using the function ***MIN*** works out the Minimum date in the column ***Date.***

The ***WHERE*** clause filters the data set to only perform calculations on ***Landing__outcome Success(ground pad).***

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL QUERY

Select Booster_version from SPACEXTBL where Landin__outcome = ‘Success (drone ship)’ and PAYLOAD_MASS__KG_ between 4000 and 6000



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

QUERY EXPLANATION

Selection only ***Booster_Version***

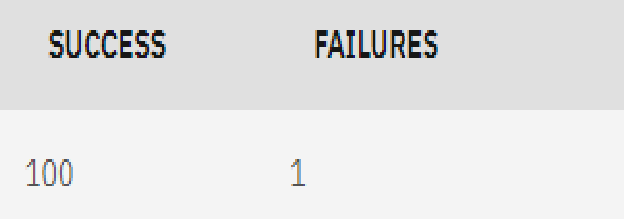
The ***WHERE*** clause filters the dataset to ***Landing__outcome = Success(drone ship).***

The ***AND*** clause specifies additional filter conditions ***PAYLOAD_MASS__KG_.***

Total Number of Successful and Failure Mission Outcomes

SQL QUERY

```
Select (select count(*) from SPACEXTBL
where MISSION_OUTCOME like
‘%Success%’) as Success, (select count(*)
from SPACEXTBL where
MISSION_OUTCOME like ‘%Failure%’)  as
Failure) from SPACEXTBL
```



SUCCESS	FAILURES
100	1

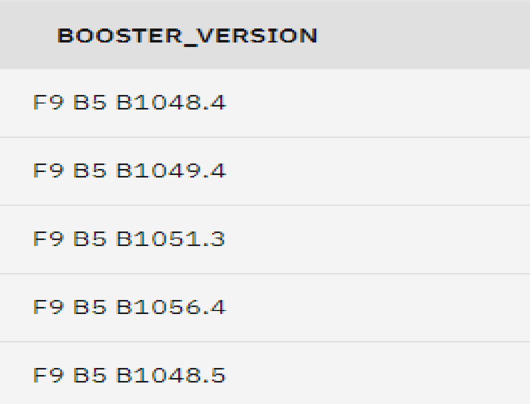
QUERY EXPLANATION

We used subqueries here to produce the result. The ***Like* *‘%Failure%‘*** and ***‘%Success%’*** wildcard shows that in the record the ***success*** or ***failure*** phrases is in any part of the string in the records.

Booster Carried Maximum Payload

SQL QUERY

select Distinct BOOSTER_VERSION from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)



BOOSTER_VERSION
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5

QUERY EXPLANATION

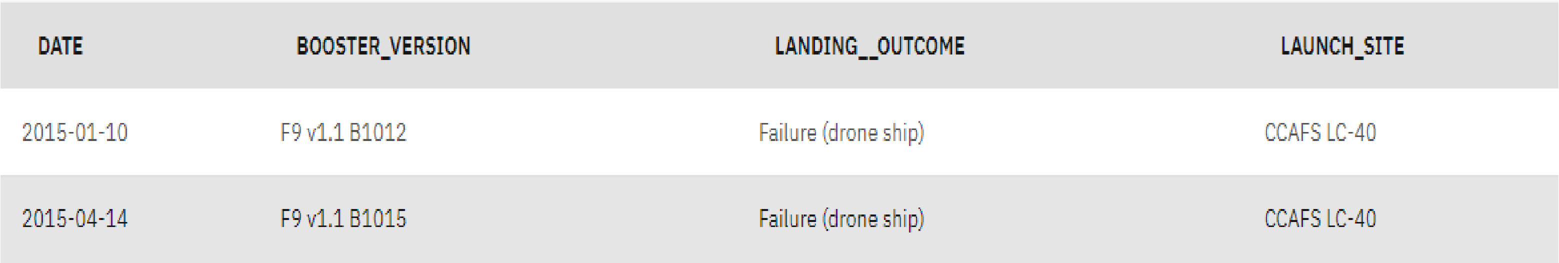
Using the word ***Distinct*** in the query means that it will only sow Unique values in the ***Booster_Version*** column from ***SPACEXTBL***.

