IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Takdanai Yodyoi
9 Jan 2022

Outline

Executive Summary

Summary of methodologies

- Data Collection with Web Scaping
- Data Wrangling
- EDA with Data Visualization
- EDA with SQL
- Interactive Visualization with Folium
- Dashboard with Plotly Dash

Summary of all results

- Data Analysis
- Prediction by Machine learning

Introduction

Project background and context

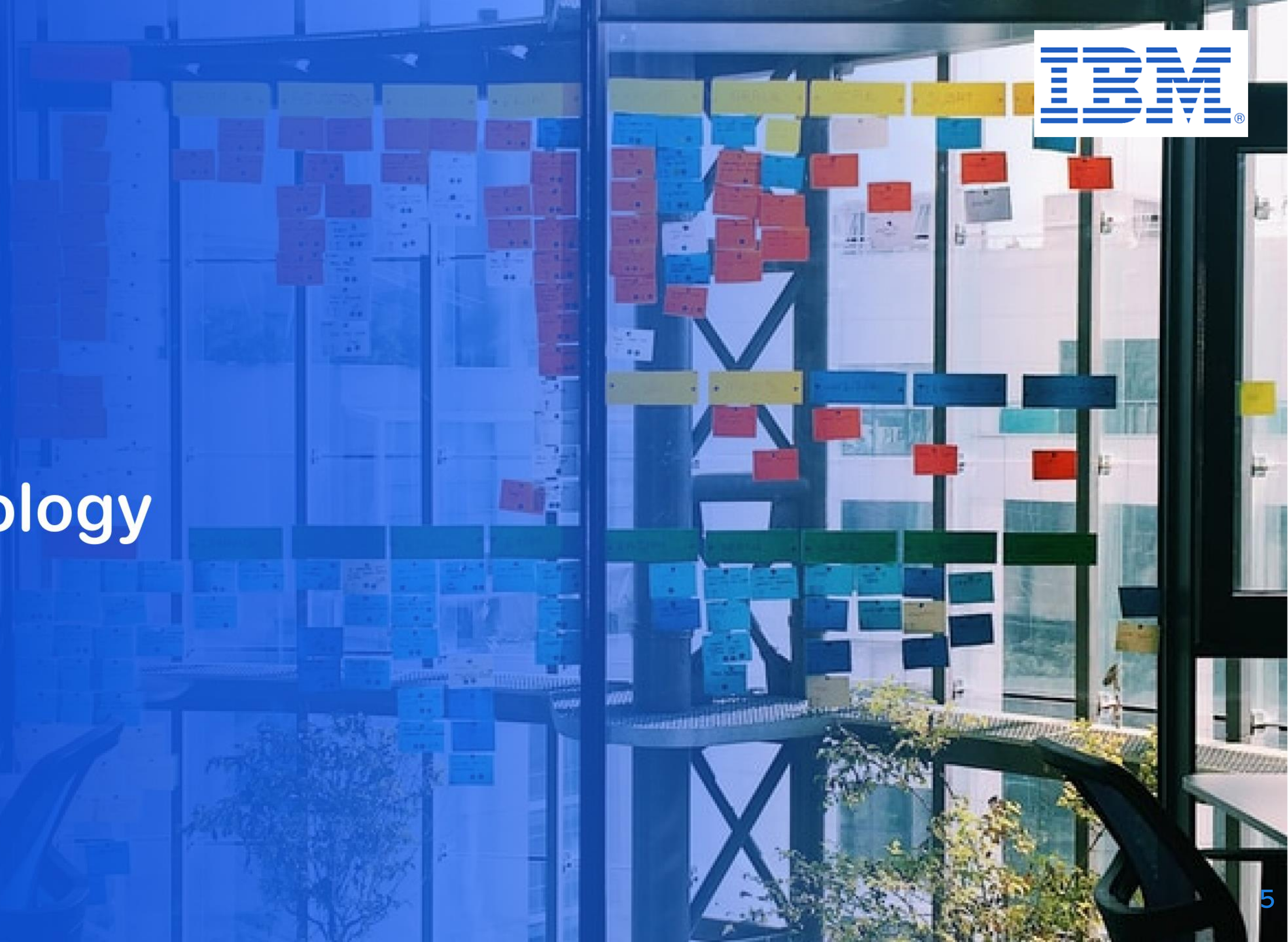
- SpaceY company is one the most successful firms trying to make space travel affordable for everybody. During launch process, there are 2 stages of a rockets involved. Both stages cost highly expensive to manufacture. Therefore, an idea of reuse comes to minimize expenses.
- As a data scientist, we are tasked to an estimate price for each rocket launch for SpaceY. All related parameters of the first stage are considered to formulate suitable models to predict.

Problems you want to find answers

- A study on rates of successful landing of Falcon 9 are considered.
- To achieve that, relation of selected parameters of the first stage is calculated to see how each parameter has an effect on the success percentage.
- Consequently, minimized cost of a rocket launch can be determined

Section 1

Methodology



Methodology

Data collection methodology:	• Web-scraping via BeautifulSoup & REST API from Falcon 9 were extracted and analyzed from Wikipedia data source
Perform data wrangling	• Data were transformed into suitable formats, i.e. one-hot-encoding method , statistical method such as standardize of data were complied to get analytic results
Perform exploratory data analysis (EDA) using visualization and SQL	• Graph plots including scatter, and bar charts showing relationship between independent and dependent valuables.
Perform interactive visual analytics using Folium and Plotly Dash	
Perform predictive analysis using classification models	• Performed analytics on Logistic Regression, Classification tree, and SVM and find accuracy of models by verification on test data

Data Collection

Data were collected via REST API of SpaceX information (utilization of BeautifulSoup package)

The information consists of

- Rocket basic information: Flight Number, Booster Version, Payload, Orbit
- Rocket launch: Date, Orbit, Location of launch site, etc
- Stage one reuse success results

Data Collection – SpaceX API

#1 Get response from REST API

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

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

#2 Transform data using Pandas package

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

#3 Call relevant function & Create Data frame

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

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

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

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

```
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}
```

#4 Filter needed data to further analysis

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
```

IBM Watson

[Link to full code]

Data Collection - Scraping

#1 Get response from HTML

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

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

#2 Create BeautifulSoup object & Get columns

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

```
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
for column_name in first_launch_table.find_all('th'):
    if column_name != "" and len(column_name) >0:
        column_names.append(column_name.text.strip())
    else:
        pass
```

#3 Make dictionary for data frame

```
# Remove an irrelvant column
del launch_dict['Date andtime (UTC)']

# 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']=[]
```

#4 Append data and define data frame

```
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
```

```
# df=pd.DataFrame(launch_dict,orient='column')
df = pd.DataFrame(dict([ (k,pd.Series(v)) for k,v in launch_dict.items() ] ))
```

IBM Watson

[Link to full code]()

Data Wrangling

Introduction

Data shows details of parameter resulting in both successful and failed landing of each launch. True oceans mission is an example of successful landing on the ocean while False oceans mission resulted otherwise.

Furthermore, there are some terms used to describe specific events as below

True RTLS: successful landing on a ground pad

False RTLS : unsuccessful landing on a ground pad

True ASDS : successful landing on a drone ship

False ASDS : unsuccessful landing on a drone ship

Data would be transformed into binary system i.e., 0 and 1 to represent unsuccessful and successful landing respectively

#1 Calculate no. of launch

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

#2 Calculate no. of occurrence of each orbit

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

#3 Calculate no. of occurrence of outcome by orbit type

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
```

