Winning Space Race with Data Science

Vikas

6 May 2023

Outline

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

Executive Summary

- **Summary of methodologies:**
  - Data Collection using API and Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis (EDA) with Data Visualization
  - Exploratory Data Analysis (EDA) with SQL
  - Interactive Map Visualization with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive Analysis (Classification)
- **Summary of all results:**
  - Exploratory Data Analysis (EDA) Results
  - Interactive Analytics with Map and Dashboard
  - Predictive Analysis

Introduction

- ## Project background and context

The aim of this project is:

- ➢ To determine the price of each launch
- ➢ To predict if the Falcon 9 first stage will successfully land

SpaceX advertises on its website that the Falcon 9 rocket launch cost 62 million dollars. Other providers cost upward of 165 million dollars each.
Here we can see clearly, a big price difference between SpaceX and other competitors because SpaceX can reuse the first stage. So, SpaceX doing a lot of saving than others. Therefore, We can determine the cost of a launch, by determining if the first stage will land.
This information is helpful for a new revival company if it wants to compete with SpaceX for a rocket launch.

- ## Problems you want to find answers

- ➢ Which factors are affect the success of landing.
- ➢ What is the influence of each relationship variables in success or failure of a landing.
- ➢ What are conditions which will allow to achieve best landing of a rocket.

Section 1

Methodology

Methodology

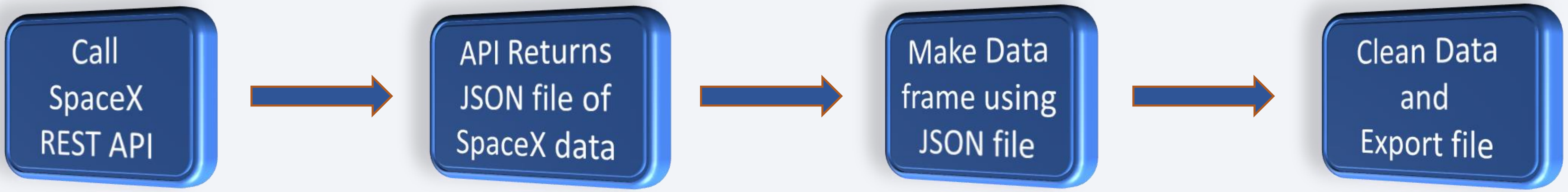
Executive Summary

- Data collection methodology
  - Using SpaceX REST API and Web Scraping (from Wikipedia)
- Perform data wrangling
  - Transform and Clean the data by drop unnecessary columns
  - One Hot Encoding data field for Machine Learning purpose in classification models
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Scatter and Bar plot using to show relationship
- Perform interactive visual analytics using Folium and Plotly Dash
  - Folium and Dash Plotly use to achieve interactive visualization
- Perform predictive analysis using classification models
  - Build, tune, evaluate classification models Like - LR, KNN, SVM and TR.

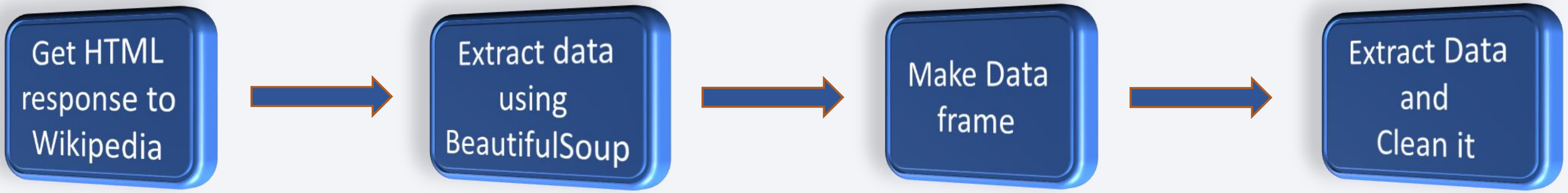
Data Collection

We collect SpaceX launch data from **SpaceX REST API** and through **Web Scrapping** Wikipedia.

- The API gives us data about launches, including information about the rocket used, payload delivered, launches specifications, landing specifications and landing outcome. The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.



- Another way to obtain launch data is Web scraping, it using the python *"BeautifulSoup"* package to web scrape valuable launch records and parse the data into data frame for further analysis.



Data Collection – SpaceX API

1. Getting Response from SpaceX API

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

2. Convert Response to a JSON file

```
# Use json_normalize meethod to convert the js
response = requests.get(static_json_url)
response.json()
data = pd.json_normalize(response.json())
```

3. Transform data

```
# Call getBoosterVersion
getBoosterVersion(data)
```

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

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

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

4. Create 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}
```

5. Create Data frame using dictionary

```
# Create a data from launch_dict
data_falcon = pd.DataFrame(launch_dict)
```

6. Filter the data frame to only include Falcon 9 launches

```
data_falcon9 = pd.DataFrame(data_falcon[data_falcon['BoosterVersion']!='Falcon 1'])
```

7. Export file as a CSV file

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

To view notebook on GitHub :- Click Here

Data Collection – Scraping

1. Getting Response from HTML

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

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

2. Create BeautifulSoup Object

```
# Use BeautifulSoup() to create a BeautifulSoup object
soup = BeautifulSoup(data)
```

3. Find All Tables

```
# Use the find_all function in the BeautifulSoup o
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

4. Get Column Names

```
column_names = []

# Apply find_all() function with `th` element on f
# Iterate each th element and apply the provided e
# Append the Non-empty column name (`if name is no

for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)
```

5. Create 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']=[]
```

6. Create Data frame from Dictionary

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

7. Append the data into dictionary and create data frame from it

```
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
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
```

(See full code in the notebook)

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

7. Export file as a CSV file

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

To view notebook on GitHub :- Click Here

Data Wrangling

In the data set, there are several different cases where the booster did not land successfully. Some example of the mission outcome are give below -

- ***True Ocean*** means successfully landed, **False Ocean** means unsuccessfully landed
- **True RTLS** means successfully landed, **False RTLS** means unsuccessfully
- **True ASDS** means successfully landed, **False ASDS** means unsuccessfully landed

We need to convert those outcomes (String variable) into Training Labels (Categorical variables). Where, 1 means the booster successfully landed and 0 means it was unsuccessful.

1. Load dataset

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_pa
df.head(10)
```

1. Calculate the number of launches on each site

```
df.LaunchSite.value_counts()
```

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

2. Calculate the number and occurrence of each orbit

```
df.Orbit.value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

3. Calculate the number and occurrence of mission outcome of the orbits

```
landing_outcomes = df.Outcome.value_counts()
landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

4. Create a landing outcome label from Outcome column

```
landing_class=[] # A list with zero element
for i in df["Outcome"]:
    if i in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

```
df['Class']=landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

5. Determine the success rate

```
df["Class"].mean()
```

```
0.6666666666666666
```

6. Export file as a CSV file

```
df.to_csv("dataset_part_2.csv", index=False)
```

-: To view notebook on GitHub :-

[Click Here]

EDA with SQL

Summary of the SQL queries that we performed:

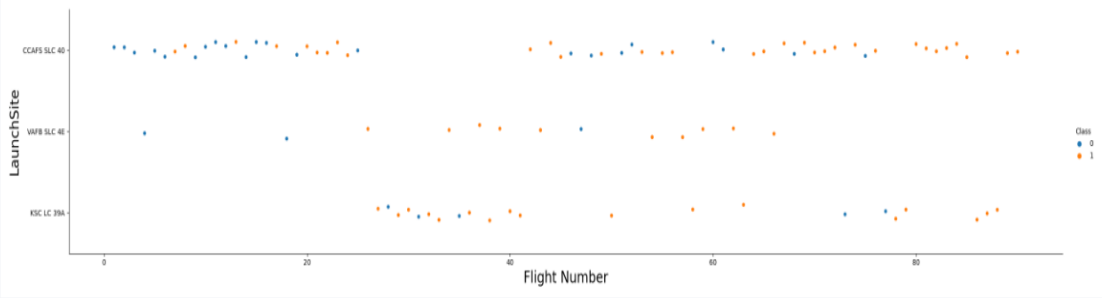
- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass using a subquery.
- List the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

