IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ziyu Li
Jan 2, 2025

Outline

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

Executive Summary

- The project follows the data science methodologies:
  - Data collection
  - Data wrangling
  - Exploratory data analysis (EDA)
    - Interactive Visual Analytics
    - SQL
  - Model development (predictive analysis) and evaluation
- Summary of all results
  - EDA result
  - Predictive analytics result

Introduction

- Background
    - 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 to give insights on the bid against SpaceX for a rocket launch.
    - In this project, we would like to predict if the Falcon 9 first stage will land successfully and thus help predict the cost for each launch.
- Problems
    - What factors determint if the rocket can land successfully?
    - The relationships between the features and the success of the landing.
    - Can we predict if the first stage will land successfully?

Section 1

Methodology

Methodology

Executive Summary

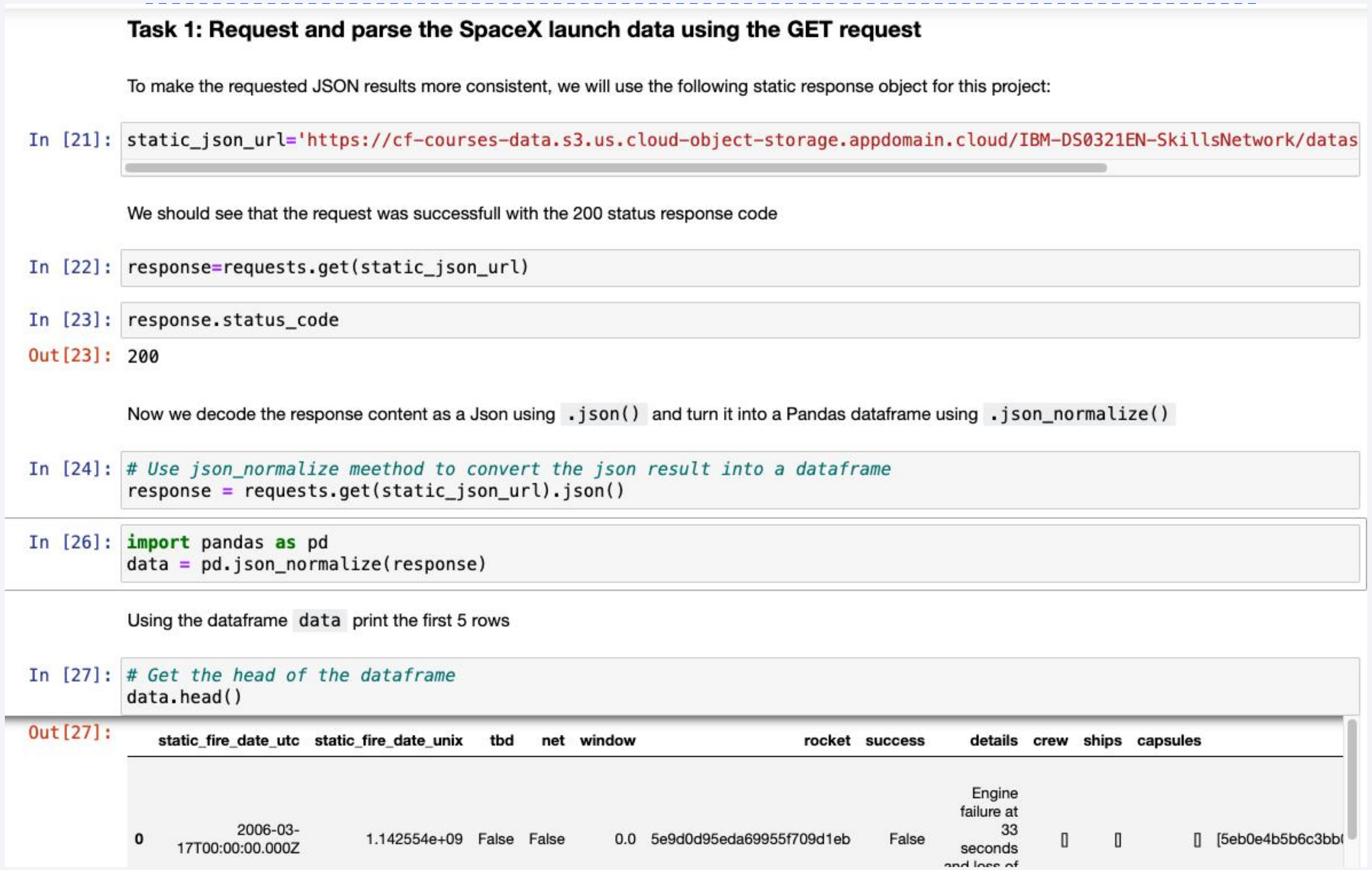
- Data collection methodology:
- Perform data wrangling
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

- The data is collected from the **SpaceX API**
  - Information includes the launches with the IDs given for each launch, and specifically, rocket, payloads, launchpad, cores, and etc
  - Content transformed from JSON to pandas dataframe
    - using .json() to transform the raw JSON content
    - using .json_normalize() to transform it into the pandas dataframe
- The data can also be scraped from the web (wikipedia)
  - using BeautifulSoup to parse the website
  - identifying the HTML table with .find_all() function

Data Collection – SpaceX API

- Use requests to obtain the JSON results
  - response = requests.get(url).json()
- Transform the data into pandas DataFrame
  - df = pd.json_normalize(response)



Task 1: Request and parse the SpaceX launch data using the GET request

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

```
In [21]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datas
```

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

```
In [22]: response=requests.get(static_json_url)

In [23]: response.status_code

Out[23]: 200
```

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

```
In [24]: # Use json_normalize meethod to convert the json result into a dataframe
         response = requests.get(static_json_url).json()

In [26]: import pandas as pd
         data = pd.json_normalize(response)
```

Using the dataframe `data` print the first 5 rows

```
In [27]: # Get the head of the dataframe
         data.head()
```

Out[27]:

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of	[]	[]	[] [5eb0e4b5b6c3bb(

https://github.com/zLizy/DataScienceAssignment/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
  - soup = BeautifulSoup(response,`html.parser`)
- We then identify the rows in the coresponding table
- We store the information in a dictionary and then convert it into a dataframe

TASK 1: Request the Falcon9 Launch Wiki page from its URL

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

In [5]:
```python
# 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`

In [8]:
```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
content = response.text
print(content[:100])
```

```
<!DOCTYPE html>
<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-la
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [12]:
```python
# Use soup.title attribute
soup = BeautifulSoup(content, 'html.parser')
title = soup.find('title')
title
```

Out[12]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

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

```
['Flight No.',
 'Date and time ( )',
 'Launch site',
 'Payload',
 'Payload mass',
 'Orbit',
 'Customer',
 'Launch outcome']
```

https://github.com/zLizy/DataScienceAssignment/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