#4 Create label columns and calculate average landing success rate

```
landing_class = []
list = df['Outcome'].to_list()
for i in list :
    if i in bad_outcomes :
        landing_class.append(0)
    else:
        landing_class.append(1)
```

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

EDA with Data Visualization

Scatter plot:
- Flight number vs. Payload mass
- Flight number vs. Launch site
- Payload vs. Launch site
- Orbit vs. Flight number
- Payload mass vs. Orbit type
- Orbit vs. Payload mass

Bar chart
- Mean vs. Orbit

Line graph
- Success rate by year

Scatter plot: show effect of one variable by another. It is suitable for large dataset

Bar chart: easy to compare values by category or continuous dependent variable

Line graph: visualize trend data and make predictions

EDA with SQL

- **SQL queries to respond to below questions**

1. Display the names of the unique launch sites in the space mission
2. *Display 5 records where launch sites begin with the string 'CCA'*
3. *Display the total payload mass carried by boosters launched by NASA (CRS)*
4. *Display average payload mass carried by booster version F9 v1.1*
5. *List the date when the first successful landing outcome in ground pad was acheived.*
6. *List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000*
7. *List the total number of successful and failure mission outcomes*
8. *List the names of the booster_versions which have carried the maximum payload mass. Use a subquery*
9. *List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015*
10. *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*

IBM Watson

Link to full code

Build an Interactive Map with Folium

Objects created for a Folium map

- Markers show all sites
- Markers show success/failed launches
- Distance line show

Also, an outcome of each launch is named as binary 0 and 1 representing failure and success landing respectively.
Distance was then calculated

Some findings:
Q: Are launch sites in close proximity to railways ? **A**: No.
Q: Are launch sites in close proximity to highways ? **A**: No.
Q: Do launch sites keep certain distance away from cities ? **A:** Yes.

IBM Watson

Link to full code

Build a Dashboard with Plotly Dash

Dashboard was created by Dash and contains

Charts

- Pie chart
  - Total launches for each site
  - Relative of multiple classes of data
  - Quantity shown as size of each circle
- Scatter plot
  - Outcome vs Payload mass by booster version

Predictive Analysis (Classification)

Build Model	• Load & Transform data by Pandas , Numpy packages • Split train and test data using Scikit-learn package • Implement algorithm for each classification model
Evaluation	• Verify accuracy of each model using f1 score, jaccard score, and confusion matrix.
Improvement	• Tuning related parameters
Model with best fit	• Employ the best model to predict data