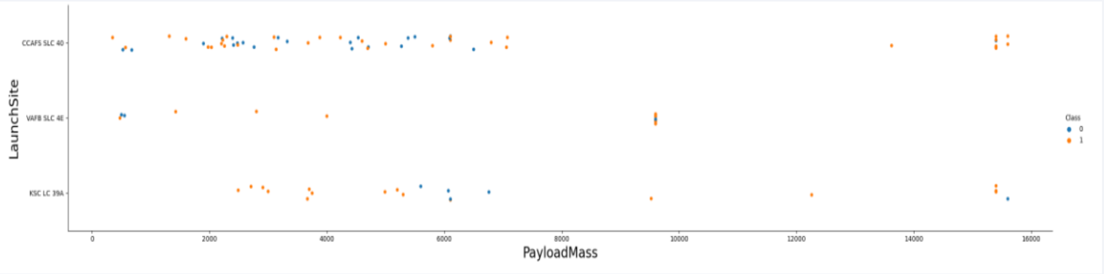
To view notebook on GitHub :- [Click Here]

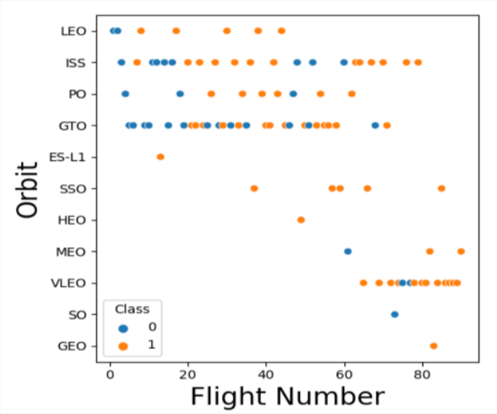
EDA with Data Visualization

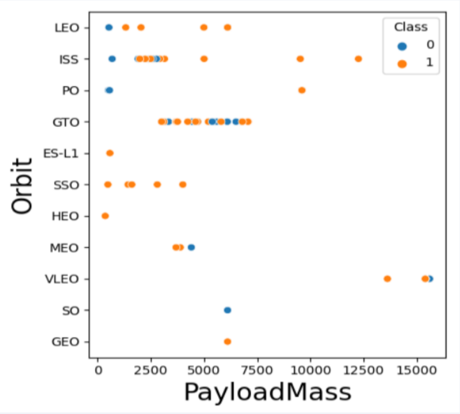
Exploratory Data Analysis (EDA) performed to understand the relationship between various variables and future trends.

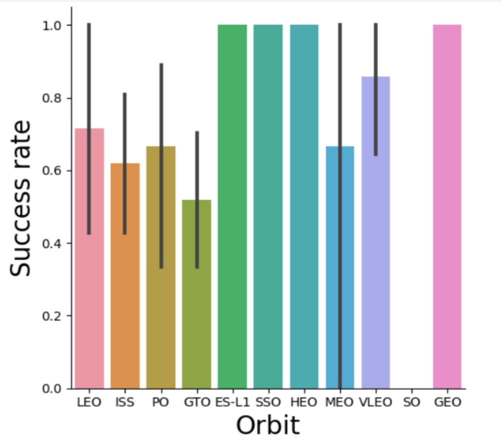
- **Scatter plots** show the relationship between variables.
- **Bar plot** shows the relationship between numeric and categorical variables.
- **Line plot** shows data variables with their trends and it can also help to make prediction.

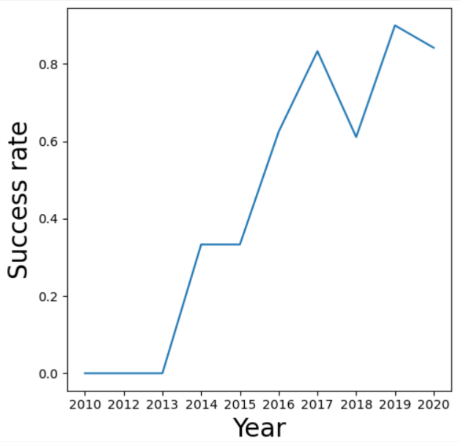












To view notebook on GitHub :- [Click Here]

Build an Interactive Map with Folium

Folium makes data visualization in Python on an interactive leaflet map. We use the latitude and longitude coordinates to creates map objects for each launch site and also can manipulate them as required.

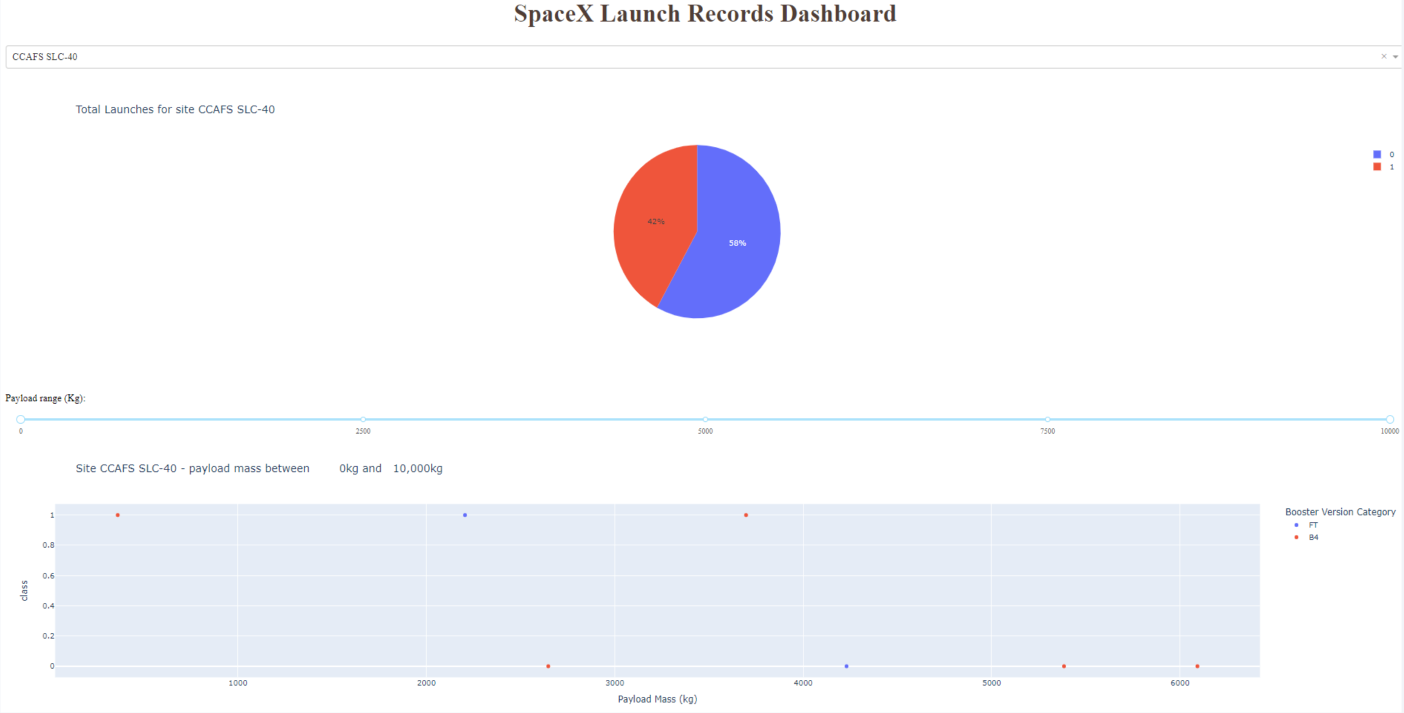
Map Object	Function Code	Explanation (Brief)
Map	folium.Map()	To create a folium map
Markers	folium.Marker()	This object was used to create a mark on the folium map.
Circles	folium.Circle()	This object was used to create a circle on the folium map.
Icons	folium.Icon()	This object was used to create an icon on the folium map which help to identify launch outcomes with Green for successful landing and Red for unsuccessful landing.
Lines	folium.PolyLine()	Used to show distance by creating a polynomial line between two points on folium map.
Ant Path	folium.AntPath()	Used to create an animated line between two points.
Marker Cluster	MarkerCluster()	A marker cluster is a good way to simplify a map because it is containing many markers that have the same coordinate.