Query in ***where*** Clause is a ***Single Row Sub*** Query which provide a single row as an output

2015 Launch Records

SQL QUERY

Select DATE, BOOSTER_VERSION, LANDING__OUTCOMES, LAUNCH_SITE from SPACEXTBL where LANDING__OUTCOME =‘Failure (drone ship)’ and Date like ‘2015%’



DATE	BOOSTER_VERSION	LANDING__OUTCOME	LAUNCH_SITE
2015-01-10	F9 v1.1 B1012	Failure (drone ship)	CCAFS LC-40
2015-04-14	F9 v1.1 B1015	Failure (drone ship)	CCAFS LC-40

EXPLANATION

Selection *__Date__*, ***Booster_Version, Landing__outcome*** and ***Launch_site.***
The ***WHERE*** clause filters the dataset to ***Landing__outcome = Failure(drone ship).***
The ***AND*** clause specifies additional filter conditions ***Date.***

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

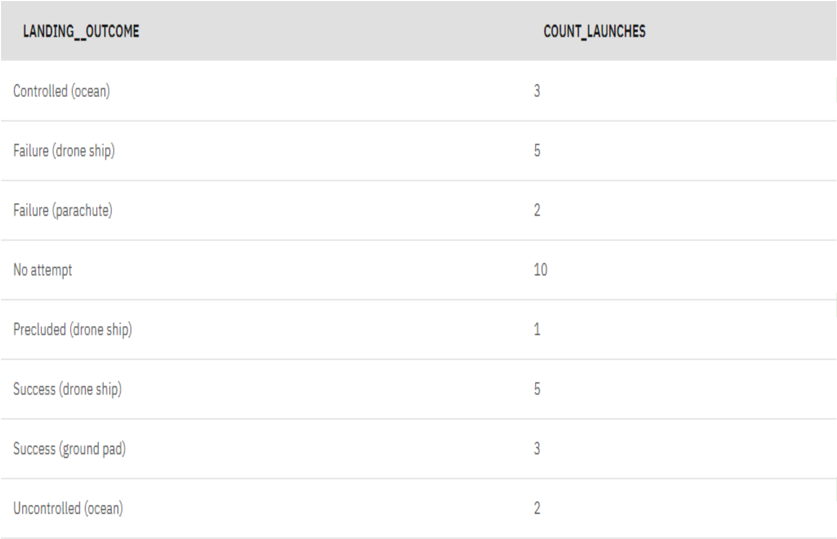
SQL QUERY

Select LANDING__OUTCOME, count(*) as COUNT_LAUNCHES from SPACEXTBL where Date between ‘2010-06-04’ And ‘2017-03-20’ GROUP BY LANDING__OUTCOME

EXPLANATION

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

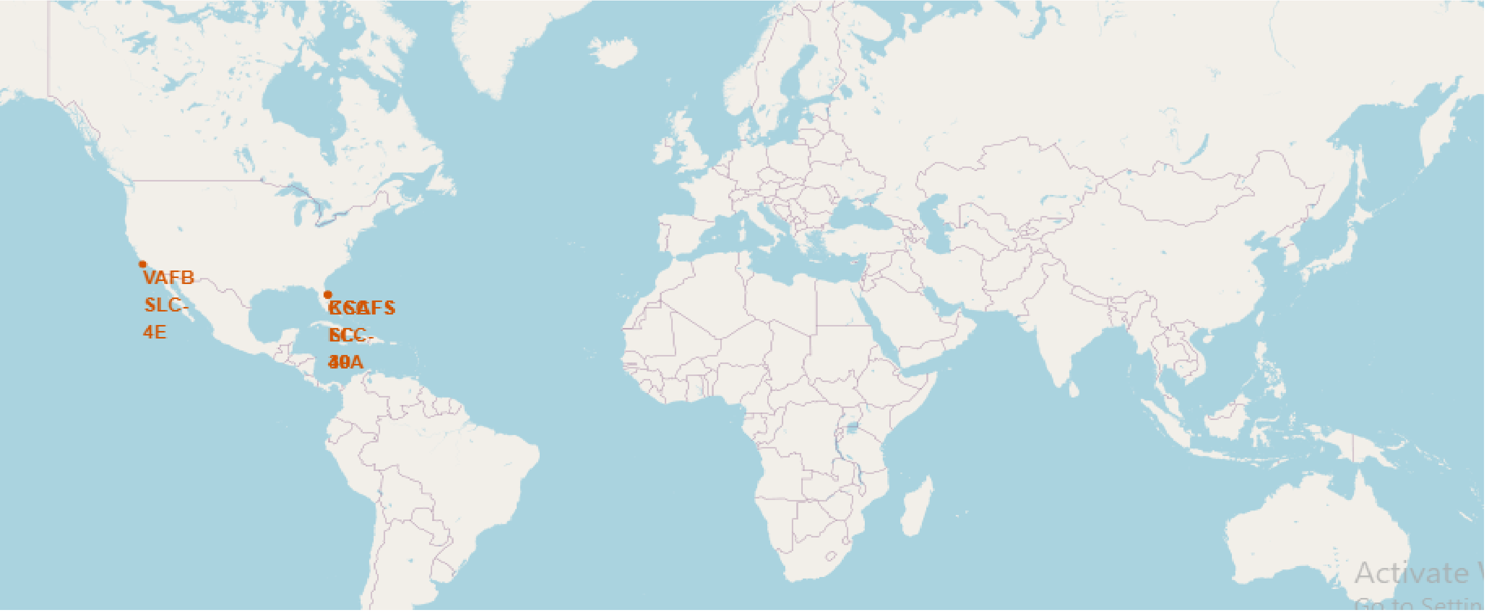
GROUP BY clause group the same kind of data together.