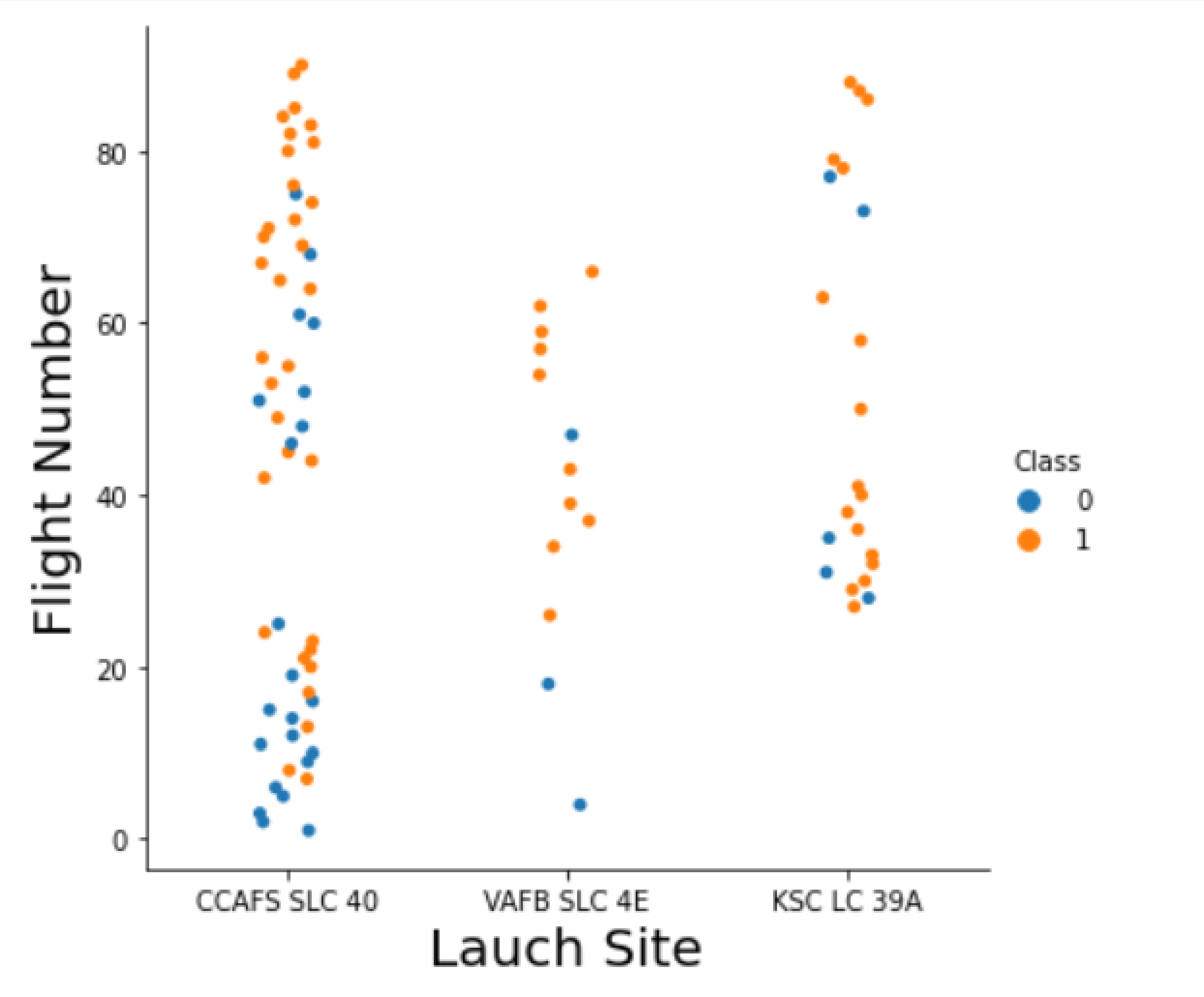
Results

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

Section 2

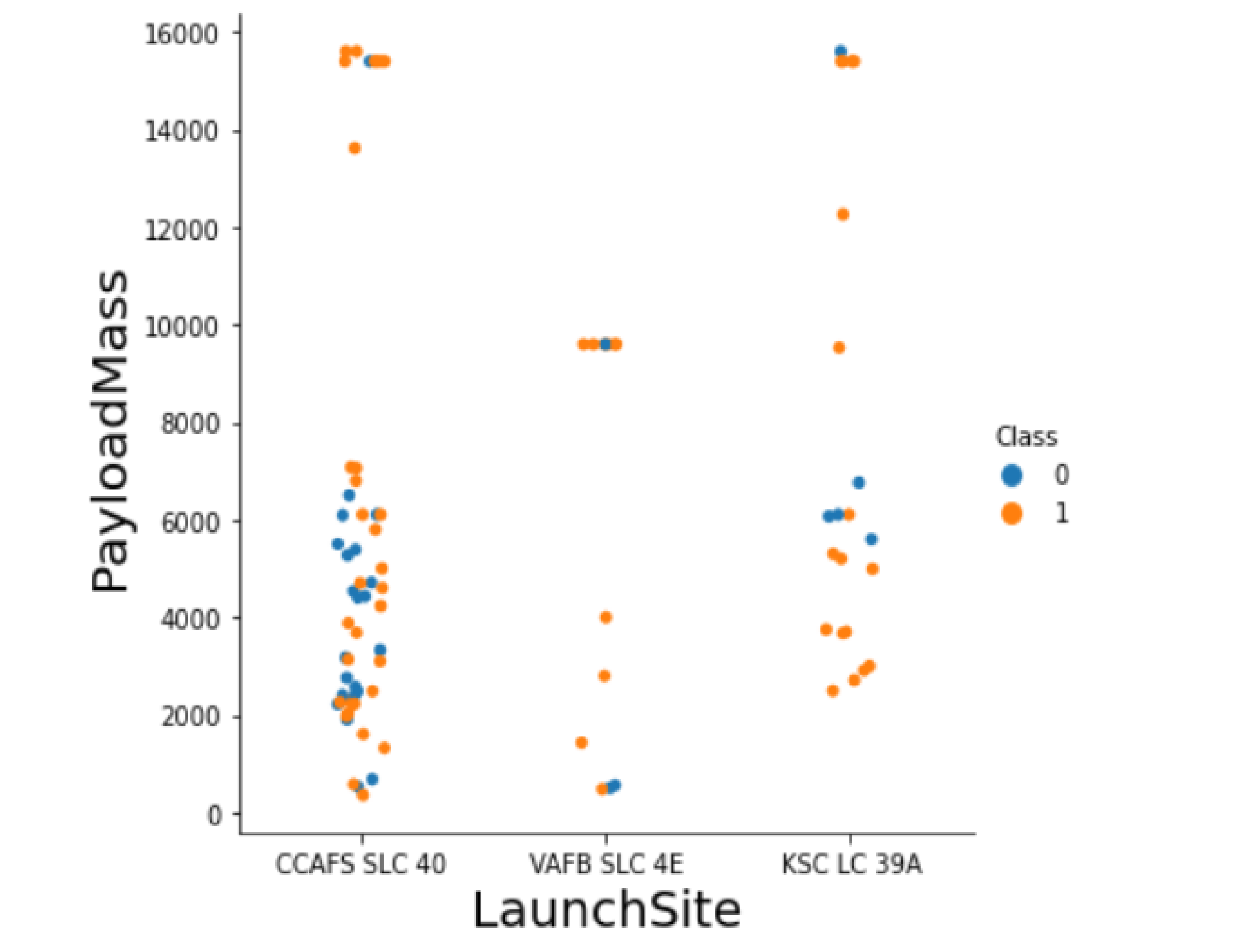
Insights drawn from EDA

Flight Number vs. Launch Site



With higher launches, successful rate is higher it would get

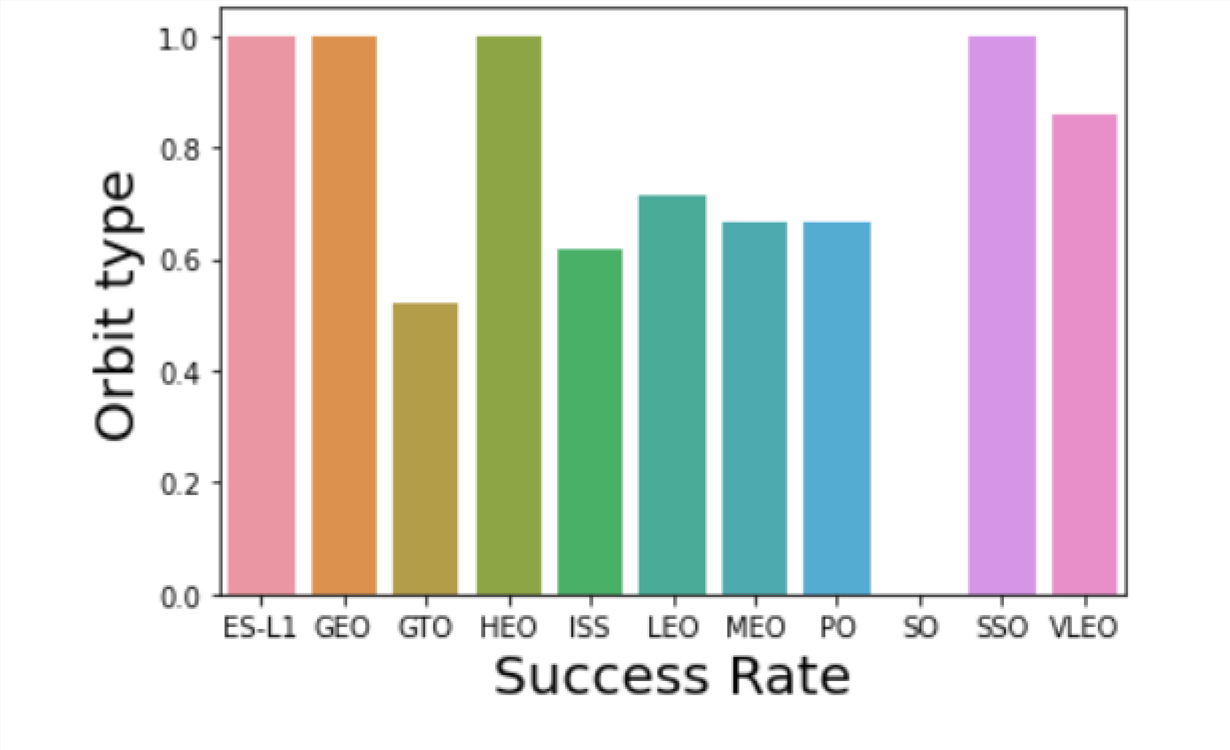
Payload vs. Launch Site



Higher payload shows the most success rate compared to lower ones.

Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).