To view notebook on GitHub :- Click Here

Build a Dashboard with Plotly Dash

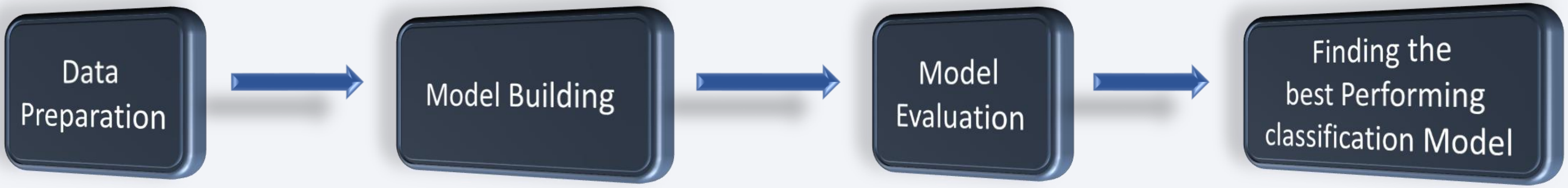
The dashboard has Dropdown, Pie chart, Range slider and Scatter plot components which allows us an interactive and seamless analysis of the relationship between launch sites (using Dropdown), payload ranges (using Range Slider) and success or failure rates (using Pie Chart).

- **Dropdown:** it allows us to choose the launch site or all launch sites.
- **Pie chart:** It shows the percentage of the total success and failure for the launch site chosen from the dropdown menu.
- **Rangeslider:** It allows to select a payload mass in a fixed range.
- **Scatter Plot:** It shows the relationship between Payload Mass and Success.



To view source code file on GitHub :- Click Here

Predictive Analysis (Classification)



Data Preparation

- Load dataset using Pandas and NumPy
- Transform data
- Split data into training and test datasets

Model Building

- Set parameters for each algorithm to GridSearchCV
- Train Machine Learning Model algorithm with training dataset

Model Evaluation

- Find best Hyperparameter
- Calculate the accuracy with test data set
- Plot confusion matrix

Finding the best Performing classification Model

- Finally, find the best performing model according to their accuracy score.
- High accuracy means High preferable model

These steps are separately performed for each type of model like SVM, DT, KNN, and Logistic Regression.

To view notebook on GitHub :- [Click Here]

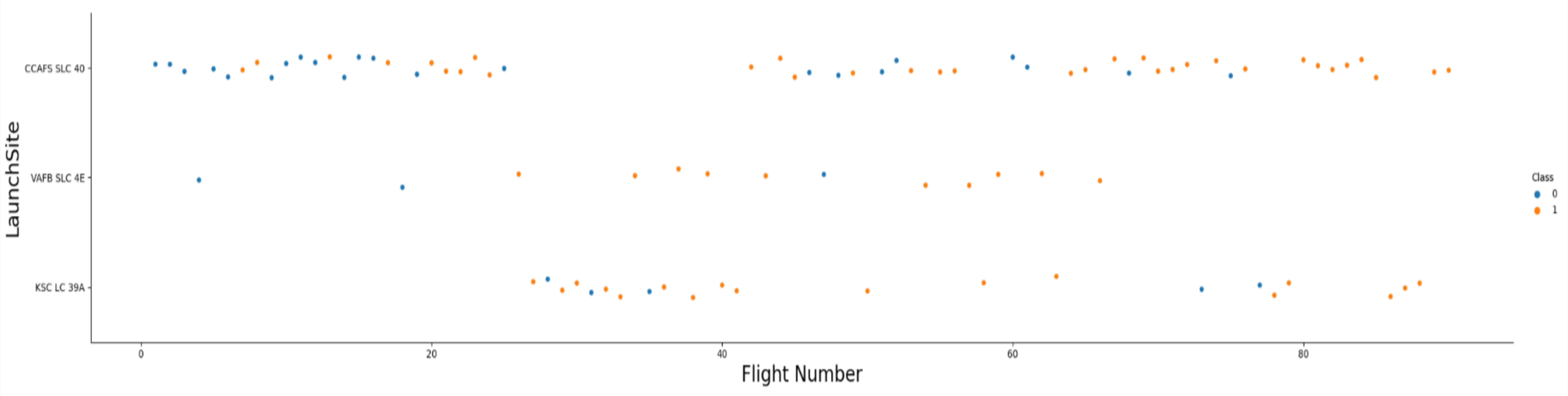
Results

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

Section 2

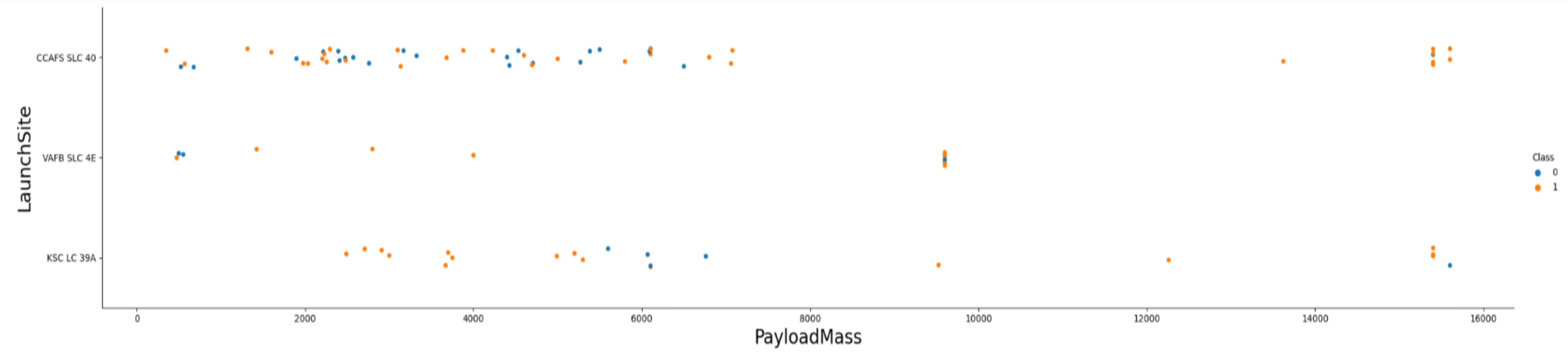
Insights drawn from EDA

Flight Number vs. Launch Site



We can observe that Launch site CCAFS SLC 40 has the highest success rate.