LANDING__OUTCOME	COUNT_LAUNCHES
Controlled (ocean)	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	10
Precluded (drone ship)	1
Success (drone ship)	5
Success (ground pad)	3
Uncontrolled (ocean)	2

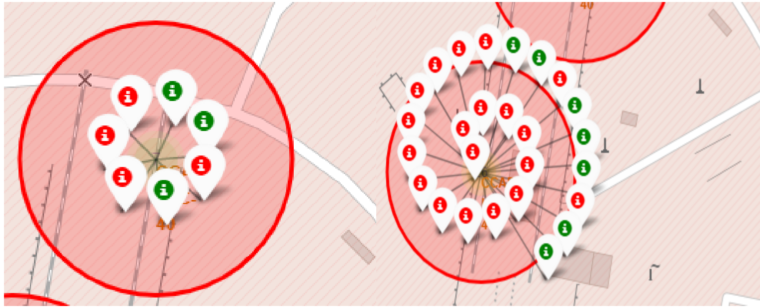
Launch sites Proximity Analysis

All Launch sites global map markers

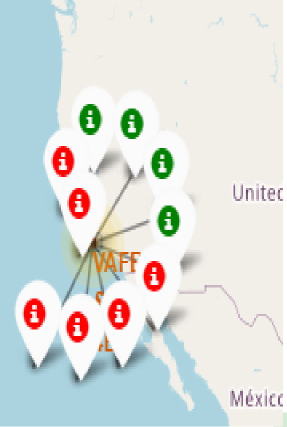


We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California.

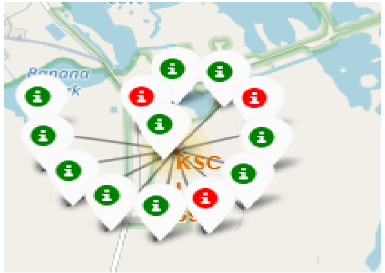
Colour Labelled Markers



7
CCAFS
SLC-
40
26
CCAFS
LC-
40



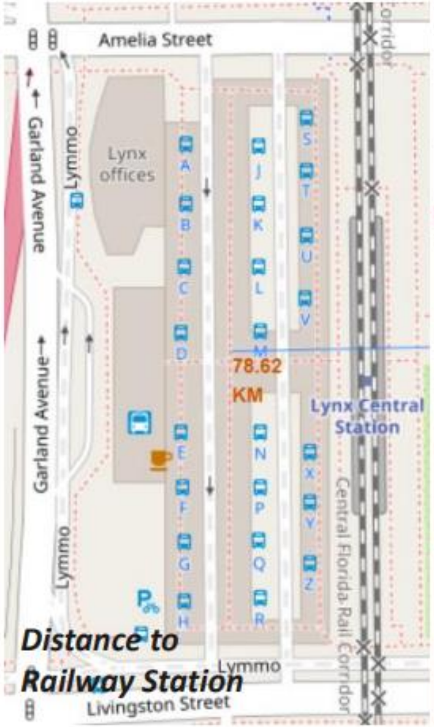
California Launch Site

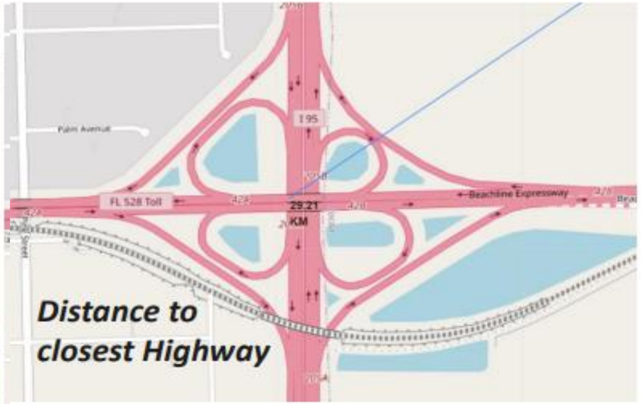


Florida Launch Site

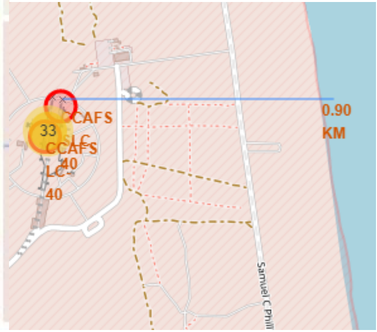
Green Marker shows successful Launches and *Red Marker* shows failures.