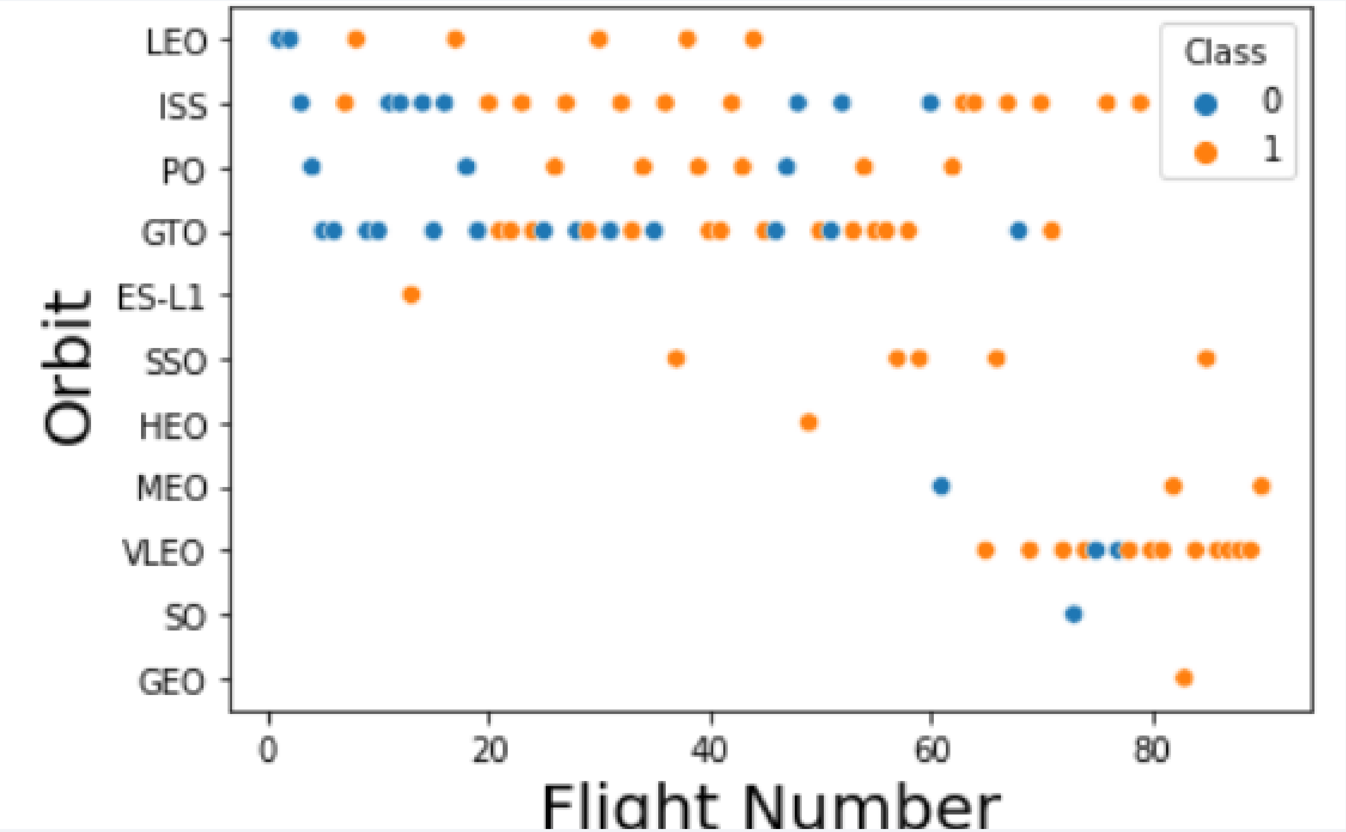
Success Rate vs. Orbit Type



Most success rate comes from orbit

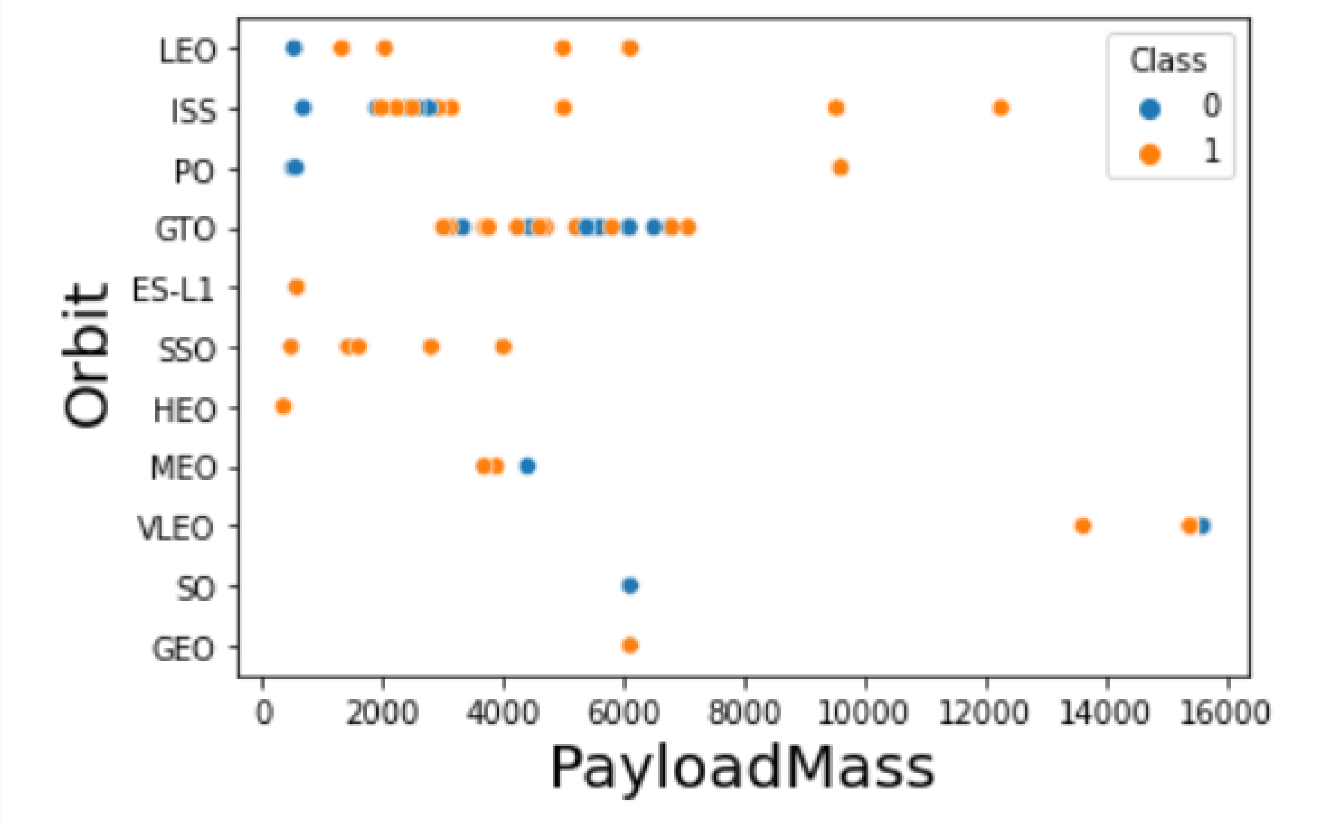
- ES-L1
- GEO
- HEO
- SSO

Flight Number vs. Orbit Type



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

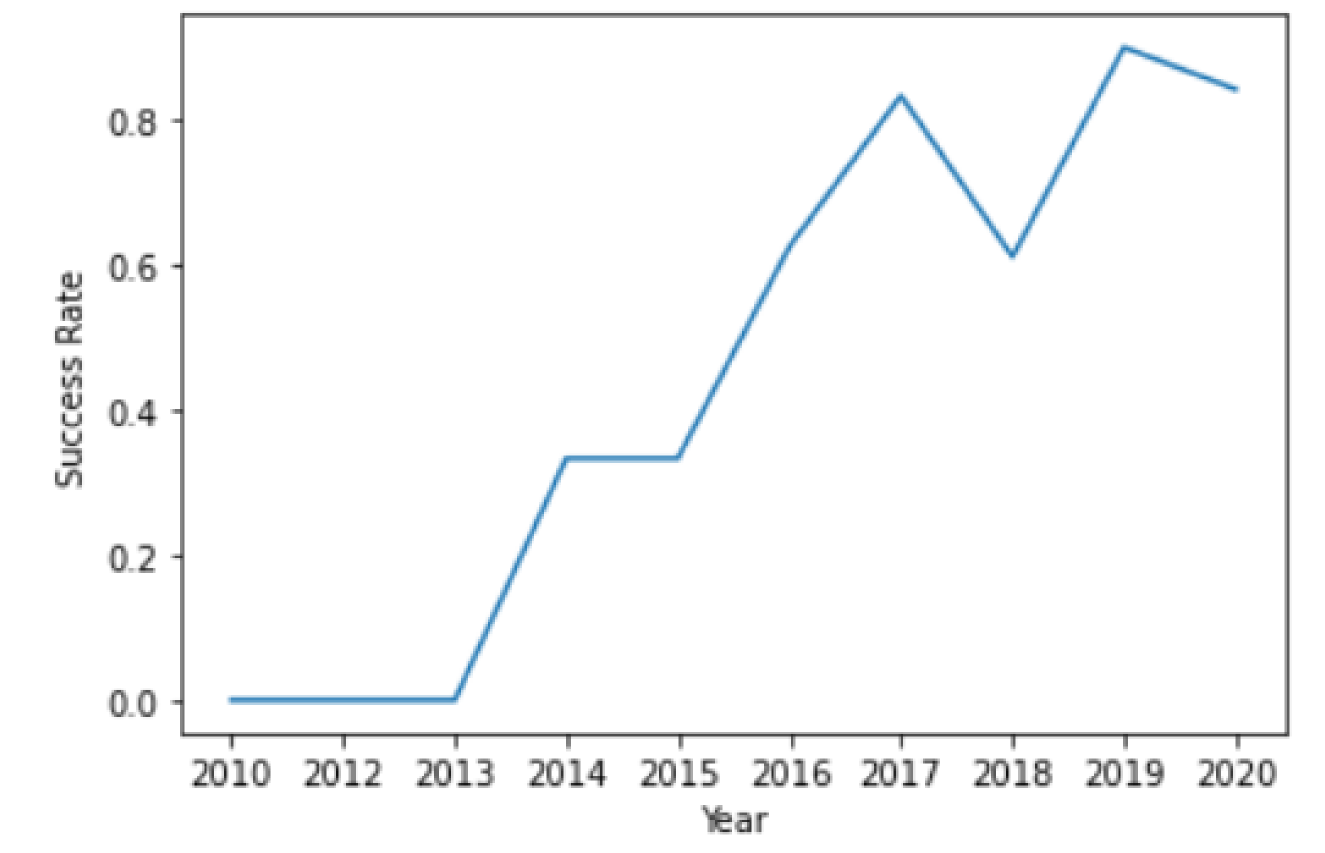


With heavy payloads the successful landing or positive landing rate are more for Polar,

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.

Launch Success Yearly Trend



We can see as years went by, the higher success rate it would get by average.

All Launch Site Names

```
%%sql

select unique(launch_site) from SPACEXTBL2;
```

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

Explanation

Apply “**Unique**” command to get distinct list from column “**launch_site**” in table **SPACEXTBL2**

Launch Site Names Begin with 'CCA'

```
%%sql

select * from SPACEXTBL2
where launch_site like '%CCA%'
;
```

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

Explanation

Apply condition **like '%CCA%'** to get all rows that has a word "CCA" containing in column '"**launch_site**" from **SPACEXTBL2** table

Total Payload Mass

```
%%sql

select sum(payload_mass__kg_) as "Total Payload Mass (kg)" from SPACEXTBL2
where customer = 'NASA (CRS)'
;
```

Total Payload Mass (kg)
45596

Explanation

Calculate total payload mass for **NASA (CBS)**

Average Payload Mass by F9 v1.1

```
%%sql

select sum(payload_mass__kg_) as "Total Payload Mass (kg)" from SPACEXTBL2
where booster_version = 'F9 v1.1'
;
```

Total Payload Mass (kg)
14642

Explanation

Calculate total payload mass for **booster version F9 v1.1**

First Successful Ground Landing Date

```
%%sql

select min(DATE) as "First date to achieve landing on ground" from SPACEXTBL2
where landing__outcome = 'Success (ground pad)'
;
```

First date to achieve landing on ground
2015-12-22

Explanation