- We first identy the percentage of missing values in each attribute
- We identify the data types, as well as the value counts for each column
- We create the landing outcome label (binary) based on the attribute `Outcome` in string

Identify and calculate the percentage

```
df.isnull().sum()/len(df)*100
```

```
FlightNumber       0.000000
Date               0.000000
BoosterVersion     0.000000
PayloadMass        0.000000
Orbit              0.000000
LaunchSite         0.000000
Outcome            0.000000
Flights            0.000000
GridFins           0.000000
Reused             0.000000
Legs               0.000000
LandingPad        28.888889
Block              0.000000
ReusedCount        0.000000
Serial             0.000000
Longitude          0.000000
Latitude           0.000000
dtype: float64
```

```
df.dtypes
```

```
FlightNumber        int64
Date               object
BoosterVersion     object
PayloadMass       float64
Orbit              object
LaunchSite         object
Outcome            object
Flights             int64
GridFins             bool
Reused               bool
Legs                 bool
LandingPad         object
Block             float64
ReusedCount         int64
Serial             object
Longitude         float64
Latitude          float64
dtype: object
```

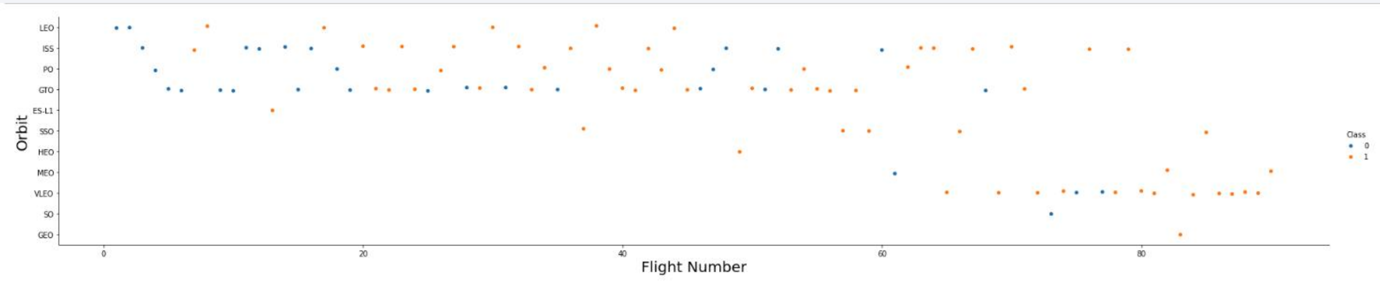
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = ~df['Outcome'].isin(bad_outcomes).values
landing_class =  np.array(landing_class, dtype=int)
landing_class
```

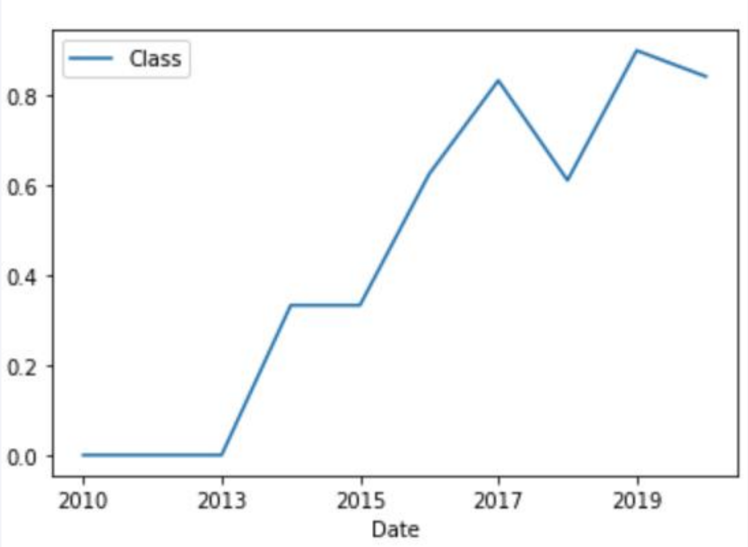
```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1])
```


https://github.com/zLizy/DataScienceAssignment/blob/main/jupyter-labs-webscraping.ipynb

EDA with Data Visualization

- We use scatter plot to explore the relationship between various attributes and the launch site as well as the orbit type
- We also visulize with a line chart to present the launch success yearly trend





https://github.com/zLizy/DataScienceAssignment/blob/main/jupyter-labs-webscraping.ipynb

EDA with SQL

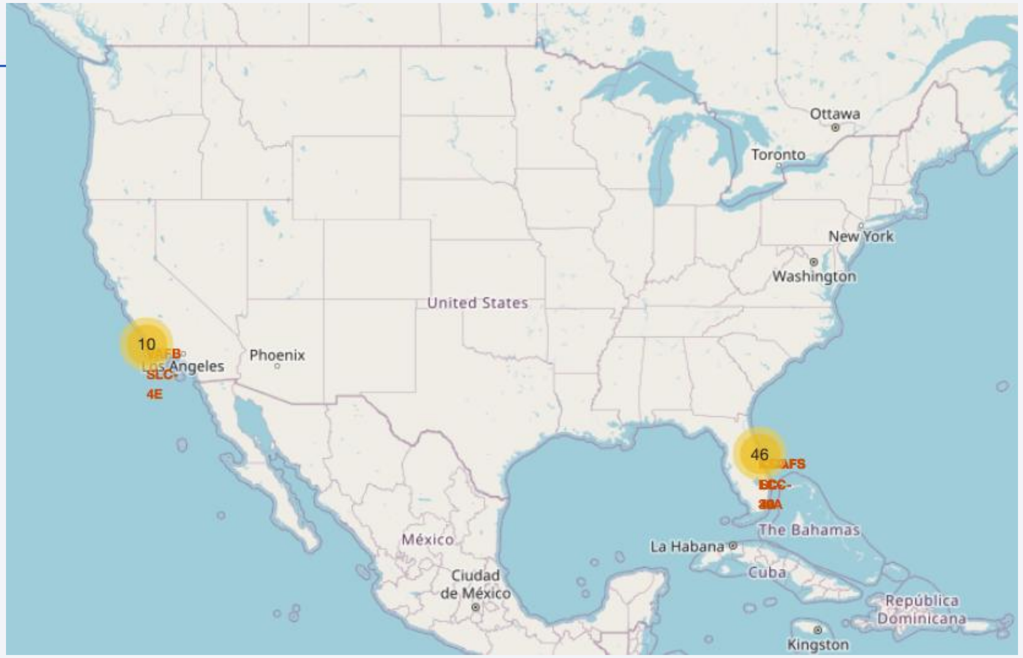
- We loaded the SpaceX dataset into a PostgreSQL database
- We applied SQL to get insight from the data. We answer the following queries to explore

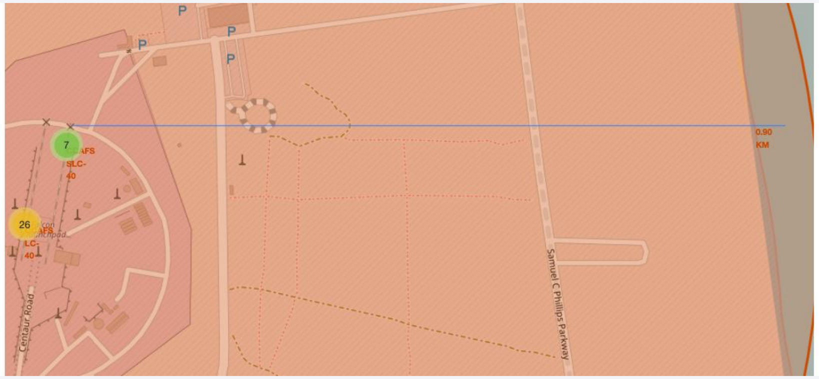