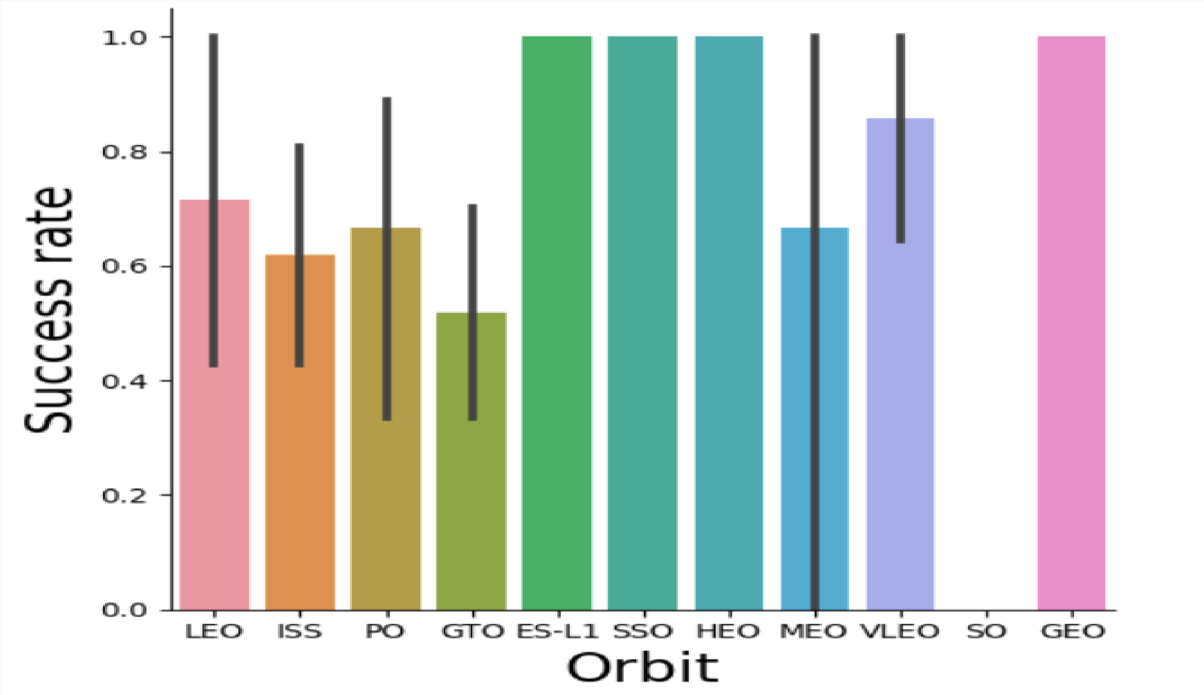
Payload vs. Launch Site



- We observe that Launch site VAFB SLC 4E launch site there are no rockets launched for heavy payload mass (greater than 10000).
- We observe, Majority of rocket launches for CCAFS SLC 40 launch site are between 2000 and 8000 and has a high success rate compared to other launch.
- We also observe low payload mass can quit favor the successful landing in comparison to too high payload mass.

Success Rate vs. Orbit Type

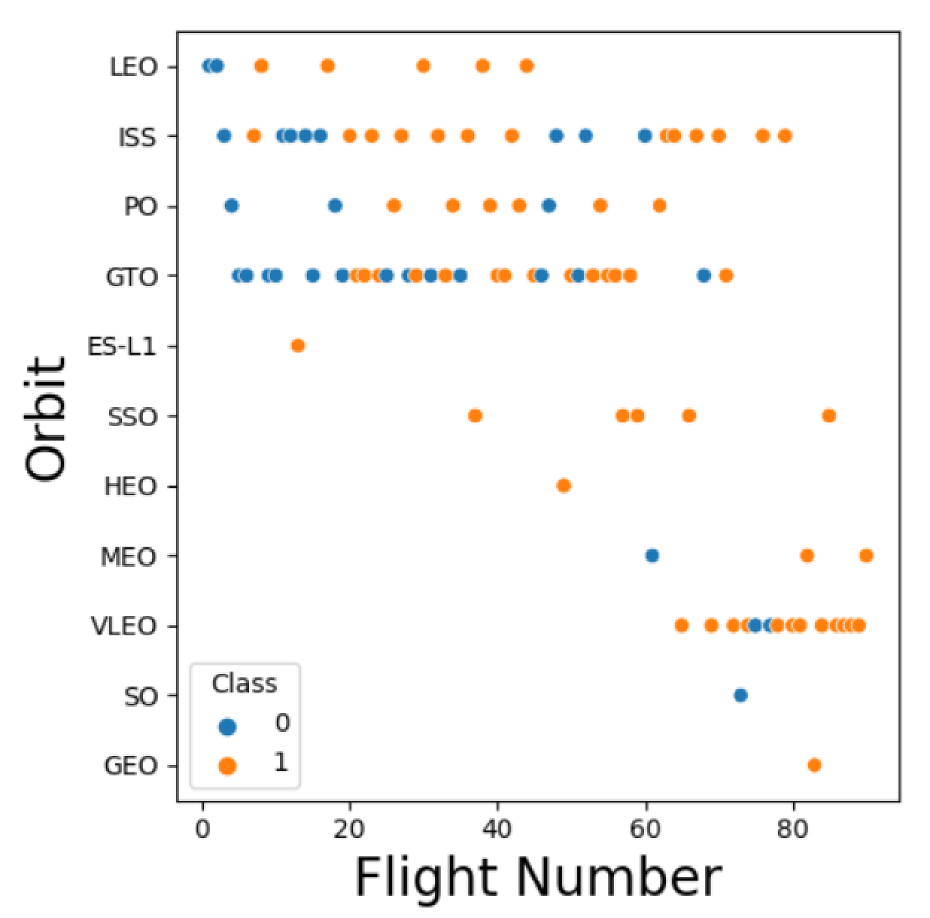
- We observe that ES-L1, SSO, HEO, and GEO orbit type have the highest success rate.
- While SO orbit has 0 (zero) success rate.



Flight Number vs. Orbit Type

- We can see that in the LEO orbit success appears related to the number of flights while GTO orbit seems to be no relationship between flight number and orbit.

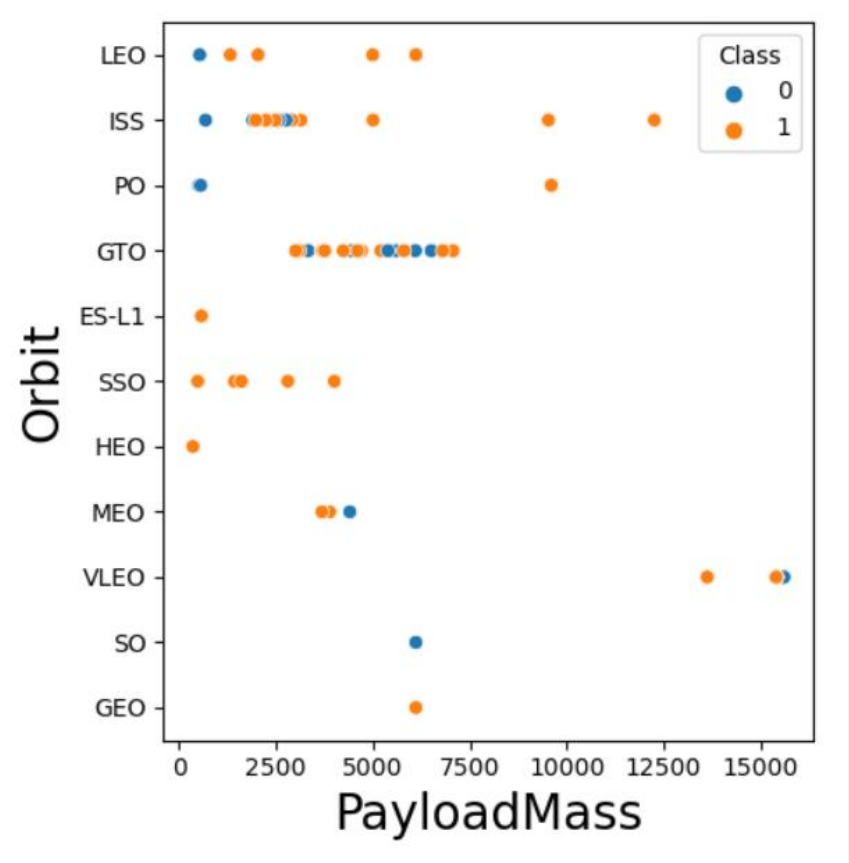
- SSO and HEO have high success rate.