Apply **min(Date)** to get the first success date with condition set as ‘**landing__outcome = ‘Success (ground pad)’**

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql

select unique(booster_version), payload_mass__kg_ from SPACEXTBL2
where landing__outcome = 'Success (drone ship)'
and payload_mass__kg_ > 4000
and payload_mass__kg_ < 6000
order by payload_mass__kg_ DESC
;
```

booster_version	payload_mass__kg_
F9 FT B1021.2	5300
F9 FT B1031.2	5200
F9 FT B1022	4696
F9 FT B1026	4600

Explanation

Set condition with payload for selected range, Extract desired columns from database and order from **highest payload** shown in table

Total Number of Successful and Failure Mission Outcomes

```
%%sql

select mission_outcome, count(mission_outcome) as "Count" from SPACEXTBL2
group by mission_outcome
;
```

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

Explanation

Get total success/failure by applying '**count**' command. Also, **group data by outcome** to see total number of each.

Boosters Carried Maximum Payload

```
%%sql

select unique(booster_version), payload_mass__kg_ from SPACEXTBL2
where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTBL2)
order by booster_version ASC
;
```

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

Explanation

Get **maximum** payload mass **by adding sub-query** into condition ‘where’. We can then extract from the table.

2015 Launch Records

```
%%sql

select landing__outcome, booster_version, launch_site, Year(Date) as "Year" from SPACEXTBL2
where Year(Date) = 2015
and landing__outcome LIKE '%Failure%'
;
```

landing__outcome	booster_version	launch_site	Year
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015

Explanation

When **set condition year = 2015**, we can also set landing out **contains 'failure'** by using "**like**" syntax to find data in the columns and show in table

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