1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string 'CCA'
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1
5. List the date when the first succesful 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 records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months 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.

https://github.com/zLizy/DataScienceAssignment/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1, 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities.
- We answered some questions for instance:
    - Are launch sites near railways, highways and coastlines.
    - Do launch sites keep certain distance away from cities.

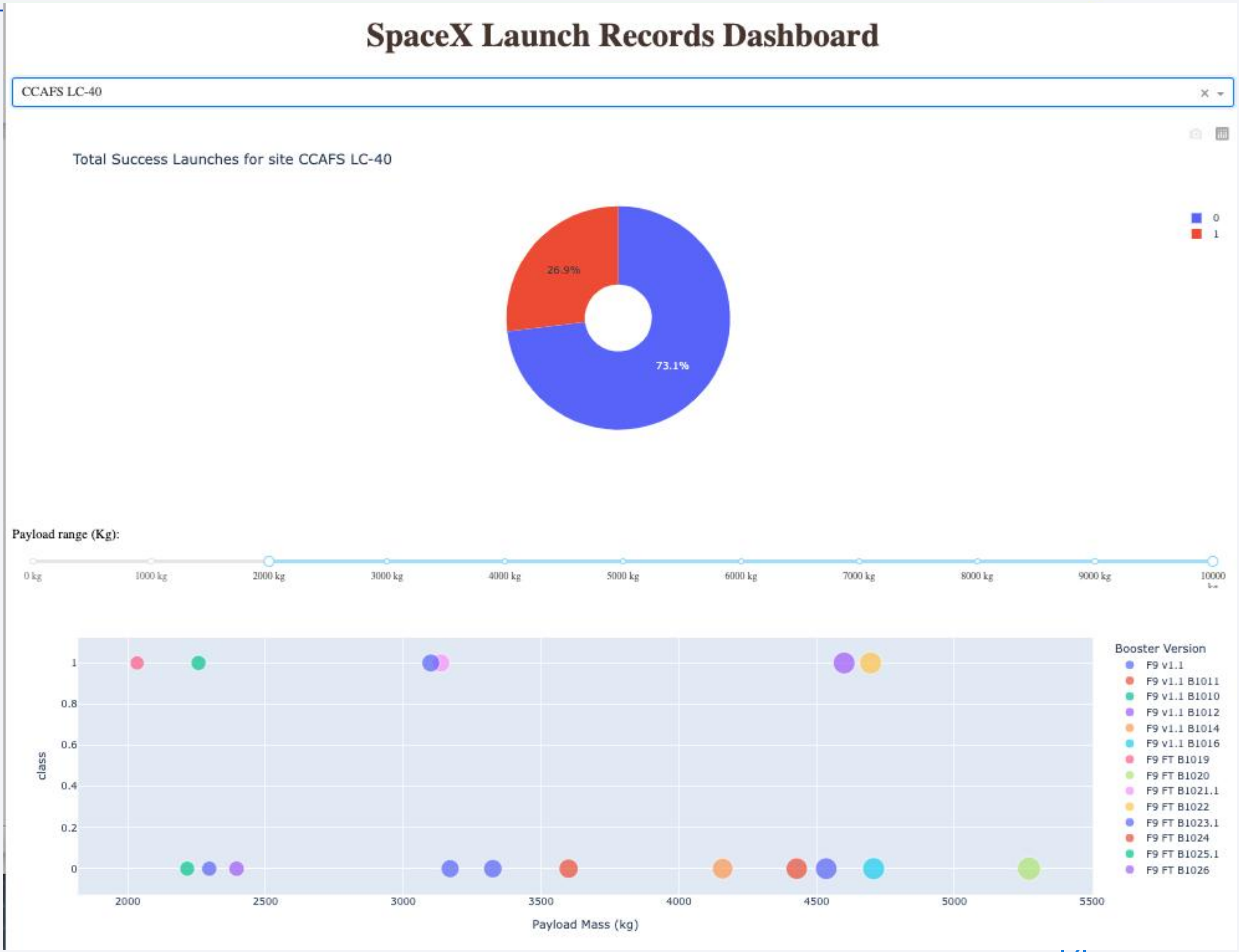




https://github.com/zLizy/DataScienceAssignment/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter plot showing the relationship between the Payload Mass (Kg) and the success rate for the different booster version



https://github.com/zLizy/DataScienceAssignment/blob/main/Dashboard-with-Ploty-Dash.py

Predictive Analysis (Classification)

- We train various classification models, such as Logistic Regression, Support Vector Machine, Decision Tree.
- We adopt a grid search training method to explore the best hyperparameters for each model
- We use confusion matrix and accuracy to evaluate the performance of the models
- Logistic Regression, SVM, and KNN receives the same accuracy on the test set, achieving 83.3% of accuracy

```
In [30]: parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                       'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                       'p': [1,2]}

         KNN = KNeighborsClassifier()

In [31]: knn_cv = GridSearchCV(KNN,parameters,cv=10)
         knn_cv.fit(X_train,Y_train)

Out[31]: ▸ GridSearchCV
         ▸ best_estimator_: KNeighborsClassifier
             ▸ KNeighborsClassifier

In [32]: print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
         print("accuracy :",knn_cv.best_score_)

         tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neig
         hbors': 10, 'p': 1}
         accuracy : 0.8482142857142858
```

TASK 11

Calculate the accuracy of knn_cv on the test data using the method `score` :

```
In [33]: knn_cv.score(X_test,Y_test)