Working out Launch Site distance to landmarks to find trends with Haversine formula using CCAFS-SLC-40 as a reference.





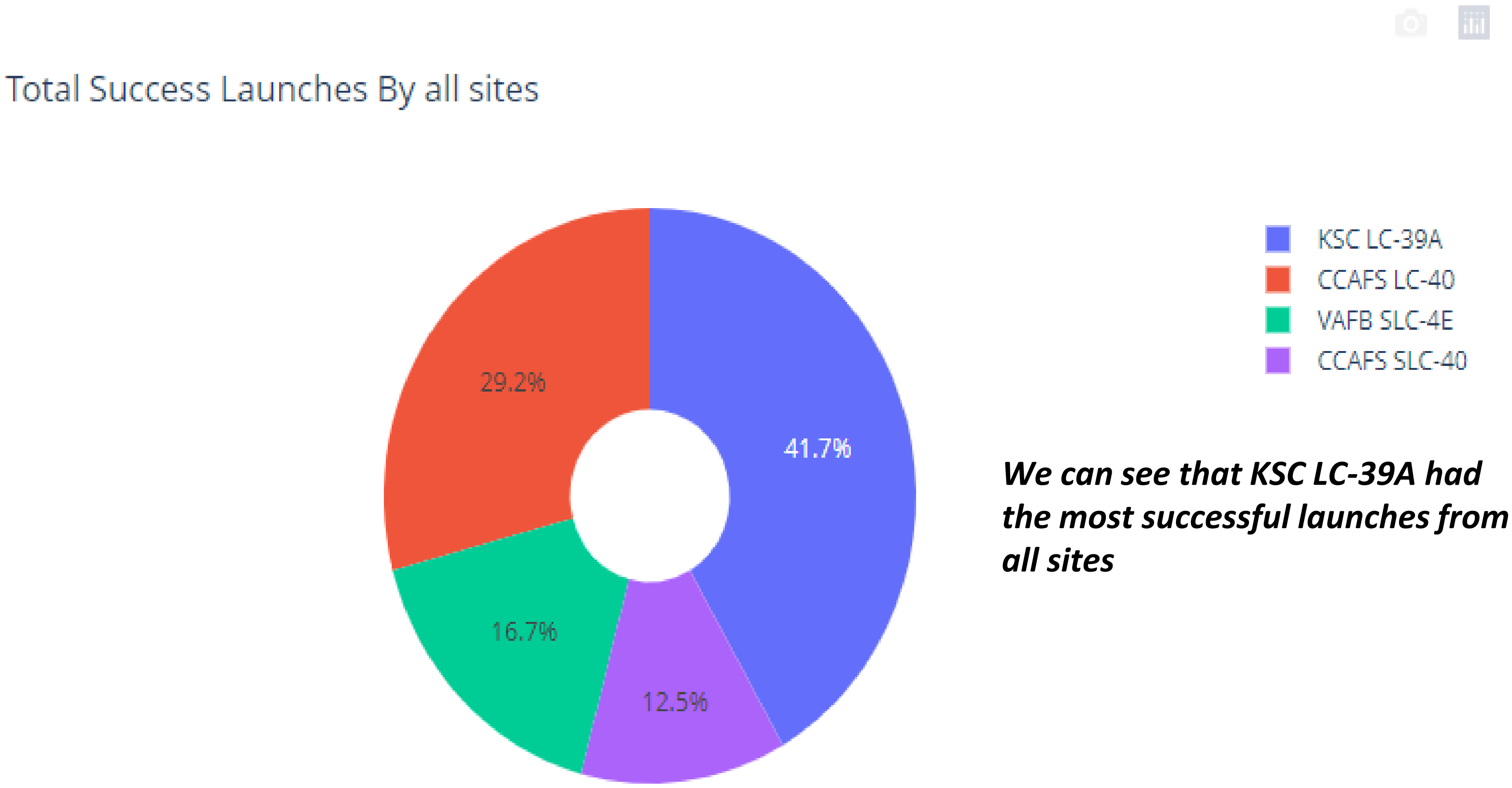
7
CCAFS SLC-40
26
CCAFS LC-40
Distance to coast