```
%%sql

select landing__outcome, count(landing__outcome) as "Count" from SPACEXTBL2
where Date >= '2010-06-04'
and Date <= '2017-03-20'
group by landing__outcome
;
```

landing__outcome	Count
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

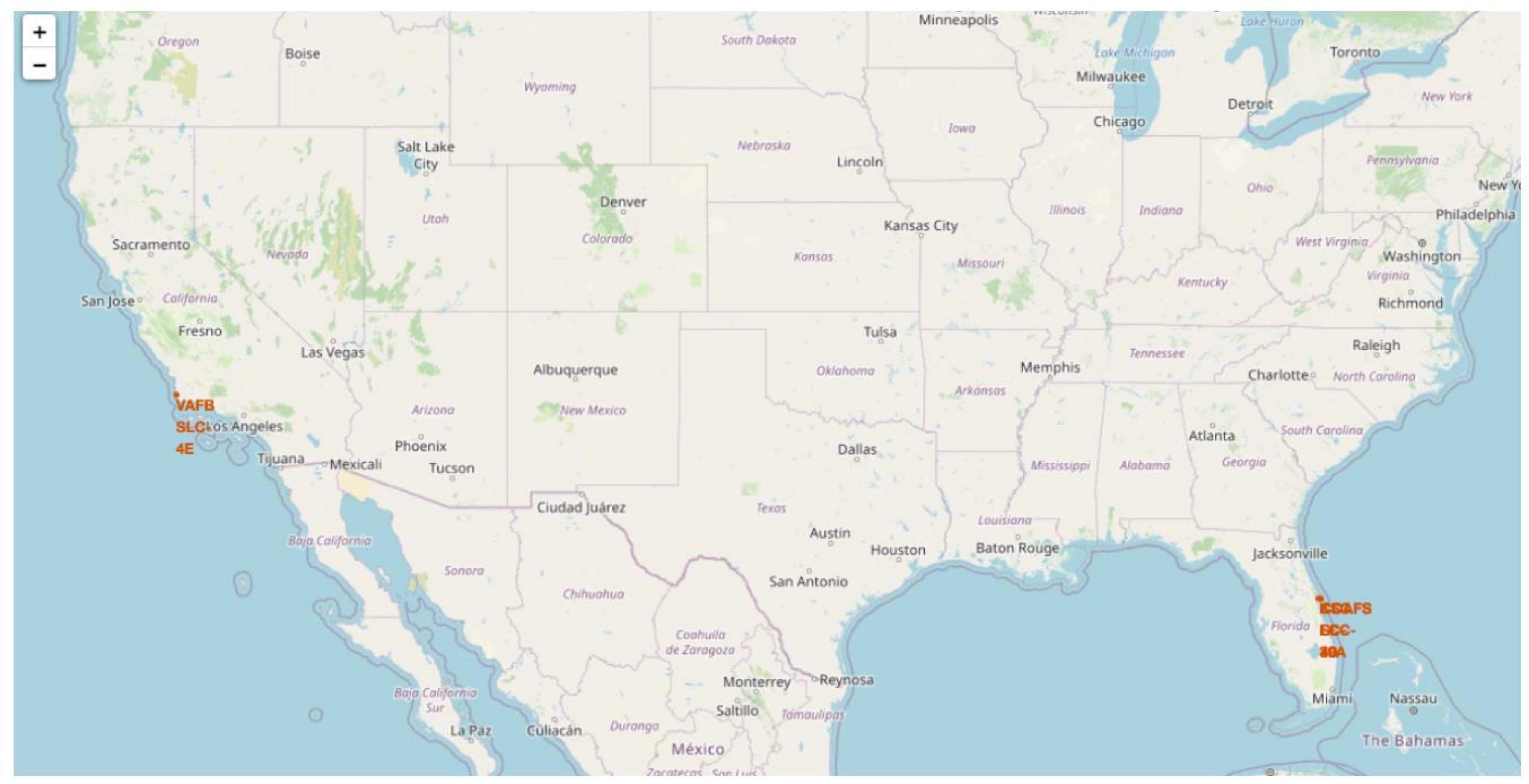
Explanation

Select '**count**' syntax to get total landing outcomes
When set condition to **desired years**, then **grouping** by the outcome to see results in table shown

Section 4

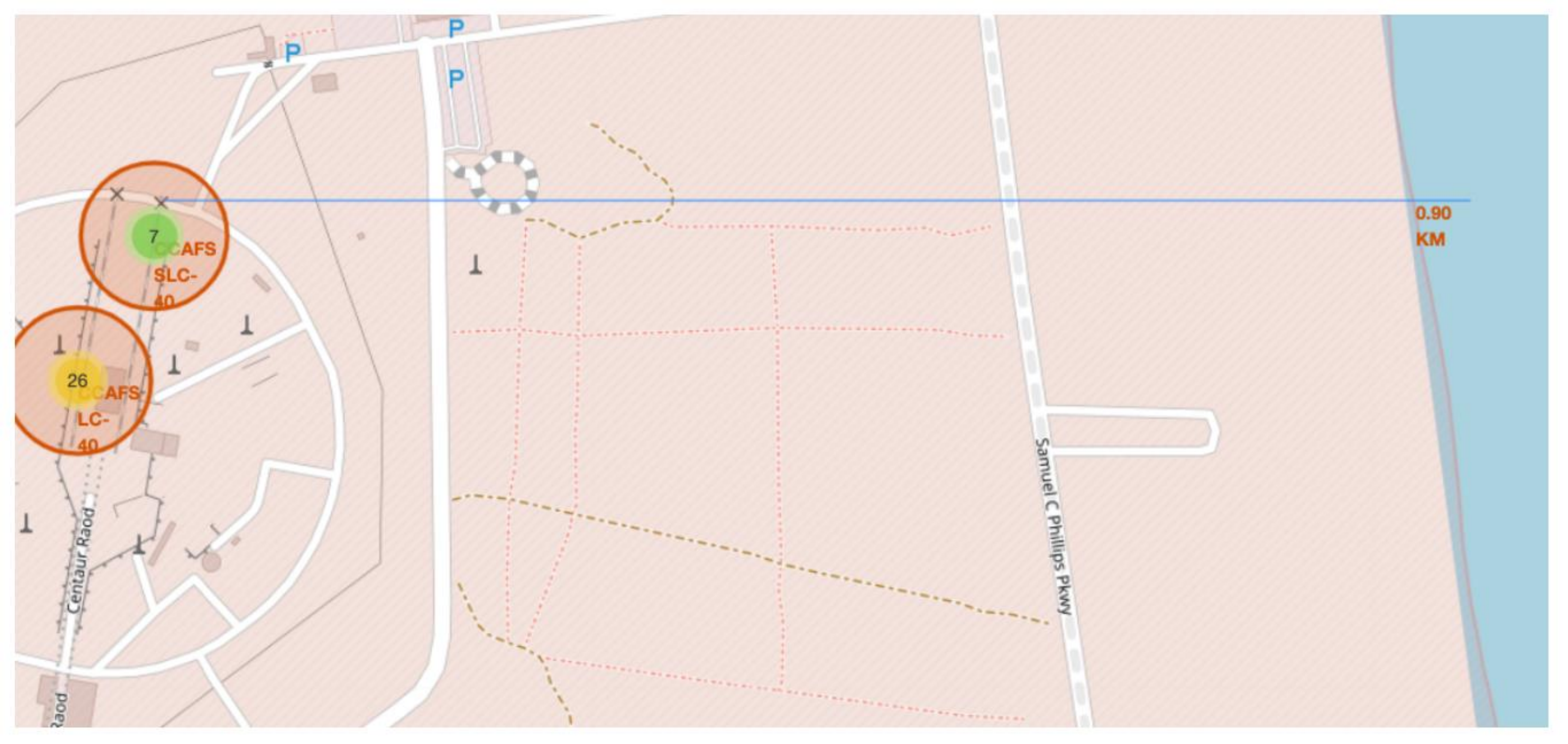
Launch Sites Proximities Analysis

Launch sites of rocket



Success landing of each site (Green = success)

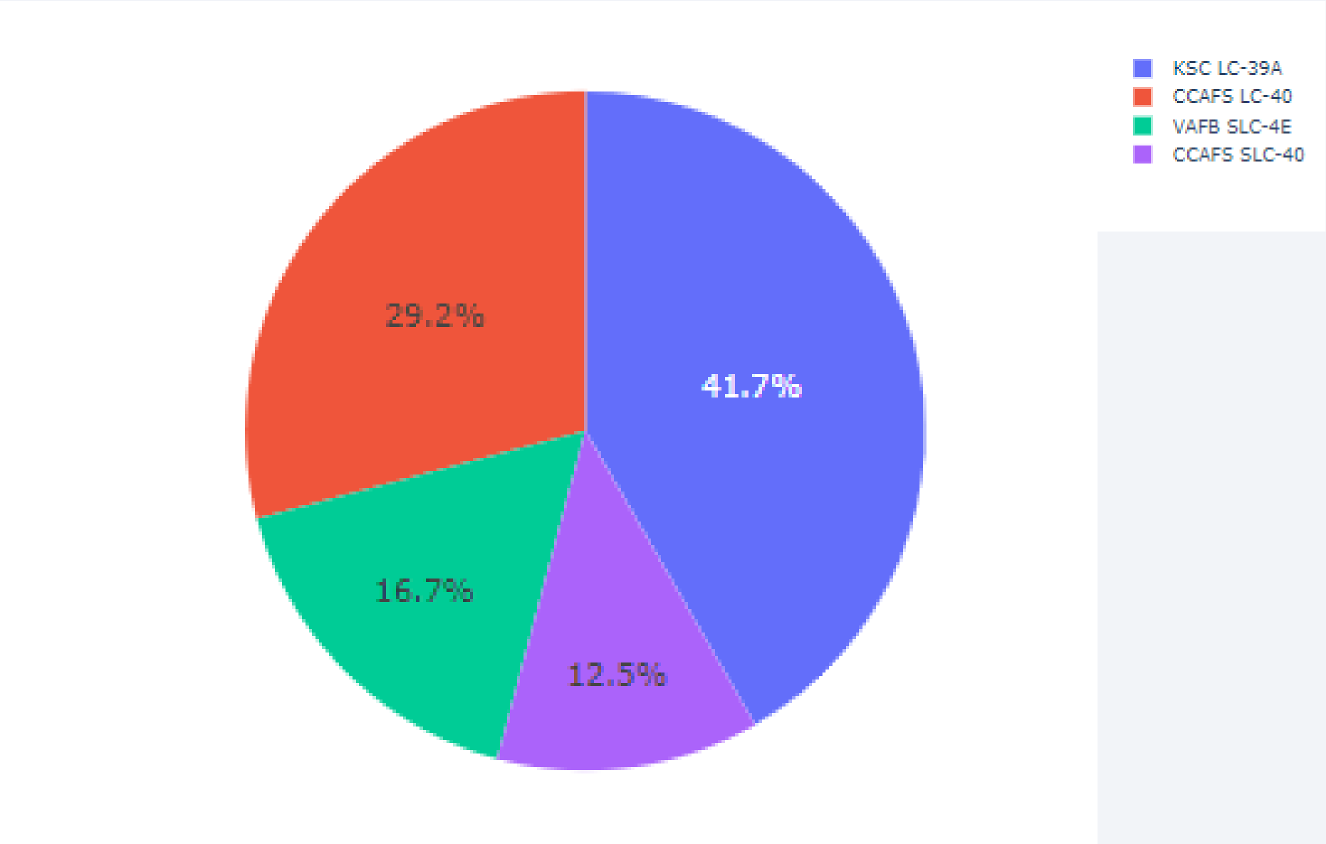
Distance from site to nearest coast



Section 5

Build a Dashboard with Plotly Dash