Out[33]: 0.8333333333333334
```


https://github.com/zLizy/DataScienceAssignment/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

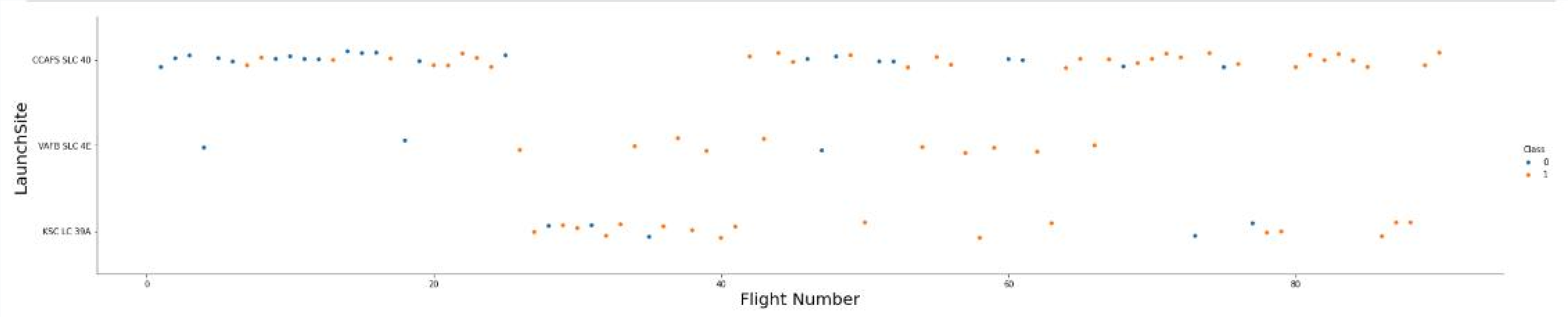
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- Scatter plot of Flight Number vs. Launch Site

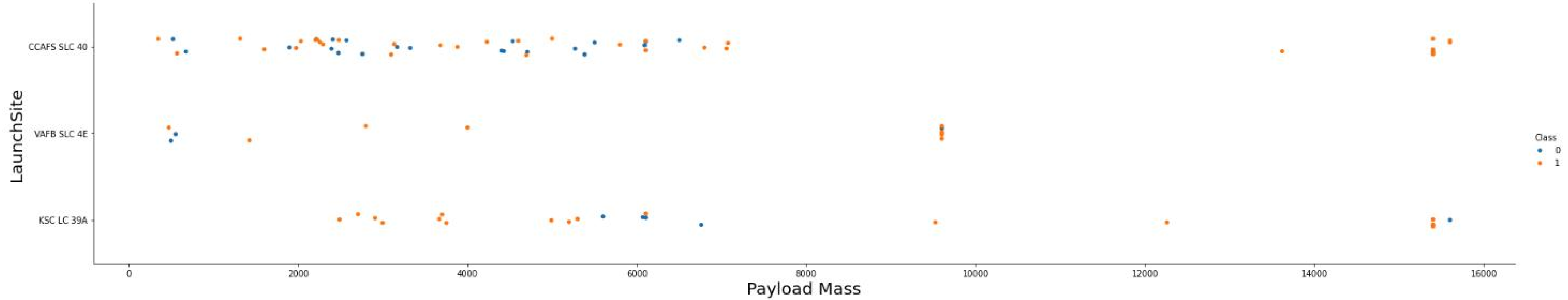


- Show the screenshot of the scatter plot with explanations

For all the launch sites, the sccuess rate is higher with a higher flight number, indicating they have improved the techniques of landing the roket with more try-out.

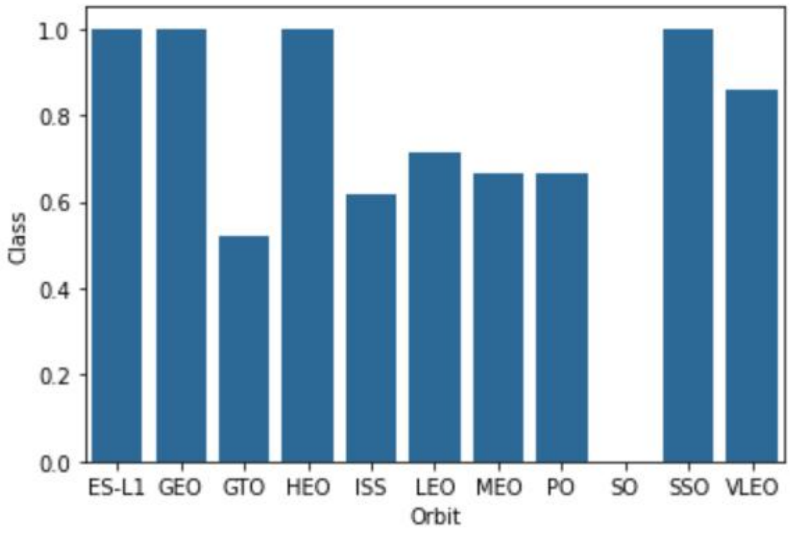
Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site



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

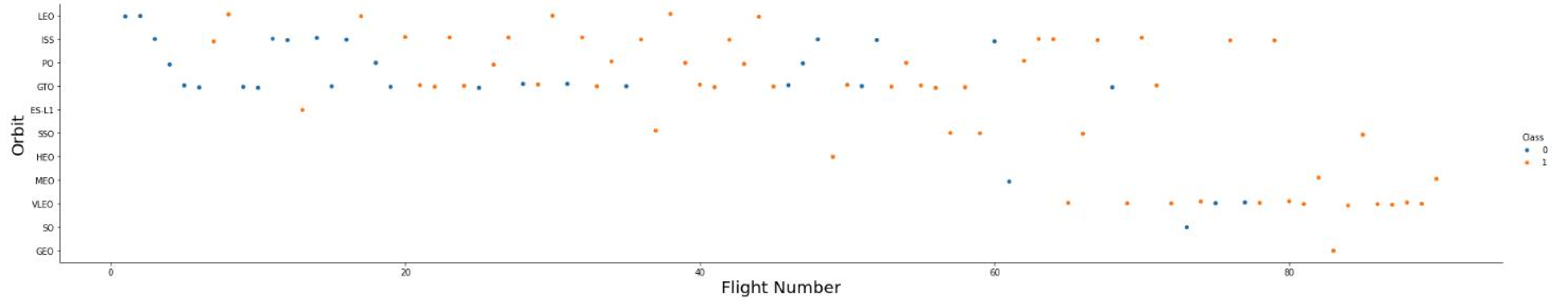
Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type
- 

ES-L1, GEO, HEO and SSO have 100% of success rate, followed by VLEO with a high success rate around 80%. For the other, the success rate is medium, ranging between 50%-70%.

Flight Number vs. Orbit Type

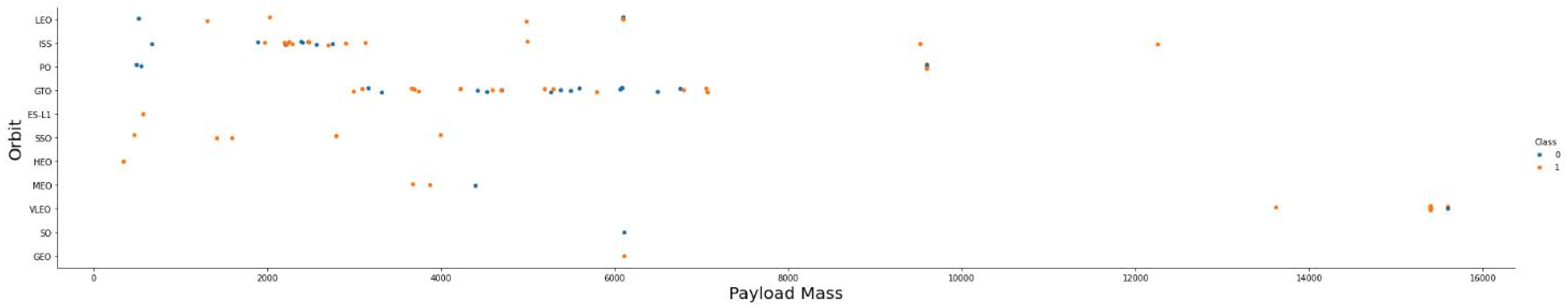
- Show a scatter point of Flight number vs. Orbit type



You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type

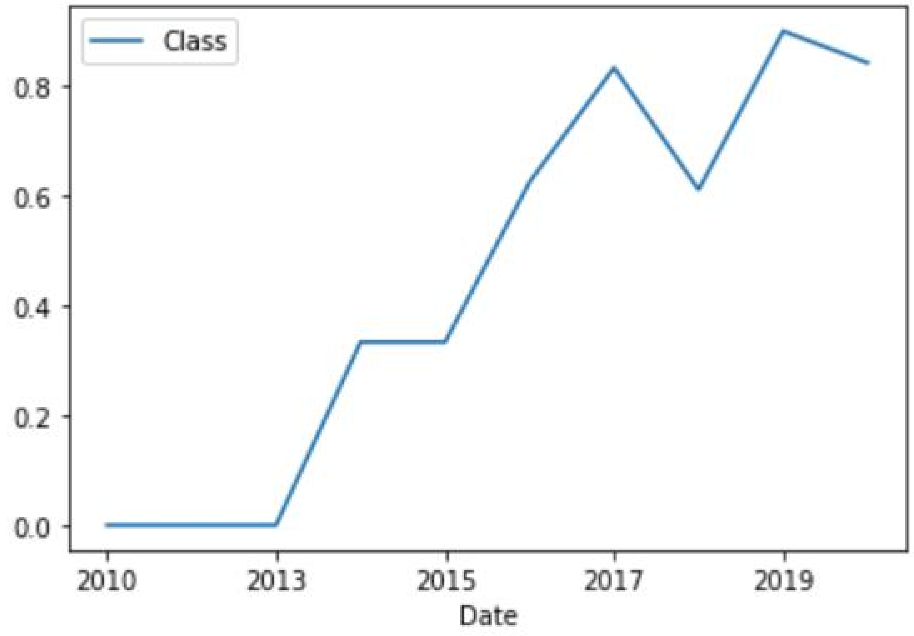


With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend

- Show a line chart of yearly average success rate



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

All Launch Site Names

- Find the names of the unique launch sites