Payload vs. Orbit Type

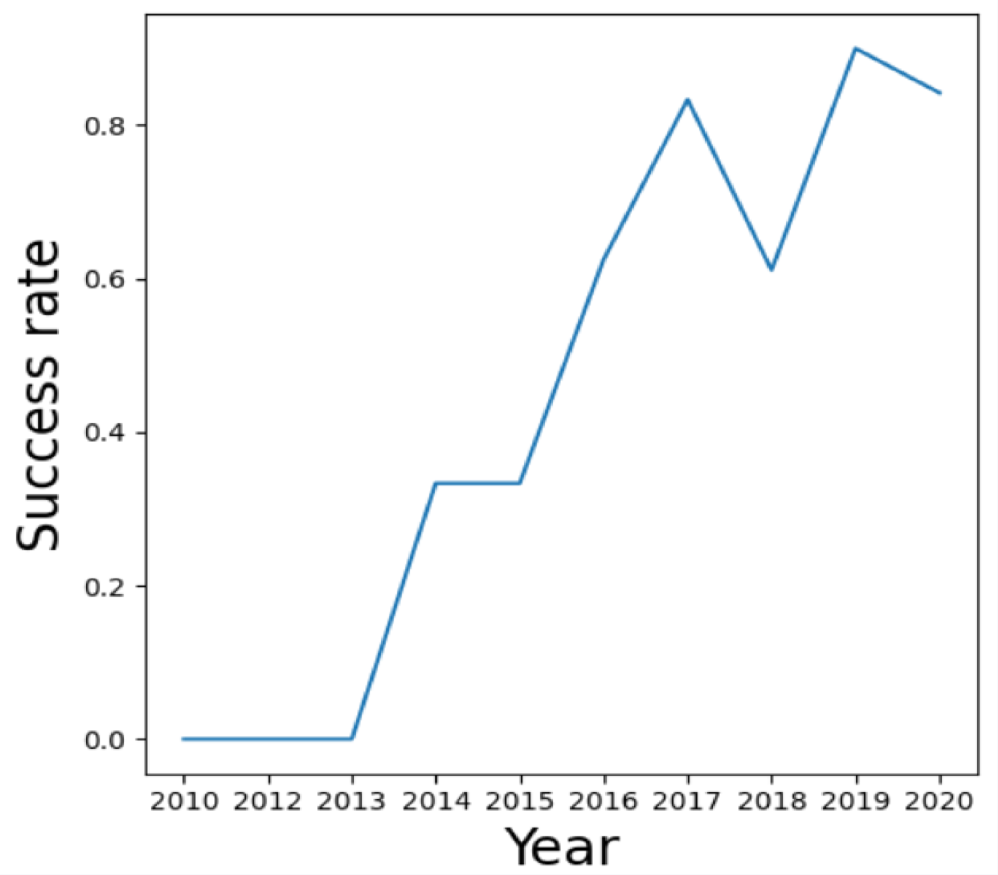
- The successful landing or positive landing rate is more for PO, LEO and ISS with heavy payloads.

- For GTO we cannot distinguish this well as both successful landing rate and unsuccessful landing rate are there.



Launch Success Yearly Trend

We can observe that the success rate has increased over the years after the year 2013.



All Launch Site Names

SQL query:

```
%sql select Distinct(Launch_Site) from SPACEX
```

Result:

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

Description: We use DISTINCT to show all launch site names from the SPACEX table.

Launch Site Names Begin with 'CCA'

SQL query:

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

Result:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	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)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Description: In the SQL query LIMIT 5 keyword is used to fetch only 5 records from table SPACEX, and in the WHERE clause LIKE keyword with wild card 'CCA%' is used to condition that suggests the launch site name must be start with CCA.

Total Payload Mass

SQL query:

```
%sql select sum(PAYLOAD_MASS__KG_) as total_payload_mass from SPACEX where Customer = 'NASA (CRS)'
```

Result:

total_payload_mass
45596

Description: The above SQL query is used to the sum of all payload masses. And WHERE clause is used to perform the query only for the NASA (CRS) customer.

Average Payload Mass by F9 v1.1

SQL query:

```
%sql select avg(PAYLOAD_MASS__KG_) as average_payload_mass from SPACEX where BOOSTER_VERSION = 'F9 v1.1'
```

Result:

average_payload_mass
2928

Description: The above SQL query is used to the calculate average of payload masses for booster version F9 v1.1 from SPACEX table and AVG() function with WHERE clause is used to perform the query only for booster version which start with ‘F9 v1.1’.

First Successful Ground Landing Date

SQL query:

```
%sql select min(Date) as date from SPACEX where LANDING__OUTCOME like 'Success (ground pad)'
```

Result:

DATE
2015-12-22

Description: The above SQL query is used to find first successful ground landing date from SPACEX table and MIN() function is used to select minimum (oldest) date from table data while WHERE clause is used to perform the query only for successful ground landing.

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL query:

```
%sql select DISTINCT(BOOSTER_VERSION) from SPACEX \
where LANDING__OUTCOME like 'Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000
```

Result:

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

Description: In the above SQL query is used to filter the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 from SPACEX table with WHERE and AND clauses for specific conditions.

Total Number of Successful and Failure Mission Outcomes

SQL query:

```
%sql select Mission_Outcome, count(Mission_Outcome) as total_number from SPACEX group by Mission_Outcome
```

Result:

mission_outcome	total_number
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Description: Using the above SQL query we find the total number of successful missions is 100 (including payload status unclear mission) and the failure mission outcome is only 1 from SPACEX table.

Boosters Carried Maximum Payload

SQL query:

```
%sql select BOOSTER_VERSION, PAYLOAD_MASS__KG_ as payload_mass from SPACEX \
where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_)  from SPACEX)
```

Result:

booster_version	payload_mass
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

Description: In the above SQL query, a subquery with MAX() function is used to return the maximum payload mass from the SPACEX table and the main query uses subquery results and returns unique booster versions with carried maximum payload mass using the DISTINCT() function.

2015 Launch Records

SQL query:

```
%sql select Date, LANDING__OUTCOME , BOOSTER_VERSION, LAUNCH_SITE from SPACEX \
where Date like '2015%' and LANDING__OUTCOME like 'Failure (drone ship)'
```

Result:

DATE	landing__outcome	booster_version	launch_site
2015-01-10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
2015-04-14	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Description: Using the above SQL query, we find the list of the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015. WHERE and AND clauses filter the result to returns data for in year 2015 and failed outcomes in drone ship.

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

SQL query:

```
%sql select LANDING__OUTCOME, count(LANDING__OUTCOME) as Count from SPACEX \
where Date between '2010-06-04' and '2017-03-20' \
group by LANDING__OUTCOME \
order by count(LANDING__OUTCOME) desc
```

Result:

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

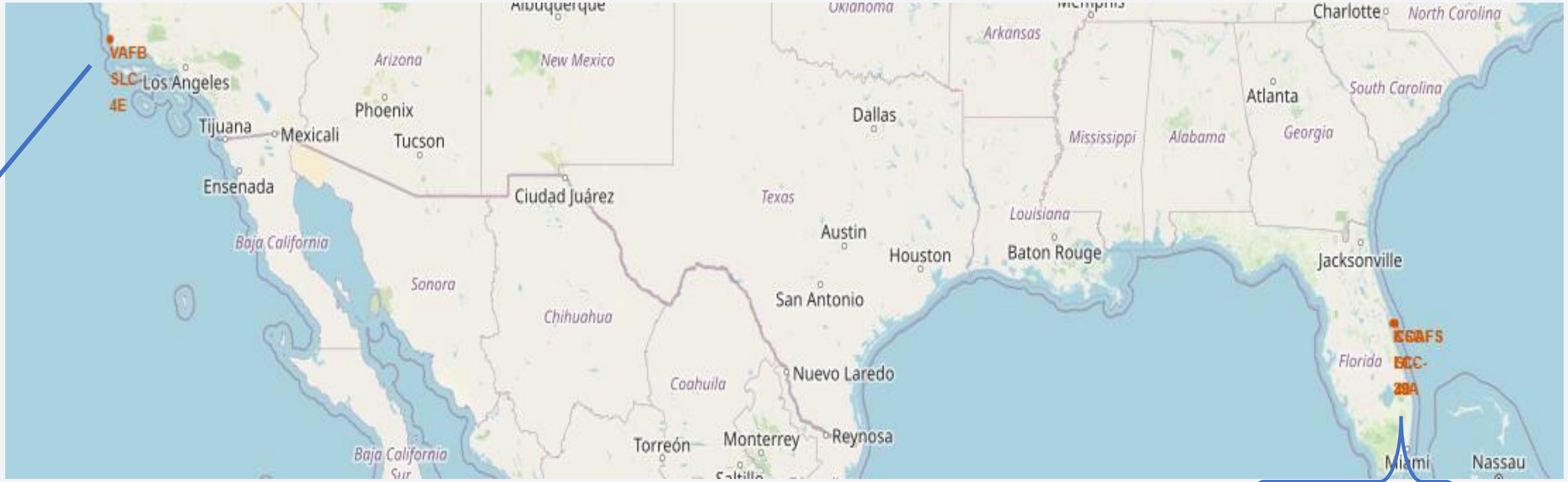
Description: The above SQL query, Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the dates 2010-06-04 and 2017-03-20, in descending order from SPACEX table.