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

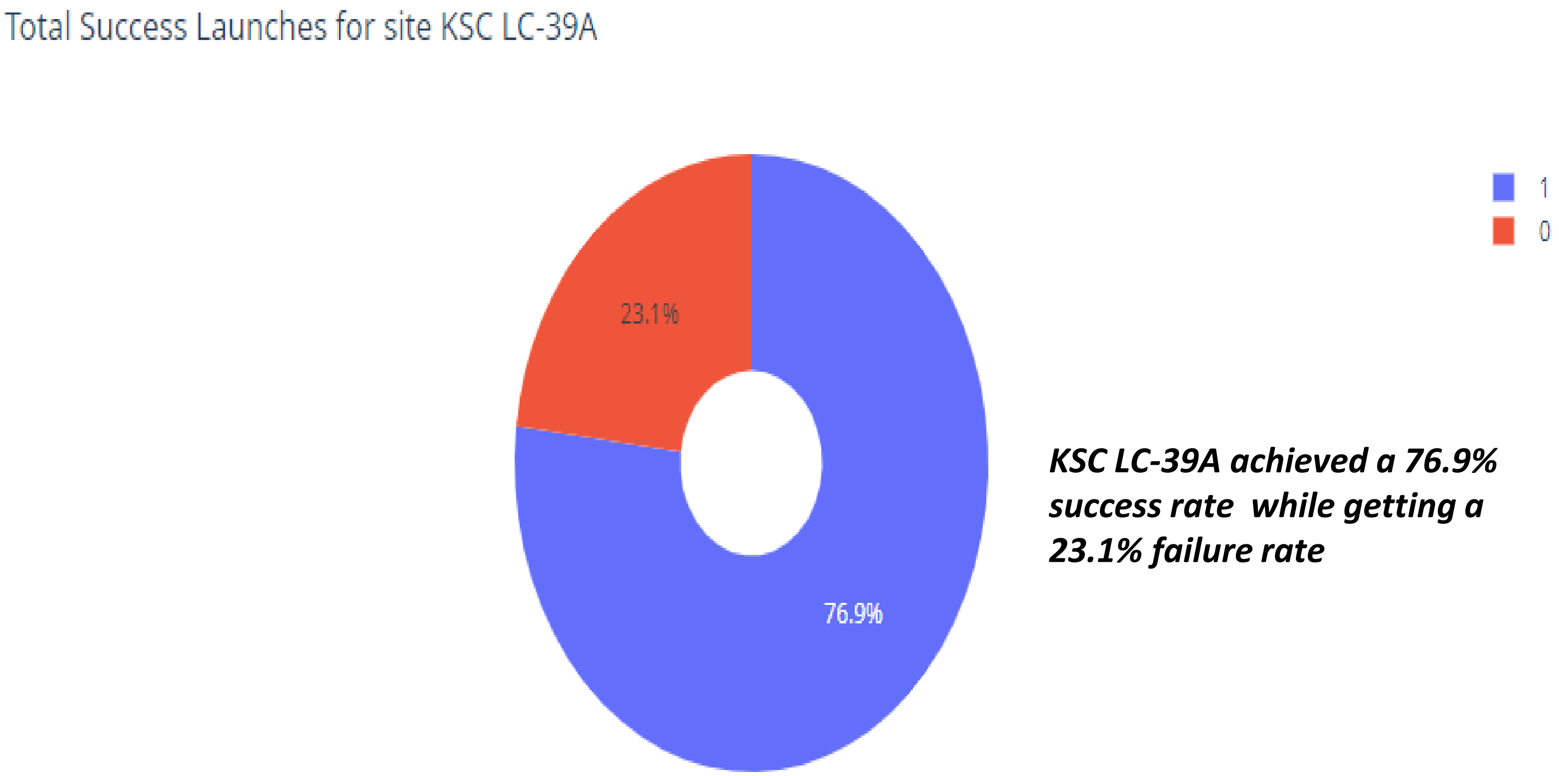
Build a Dashboard with Plotly Dash

DASHBOARD- Pie chart showing the success percentage achieved by each launch site.



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

DASHBOARD- Pie chart for the launch site with the highest launch success ratio.