```
%sql select distinct Launch_Site from SPACEXTABLE
```

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

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- We use LIKE and `%` syntax to identify the strings begin with `CCA`

```
13]:  %sql select Launch_Site from SPACEXTABLE where Launch_Site like 'CCA%' limit 5

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

13]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- The total payload is 45596 kg

```
%%sql
select SUM(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer = 'NASA (CRS)';
```

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

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_),Booster_Version from SPACEXTABLE where Booster_Version like 'F9 v1.1%'
```

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

avg(PAYLOAD_MASS__KG_)	Booster_Version
2534.6666666666665	F9 v1.1 B1003

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- We use min(Date) to identify the earliest date for the launch

```
%sql select min(Date) from SPACEXTABLE where Landing_Outcome = 'Success (ground pad)'
```

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

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
%%sql
select Booster_Version, Landing_Outcome, PAYLOAD_MASS__KG_ from SPACEXTABLE
where Landing_Outcome = 'Success (drone ship)'
and PAYLOAD_MASS__KG_ > 4000
and PAYLOAD_MASS__KG_ < 6000
```

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

Booster_Version	Landing_Outcome	PAYLOAD_MASS__KG_
F9 FT B1022	Success (drone ship)	4696
F9 FT B1026	Success (drone ship)	4600
F9 FT B1021.2	Success (drone ship)	5300
F9 FT B1031.2	Success (drone ship)	5200

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- We use COUNT() fuction to compute the number of mission outcomes

```
%sql select count(Mission_Outcome), Mission_Outcome from SPACEXTABLE group by Mission_Outcome
```

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

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- We use subquery to compute the maximum mass and then find the matches

```
%%sql
select distinct Booster_Version, PAYLOAD_MASS__KG_ from SPACEXTABLE
where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)
```

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

Booster_Version	PAYLOAD_MASS__KG_
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

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- We use substr() to identify the month and year

```
%%sql
select substr(Date,6,2) , Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE
where Landing_Outcome like 'Failure%' and substr(Date,0,5)='2015'
```

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

substr(Date,6,2)	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- 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

```
%%sql
select count(Landing_Outcome), Landing_Outcome, Date from SPACEXTABLE
where Date between '2010-06-04' and '2017-03-20' group by Landing_Outcome
order by count(Landing_Outcome) DESC
```