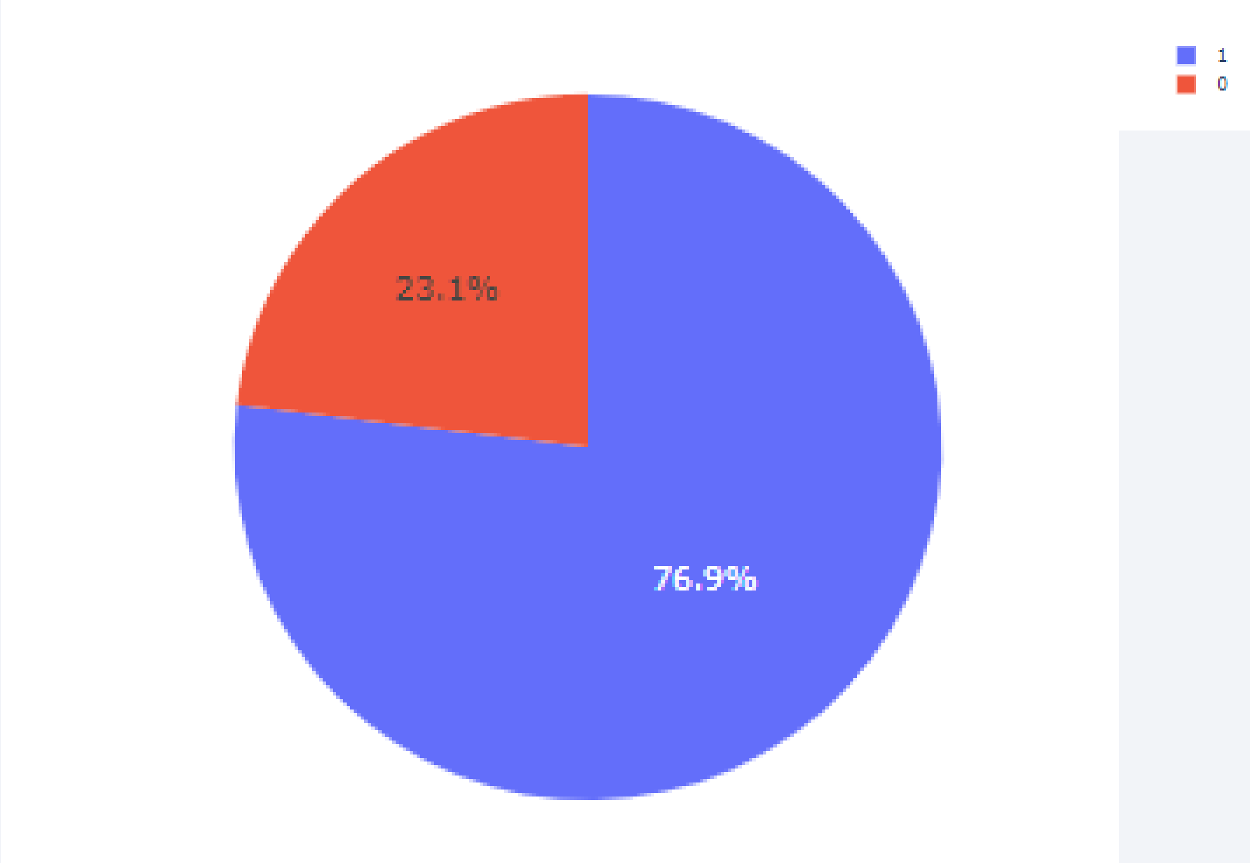
Total launches by sites



Explanation

Site KSC LC-39A showed the most success percentage.

Launch site with highest success

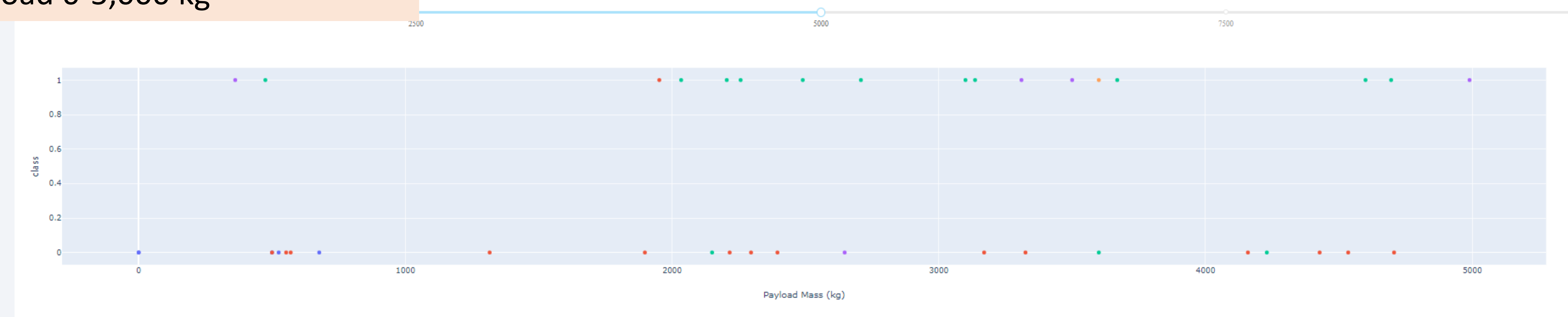


Explanation

Site KSC LC-39A showed the most success at 76.9%

Payload vs. Launch Outcome

Payload 0-5,000 kg



Payload 5,000-10,000 kg

When Payload mass is low, it is likely to have success landing outcomes

Section 6

Predictive Analysis (Classification)

Classification Accuracy

KNN, Logistic Regression, and Tree classification model were employed.
After tuned, we found the best model to present is

Tree algorithm
(see code as below)

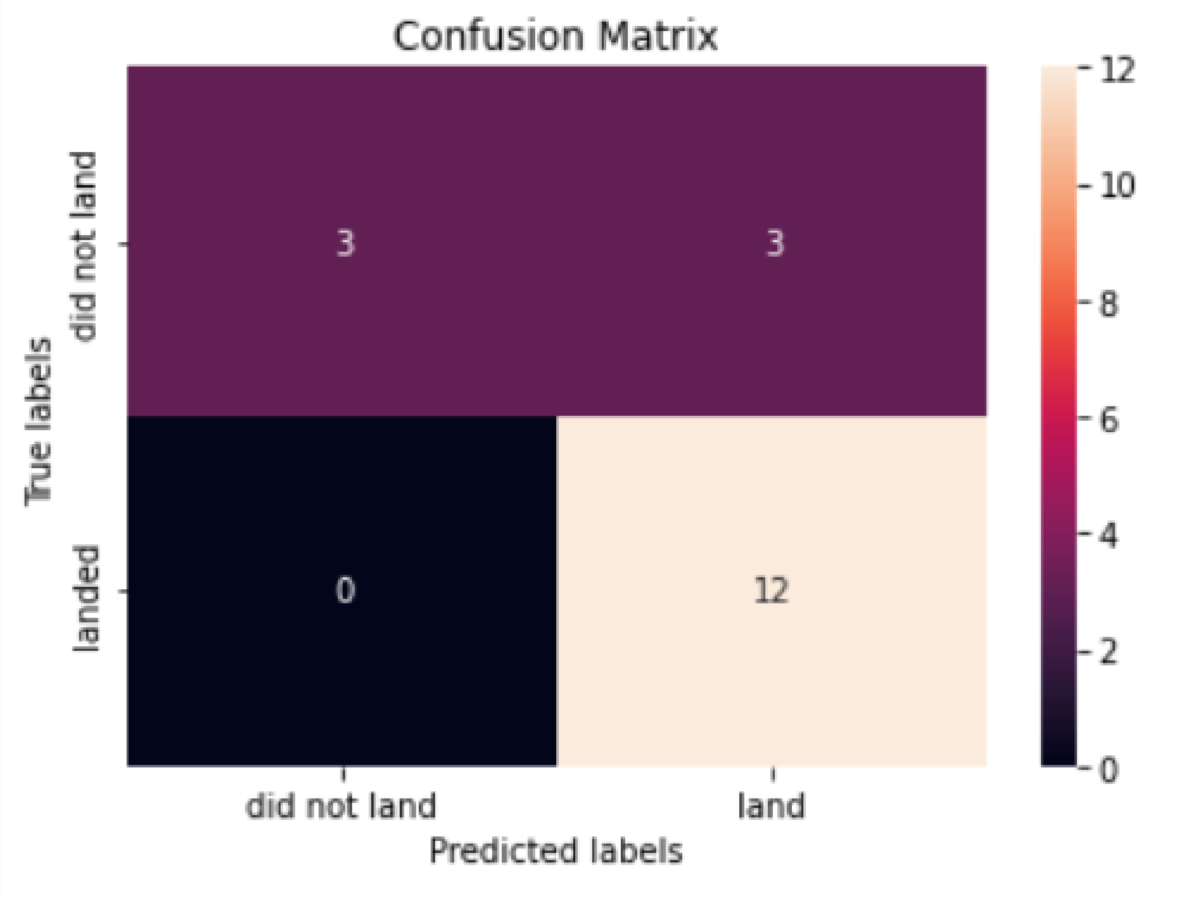
```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params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

```
Best Algorithm is Tree with a score of 0.8767857142857143
Best Params is : {'criterion': 'gini', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'random'}
```

IBM Watson

[Link to full code]

Confusion Matrix



Explanation

Tree confusion matrix

Confusion matrix shows major false prediction is “true positive” where the model predicts failure landings to be successful.

Conclusions

Successful rate go higher as time go by and number of rocket launches

Lower payload mass & specific orbit type show the better success rate of an outcome

Launch site: KSC LC-39A performs the best succuss rate of landing

Tree Classifier is considered as the best algorithm to predict the outcomes with accuracy around 84%

Thank you!