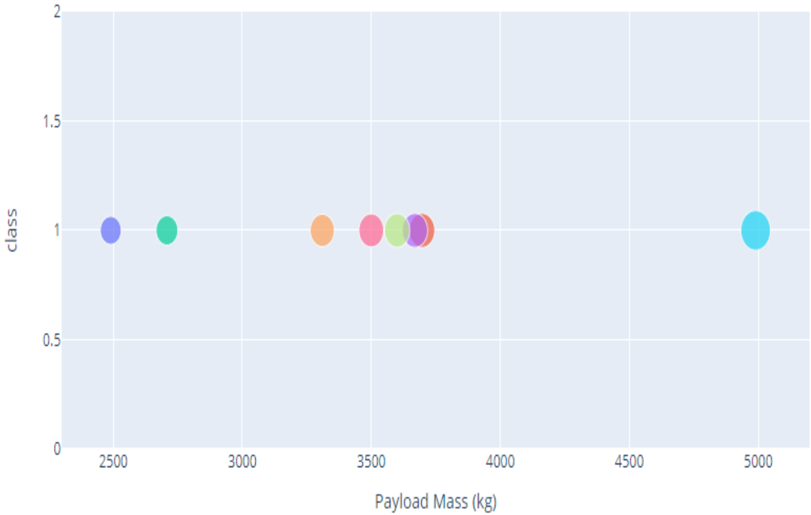


Total Success Launches for site KSC LC-39A

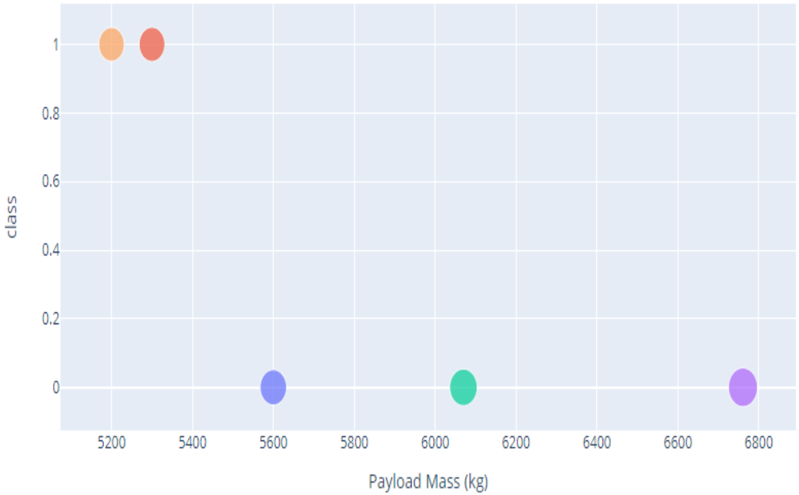
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

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

Low- Weighted PayLoad 0kg-5000kg



Heavy Weighted PayLoad 5000kg-10000 kg

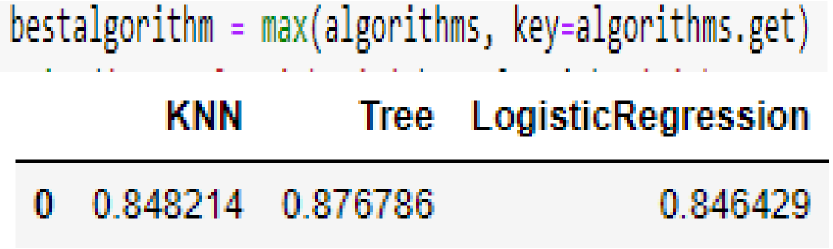


We can see the success rate for low-weighted Payloads is higher than the heavy weighted payloads

Predictive Analysis (Classification)

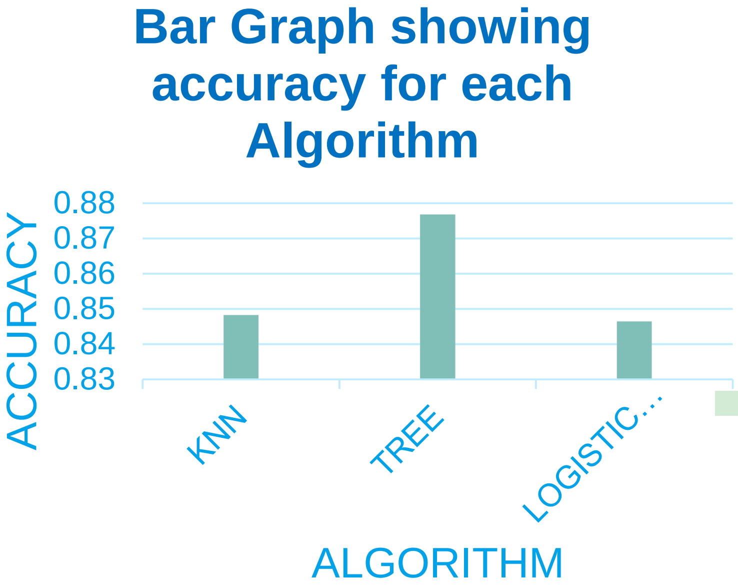
Classification Accuracy using training data