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

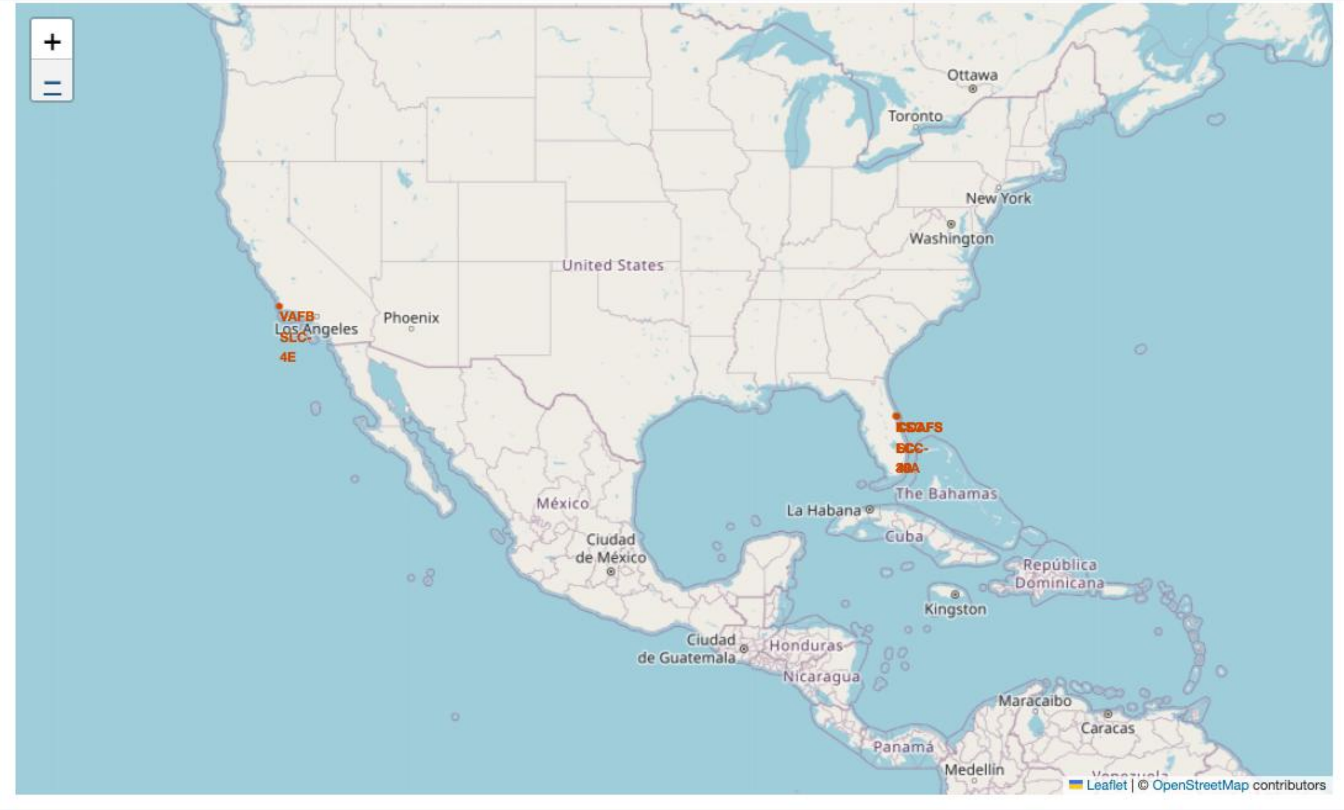
count(Landing_Outcome)	Landing_Outcome	Date
10	No attempt	2012-05-22
5	Success (drone ship)	2016-04-08
5	Failure (drone ship)	2015-01-10
3	Success (ground pad)	2015-12-22
3	Controlled (ocean)	2014-04-18
2	Uncontrolled (ocean)	2013-09-29
2	Failure (parachute)	2010-06-04
1	Precluded (drone ship)	2015-06-28

Section 3

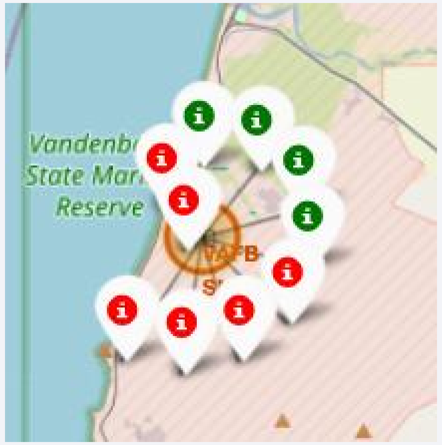
Launch Sites Proximities Analysis

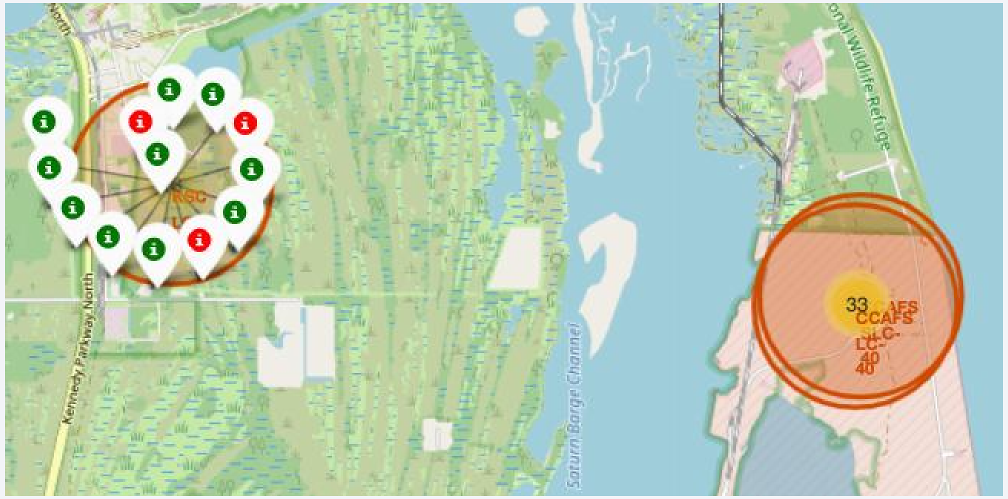
Launch sites in a global map

- All the launch sites locate in the US along the see coast



Launch sites with color markers and labels





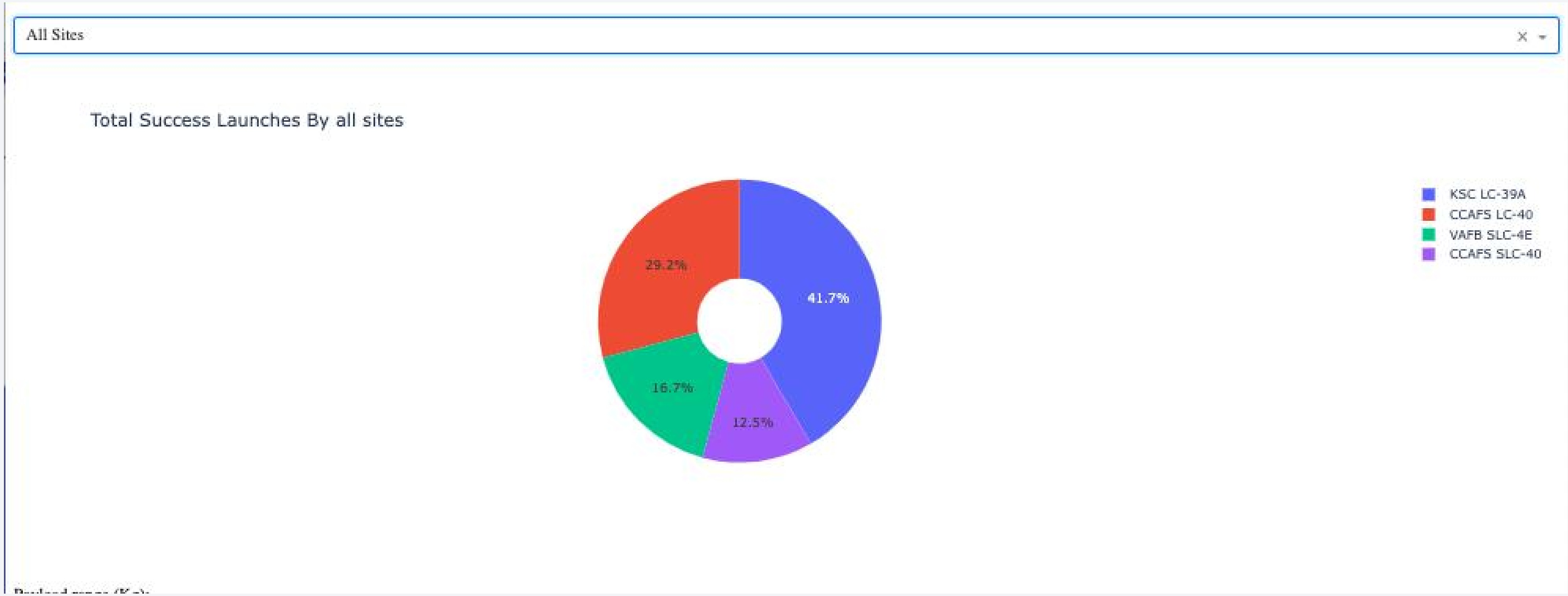
Launch sites on map with distance calculated to proximities