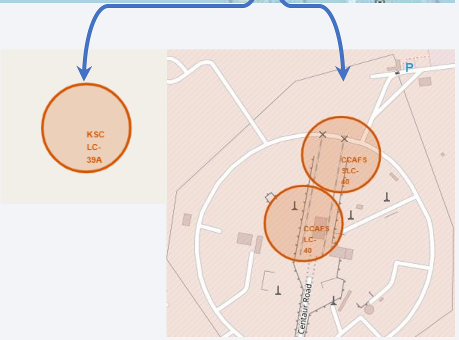
COUNT() function is used to count the landing outcomes with WHERE clause condition between two dates and GROUP BY clause group the results by landing outcome and results counts show in descending order with ORDER BY clause.

Section 3

Launch Sites Proximities Analysis

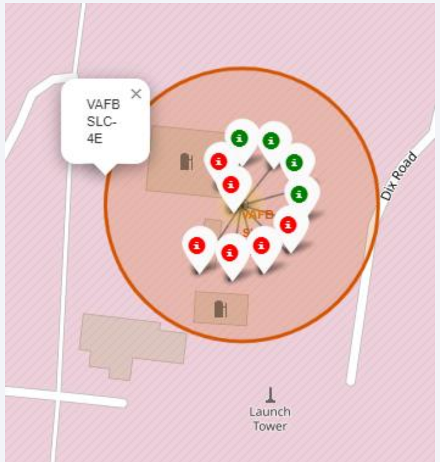
Marked All Launch Sites on a Map using Folium





After zoom-in/zoom-out the map we can observe that all launch sites are very close to the coastal area and proximity to the equator line.

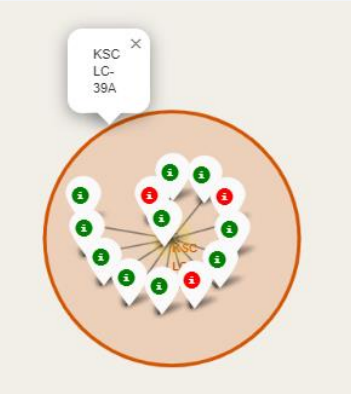
The color-labeled launch outcomes on the map



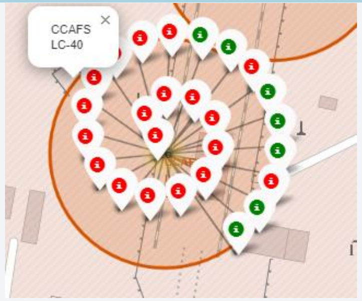
1st Launch site

In the color-labeled markers in the marker cluster Green color marker represents the successful launch while the Red color marker represents the unsuccessful launch.

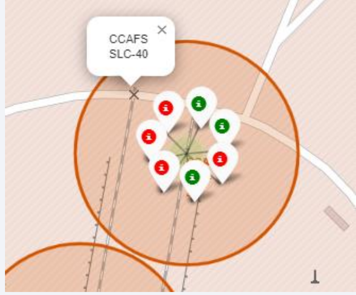
We can easily identify that the KSC LC 39A launch site has a high success rate of launch outcomes.



2nd Launch site

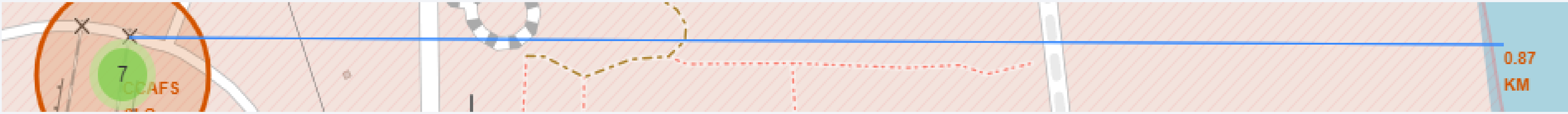


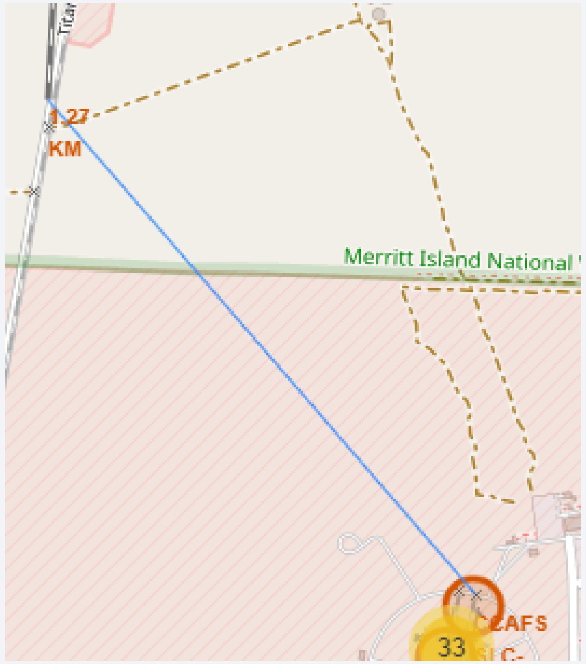
3rd Launch site

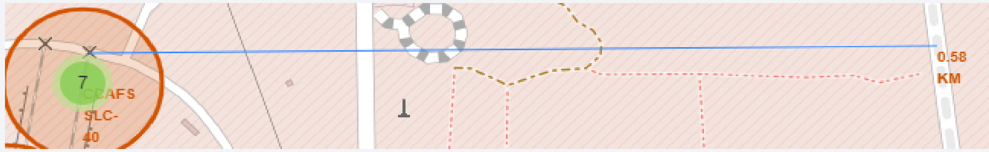


4th Launch site

Distance Between CCAFS SLC-40 and its Proximities







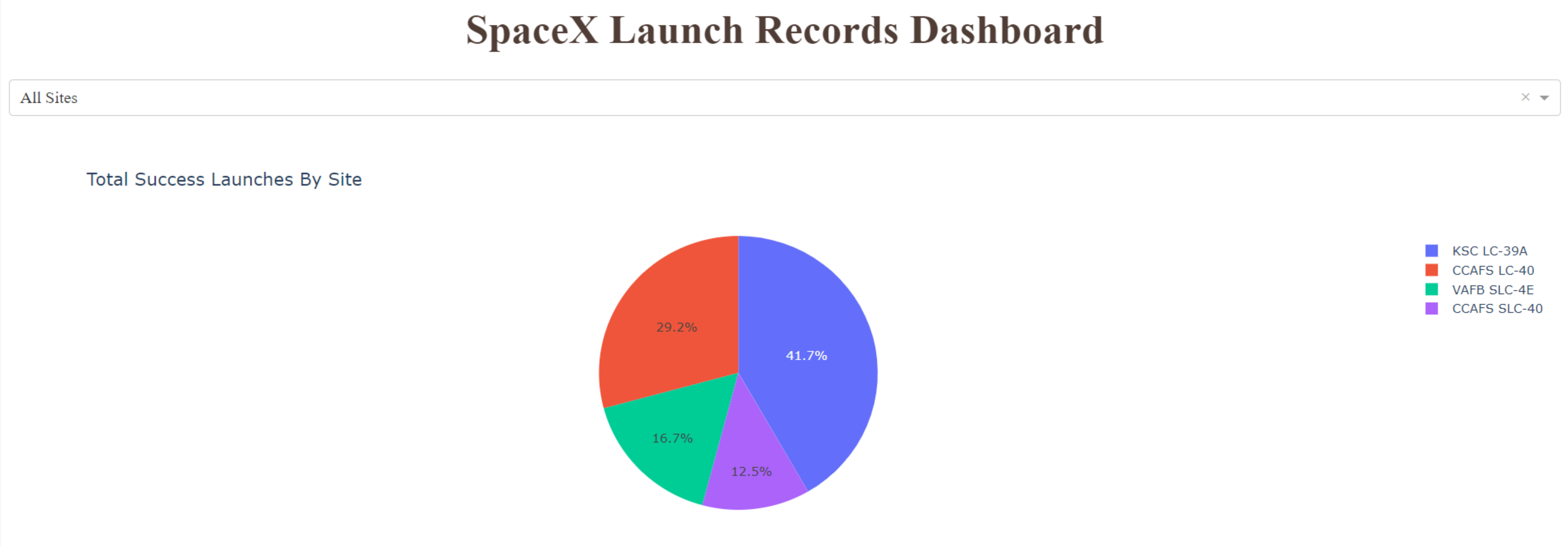
Safety Measurements:

- Are launch sites in close proximity to railways? – **YES (1.27KM)**
- Are launch sites in close proximity to highways? – **YES (0.58KM)**
- Are launch sites in close proximity to coastline? – **YES (0.87KM)**
- Do launch sites keep certain distance away from cities? – **YES (23.12 KM)**

Section 4

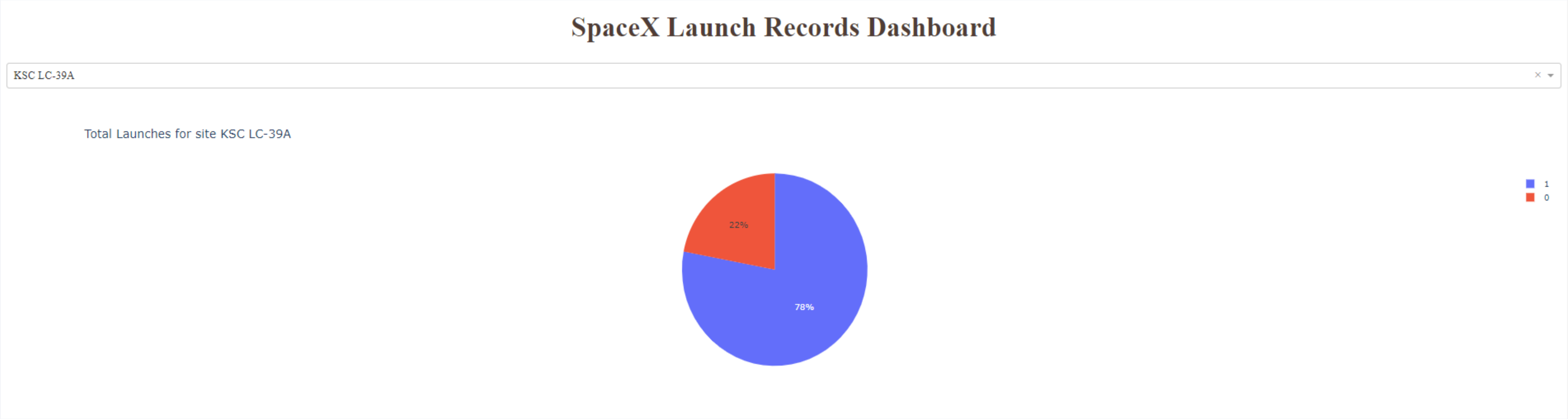
Build a Dashboard with Plotly Dash

Dashboard – Total Success Launches By All Sites



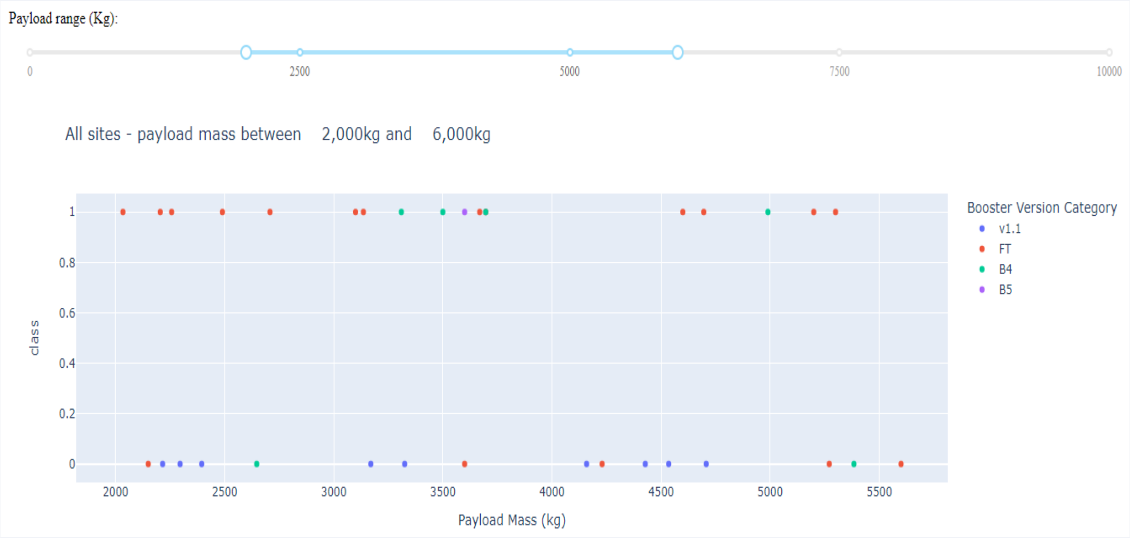
We can observe that the KSC LC-39A launch site has the highest success rate of the launches in comparison to other sites.

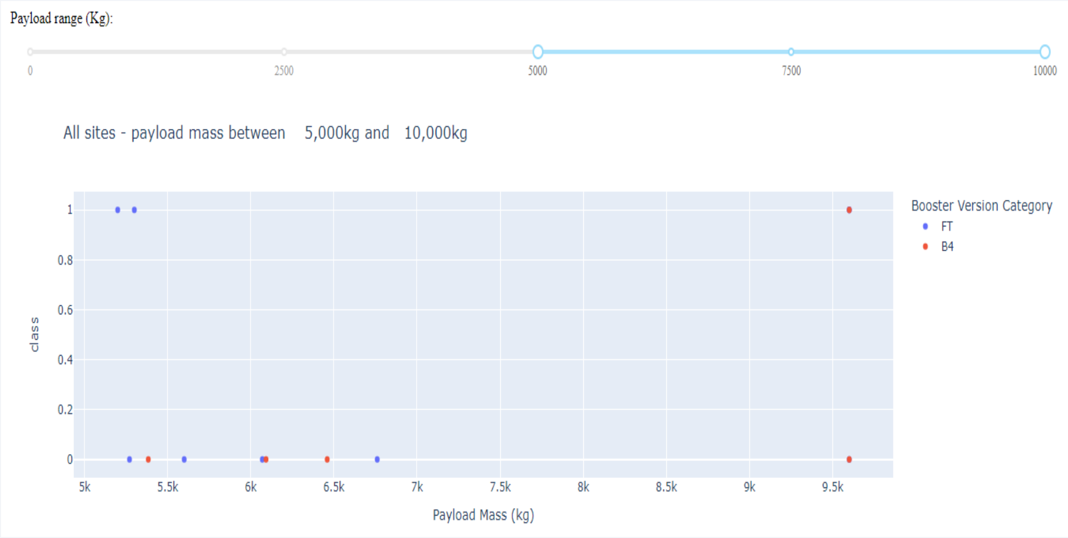
Dashboard – A Launch Site with Highest Launch Success Rate



We observe that the **KSC LC-39A** launch site has a **78% success rate** and a 22% failure rate of launches.