As you can see our accuracy is extremely close but we do have a winner its down to decimal places! using the below function



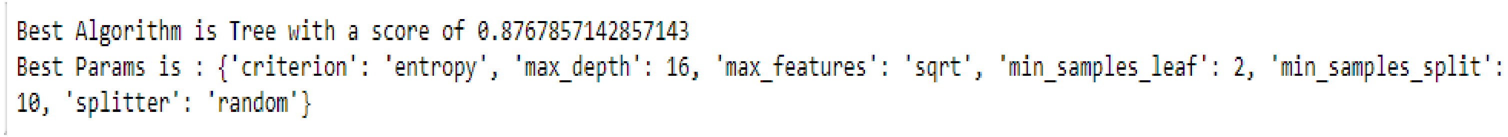
```
bestalgorithm = max(algorithms, key=algorithms.get)
```

	KNN	Tree	LogisticRegression
0	0.848214	0.876786	0.846429



Bar Graph showing accuracy for each Algorithm

The Tree Algorithm wins !!!!

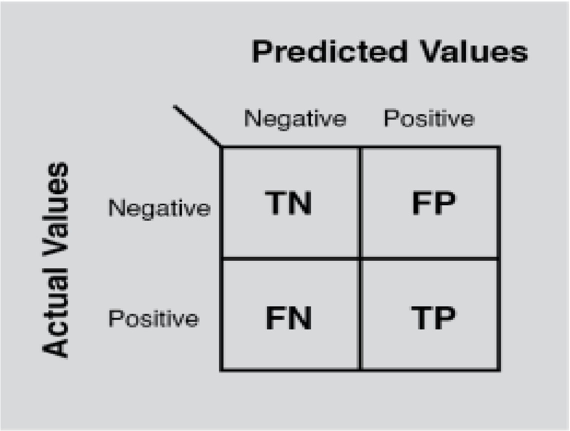


```
Best Algorithm is Tree with a score of 0.8767857142857143
Best Params is : {'criterion': 'entropy', 'max_depth': 16, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split':
10, 'splitter': 'random'}
```

After selecting the best hyper parameters for the decision tree classifier using the validation data, we have achieved 88.88% accuracy on the test data.

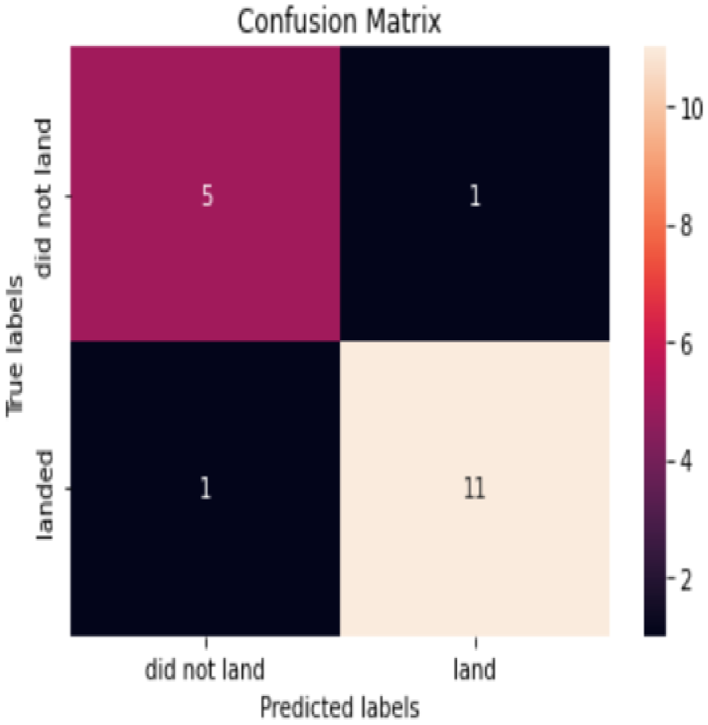
Confusion Matrix for the Tree

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



Predicted Values

Actual Values		Negative	Positive
	Negative	TN	FP
	Positive	FN	TP



Conclusions

- The Tree Classifier Algorithm is the best for Machine Learning for this data set.
- Low weighted payloads perform better than the heavier payloads.
- The success rates for SpaceX launches is directly proportional to 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.

.