Section 4

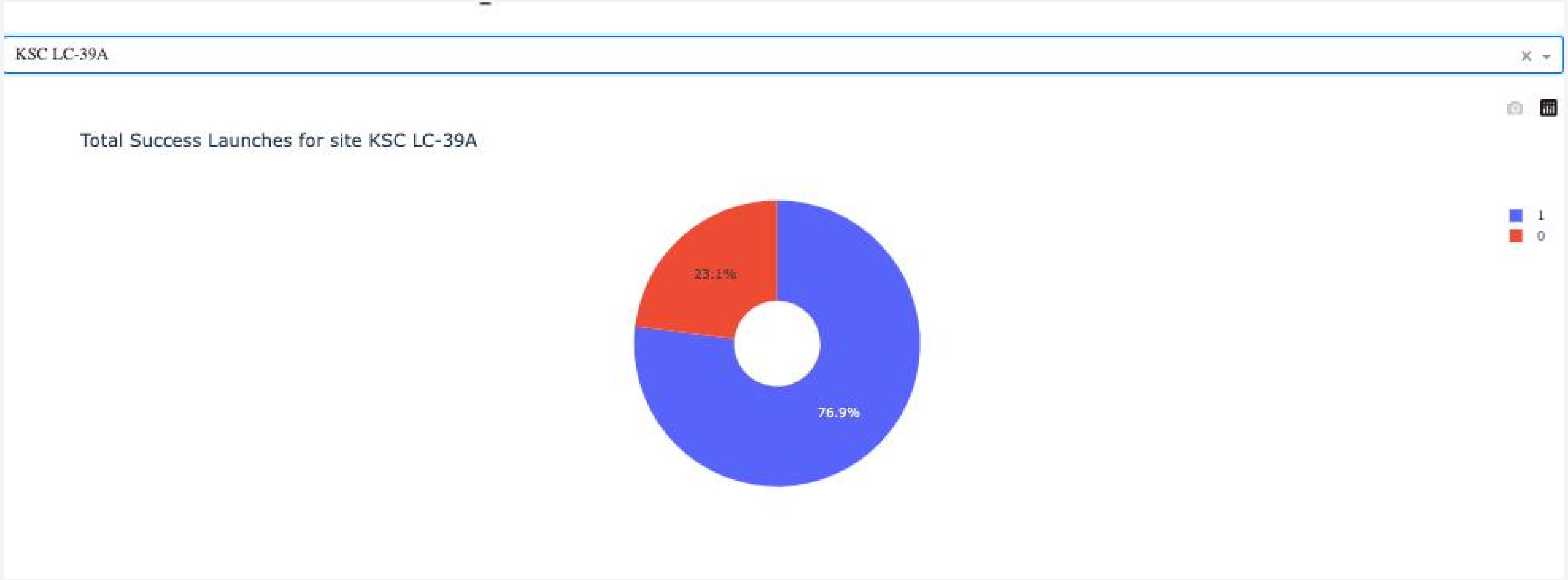
Build a Dashboard with Plotly Dash

Pie chart showing the success rate achieved by each launch site

- KSC LC has the largest success rate, followed by CCAFS LC, and VAFB. CCSFS SLC has the lowest success rate of 12.5%.

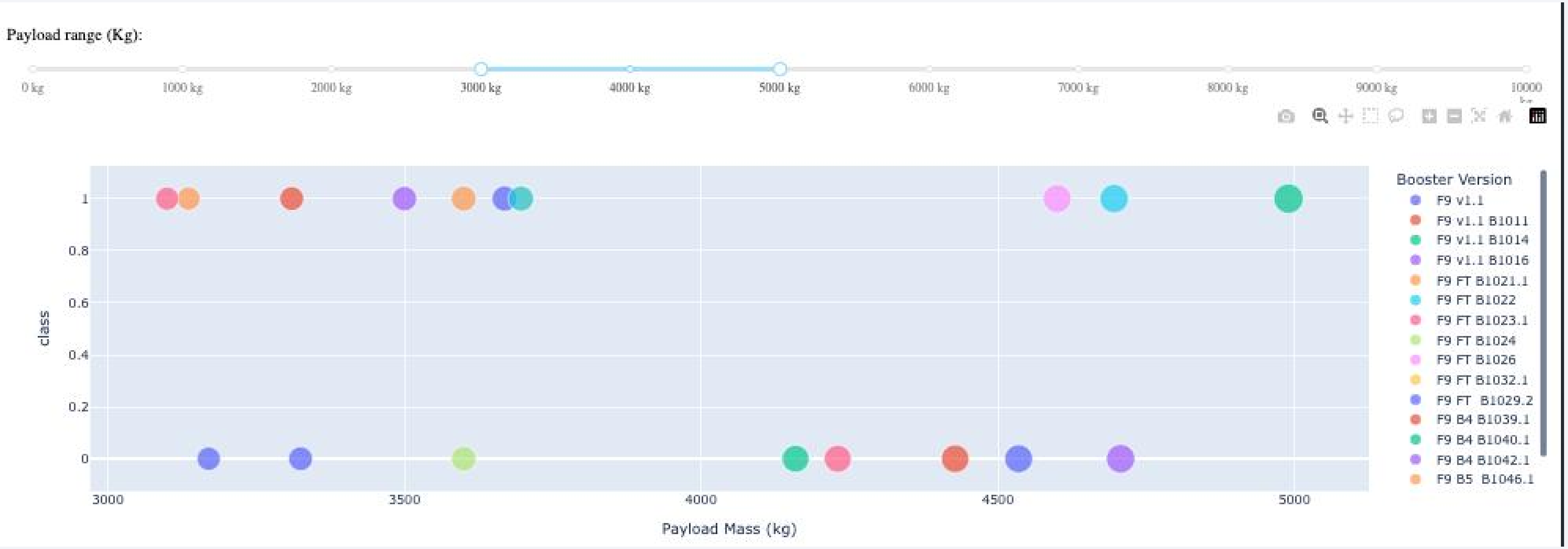


Pie chart showing the success ratio of the launch site with the highest success rate