Dashboard – Payload vs. Launch Outcome for all sites with different payload mass ranges





We observe that the Payload masses (between 2,000kg and 6,000kg) have the highest success rate than the too heavy Payload masses (above 6,000kg).

Section 5

Predictive Analysis (Classification)

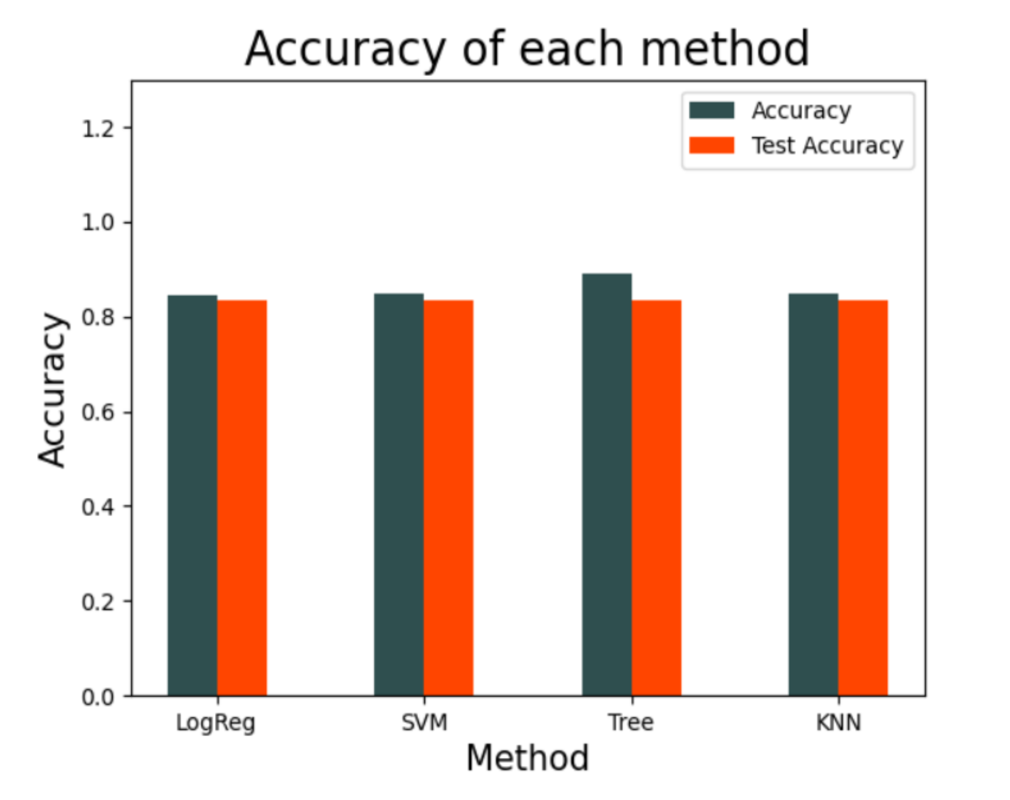
Classification Accuracy

- To find the best method using Exploratory Data Analysis we used the following four types of classification models/methods:
    1. Logistic Regression
    2. SVM
    3. Decision Tree
    4. KNN

	Accuracy	Test Accuracy
LogReg	0.84643	0.83333
SVM	0.84821	0.83333
Tree	0.88929	0.83333
KNN	0.84821	0.83333

- We can observe that **Test accuracy** has the **same** for all models.

- While the **accuracy** of the **Decision Tree** model has the **highest** than others.



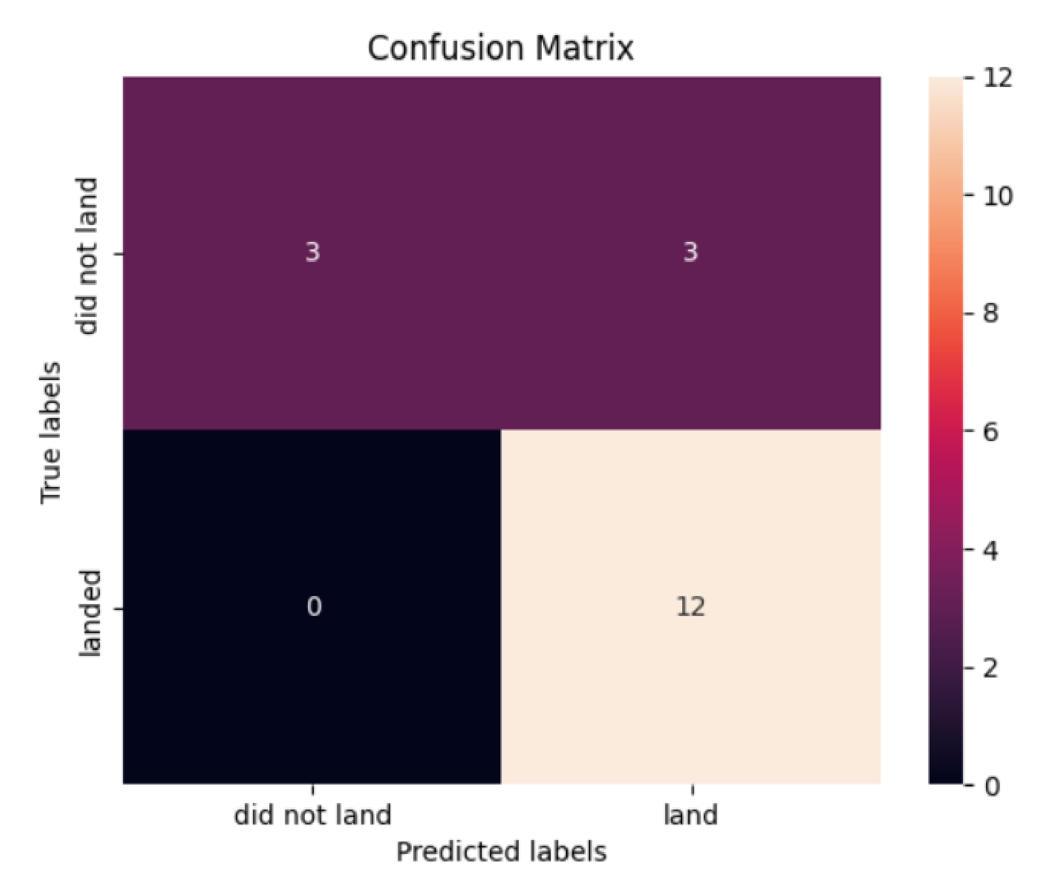
Confusion Matrix

The matrix of all models is identical because of their similar test accuracy.

From the confusion matrix, we can find the following:

1. **Accuracy** = $\frac{TP+TN}{TP+TN+FP+FN}$ = 83.33
2. **Precision** = $\frac{TP}{TP+FP}$ = 0.80
3. **Recall** = $\frac{TP}{TP+FN}$ = 12
4. **F1-Score** = $\frac{2(Precision \times Recall)}{Precesion + Recall}$ = 3

True Positive	(TP)	=	12
True Negative	(TN)	=	3
False Positive	**(FP)**	**=**	**3**
False Negative	(FN)	=	0



We observe that False positives are the main problem of these models.

Conclusions

- The first successful ground landing of the SpaceX launch mission was on 22 Dec 2015.
- The ES-L1, SSO, HEO, and GEO have the highest success rate with 100%.
- The Launch success rate has increased over the years after the year 2013.
- Generally low weighted payloads (between 2,000kg and 6,000kg) have the highest success rate of launches than too-heavy weighted payloads.
- The KSC LC-39A site has the highest success rate of the launch outcomes than others.
- The Decision Tree classifier or method is the best algorithm as a Machine Learning Model for this SpaceX dataset according to the accuracy of train data.

Appendix

- All Notebooks and Dashboard source code related to this project are uploaded on GitHub inside the repository *"IBM - Applied Data Science Capstone Project"*.
- To find all notebooks Click Here

Thank you!