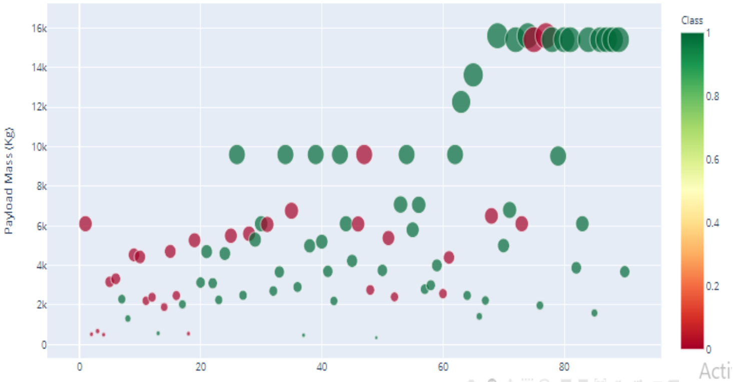
Appendix

- Interactive Plotly
- Decision Tree Construction

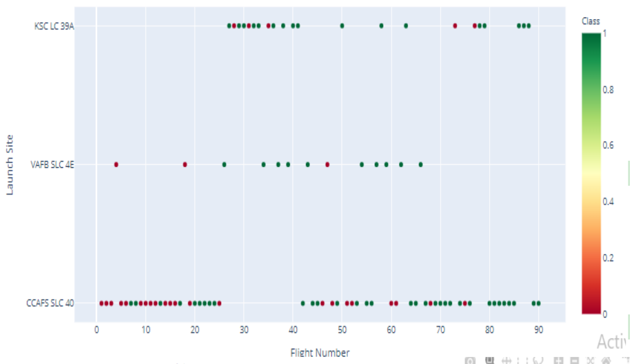
Interactive Plotly

Used plotly instaed of seaborn. They allow a high degree of customization and interactivity, also support multiple languages.

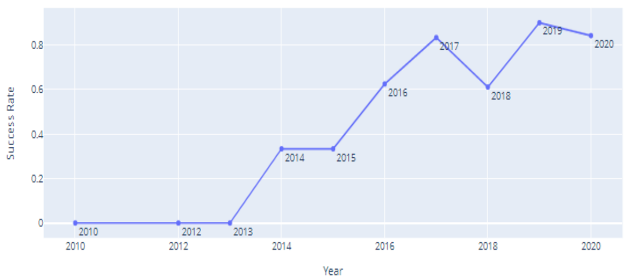
Flight Number vs. Payload Mass



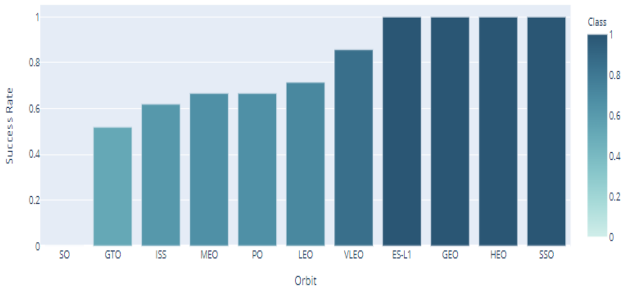
Flight Number vs. Launch Site



Space X Rocket Success Rates

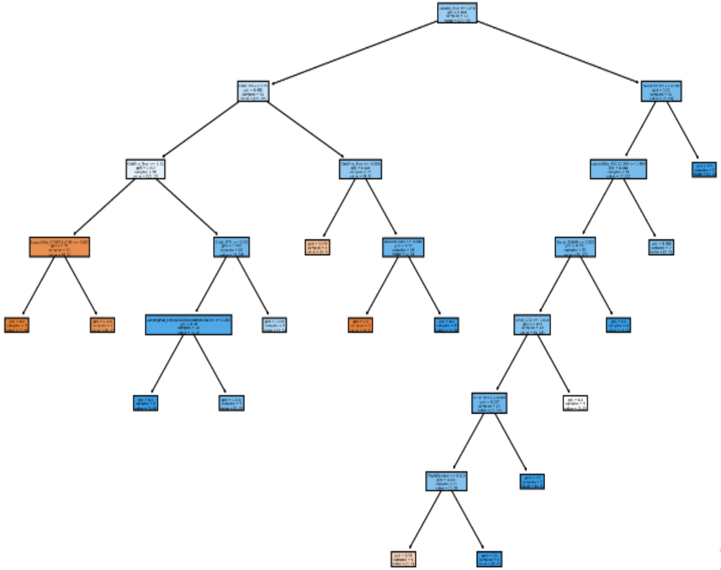


Success Rate vs. Orbit Type



Github URL

DECISION TREE CONSTRUCTION



Decision Tree has been constructed, with Decision Tree Model. We see that we have reached **Gini impurity almost near to 0** via this tree Model. From this we can determine the correct combination of conditions **where the probability of the success will be the highest.**

THANKS!

Any questions?

You can find me at:

swatinarang1225@gmail.com