Scatter plot showing relationships between Payload and Outcome given different booster version

- For payload ranging from 3000–4000kg, and 4600–5000kg, the success rate is higher

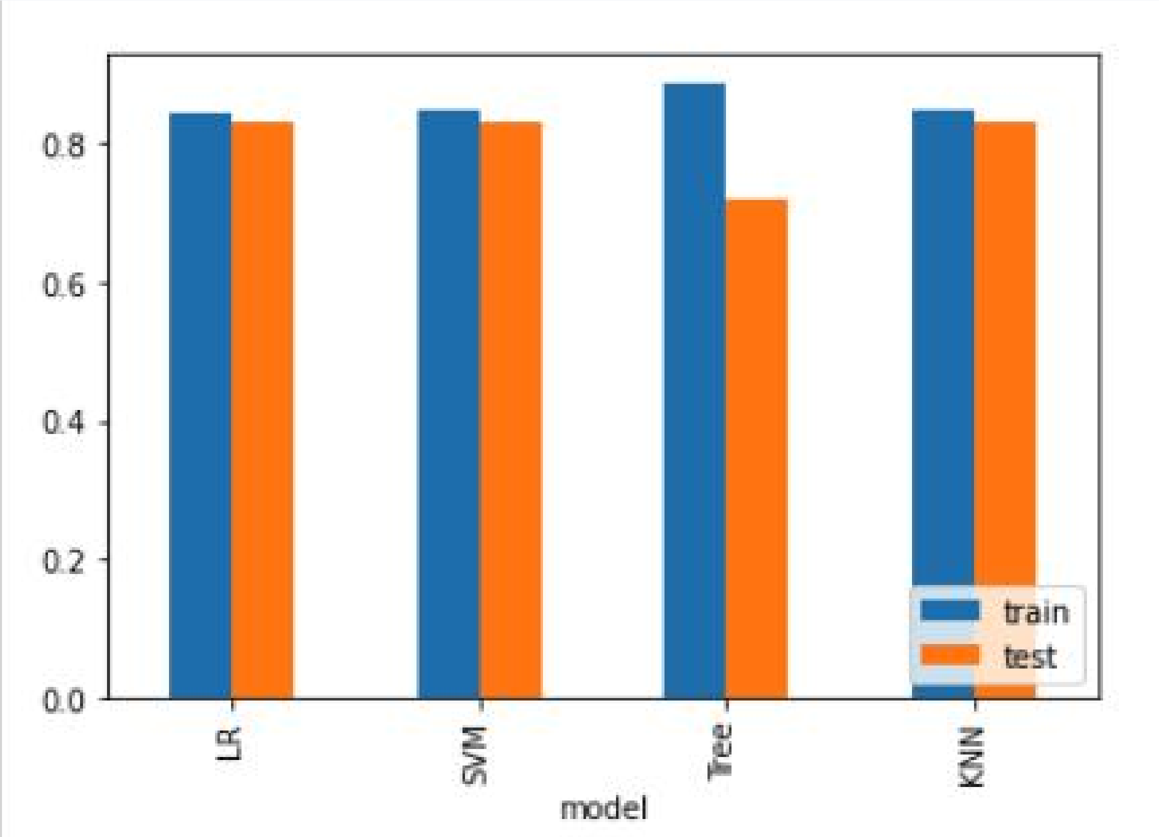


Section 5

Predictive Analysis (Classification)

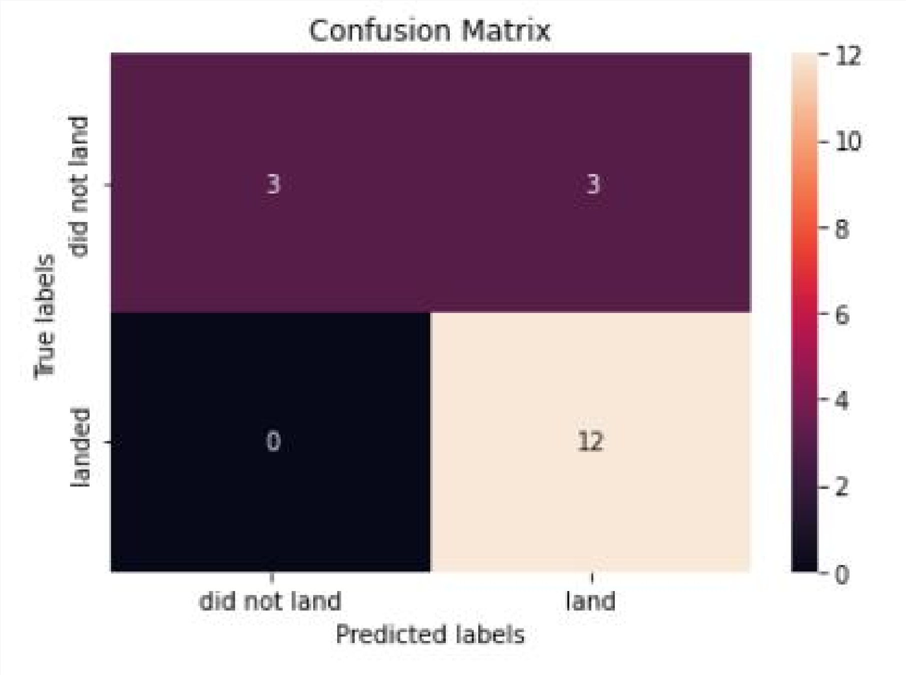
Classification Accuracy

- We train four types of ML models, i.e., Logistic Regression, SVM, Decision tree and KNN. LR, SVM and KNN show the same performance on both training set and testing set, with approximately 84% of accuracy.



Confusion Matrix

- All three models show the same results, with three False Positive samples.



Conclusions

- We can conclude that:
- All the launch sites have improved their techniques with more launch try–out, resulting in higher success rate.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES–L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC–39A had the most successful launches compared to other sites.
- 83% of accuracy is achieved with the best machine learning algorithm to predict whether the launch is successful.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!