IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Bianca Diana Șmalbelgher

April, 2023

Outline

Executive Summary

- Summary of methodologies

1. Data Collection API
2. Data Collection with Web Scraping
3. EDA
4. EDA with SQL
5. EDA with Visualization
6. Interactive Visual Analytics with Folium
7. Interactive Dashboard with Ploty Dash
8. Machine Learning Prediction

- Summary of all results

- Exploratory Data Analysis
- Interactive Analysis
- Predictive Analysis

Introduction

- Project background and context

In this project, we investigate how SpaceX can do rocket launches that are relatively inexpensive. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards of 165 million dollars each. We know that 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. Spaces X's Falcon 9 launch like regular rockets. Stage two, or the second stage, helps bring the payload to orbit, but most of the work is done by the first stage. The first stage is also much larger than the second stage. Unlike other rocket providers, SpaceX's Falcon 9 can recover the first stage. Sometimes the first stage does not land. Sometimes it will crash. Other times, Space X will sacrifice the first stage due to the mission parameters like payload, orbit, and customer. In this capstone, we will take the role of a data scientist working for a new rocket company, Space Y that would like to compete with SpaceX, founded by Billionaire industrialist Allon Musk. Our job is to determine the price of each launch. We will do this by gathering information about Space X and creating dashboards for our team. We will also determine if SpaceX will reuse the first stage. Instead of using rocket science to determine if the first stage will land successfully, we will train a machine learning model and use public information to predict if SpaceX will reuse the first stage.

- Problems you want to find answers

- Predict if the SpaceX will land successfully and determine the cost of a launch
- Collect Falcon 9 historical launch records from a Wikipedia page
- Find some patterns in the data and determine what would be the label for training supervised models
- Find an optimal location for building a launch site by finding out what factors influenced the most SpaceX launches
- Predict if the first stage will land given the data from Space X, and find the method that performs best using test data

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:

Data was collected using REST API and Web scraping.

- Perform data wrangling

We dealt with the missing values.

- Perform exploratory data analysis (EDA) using visualization and SQL

We calculated the number of launches on each site, the number and occurrence of each orbit, the number and occurrence of mission outcome per orbit type, then we created a landing outcome label. After that, we used SQL queries to retrieve information about launch sites and the features that influenced them.

- Perform interactive visual analytics using Folium and Plotly Dash

We first performed Exploratory Data Analysis, then we visualized a few graphs to understand the relationships between different features and different Launch Sites or Orbits. Then we performed different Folium steps to better understand the distance between the launch sites and different points (such as the coastline, railway, and so on). Then we looked at the Launch Success Rate for different Launch Sites.

- Perform predictive analysis using classification models

We standardized the data, we split the data into training and test data, then we created a logistic regression, a support vector machine, a decision tree classifier, and k nearest neighbors objects and we analyzed which was the method that performed best at predicting the landings.

Data Collection

- Describe how data sets were collected.
- Data was collected using REST API and Web Scraping from a Wikipedia page
- You need to present your data collection process use key phrases and flowcharts
- For REST API we used the get methods and BoosterVersion method
- For Web Scraping we used BeautifulSoup method

Data Collection – SpaceX API

[Link to Data Collection using REST API]

For REST API we used the get and BoosterVersion methods

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 [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

In [10]:
```
response.status_code
```

Out[10]: 200

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

In [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

In [12]:
```
# Get the head of the dataframe
data.head()
```

Out[12]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	payloads	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e45
1	None	NaN	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to premature	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min	[]	[]	[]	[5eb0e4b6b6c3bb0006eeb1e2]	5e9e45

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

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

In [29]:
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data_falcon9[data_falcon9["BoosterVersion"] == "Falcon 9"]
data_falcon9.head()
```

Out[29]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitu
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-80.577366	28.561
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-80.577366	28.561
7	11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003	-120.610829	34.632
8	12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-80.577366	28.561

Now that we have removed some values we should reset the FlgihtNumber column

In [30]:
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Out[30]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Lo
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-8
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-8
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-8
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003	-12
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	

Data Collection - Scraping

[Link to Data Collection with Web Scraping]

For Web Scraping we used BeautifulSoup method and then created 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 [19]:

```python
# use requests.get() method with the provided static_url
req = requests.get(static_url)

# assign the response to a object
response = req.text
```

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

In [20]:

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

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

In [21]:

```python
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the extern
this lab

In [22]:

```python
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all("table")
html_tables
```

In [60]:

```python
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
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            datatimelist=date_time(row[0])
            launch_dict['Flight No.'].append(flight_number)
            print(flight_number)

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            print(time)

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
                launch_dict['Version Booster'].append(bv)
            print(bv)

            # Launch Site
            # TODO: Append the launch_site into launch_dict with key `Launch Site`
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site)
            print(launch_site)

            # Payload
            # TODO: Append the payload into launch_dict with key `Payload`
            payload = row[3].a.string
            launch_dict['Payload'].append(payload)
            print(payload)
```

Data Wrangling

[Link to EDA lab]

We calculated the number of launches on each site, the number and occurrence of each orbit, the number and occurrence of mission outcome per orbit type, then we created a landing outcome label.

TASK 1: Calculate the number of launches on each site

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

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

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

In [11]:
```
# Apply value_counts() on column LaunchSite
df["LaunchSite"].value_counts()
```

Out[11]:
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

In [12]:
```
# Apply value_counts on Orbit column
df["Orbit"].value_counts()
```

Out[12]:
```
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
```

TASK 3: Calculate the number and occurence of mission outcome per orbit type

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

In [13]:
```
# landing_outcomes = values on Outcome column
landing_outcomes = df["Outcome"].value_counts()
landing_outcomes
```

Out[13]:
```
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
```

In [23]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []

for outcome in df["Outcome"]:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

In [24]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

Out[24]:

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

In [25]:
```
df.head(5)
```

Out[25]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Lat
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.5

EDA with Data Visualization

[Link to EDA with Visualization]

We used scatter plots to show the dependency relationship of different attributes between each other, such as:

- Launch Site and Flight Number
- Launch Site and Payload Mass
- Orbit and Flight Number
- Orbit and Payload Mass

We also used a bar chart and a line chart:

- To see the success rate of each orbit
- To get the average launch success trend by year

After that, we also predicted the impact of different features using Features Engineering with dummy variables.

EDA with SQL

[Link to EDA with SQL]

- We found out the unique launch sites in the space mission
- We displayed the launch sites beginning with ‘CCA’
- We displayed the total payload mass carried by boosters launched by NASA (CRS)
- We displayed average payload mass carried by booster version F9 v1.1
- We listed the date when the first successful landing outcome in ground pad was achieved
- We listed the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- We listed the total number of successful and failure mission outcomes
- We listed the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- We listed the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- We ranked 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

Please note: Given that on my github I get an error of notebook failed (you can check here, I inserted above the link to my notebook in cloud.ibm.com.

Build an Interactive Map with Folium

Link to Interactive Visual Analytics with Folium

- We created circles and markers for each launch site (CCAFS LC-40, CCAFS SLC-40, KSC LC-39A, VAFB SLC-4E) so we can answer the following questions:
  - Are all launch sites in proximity to the Equator line?
  - Are all launch sites in very close proximity to the coast?
- Then we created circles and coordinates to see the distance from the launch site locations and the coastline, and other places such as railway, highway, etc., so we can answer the following questions:
  - Are launch sites in close proximity to railways?
  - Are launch sites in close proximity to highways?
  - Are launch sites in close proximity to coastline?
  - Do launch sites keep certain distance away from cities?

Build a Dashboard with Plotly Dash

Link to Interactive Dashboard with Ploty Dash

- We created a dropdown so users can choose a Launch Site
- We created a success pie chart so visitors can see the outcomes pie chart for a selected site
- We created a success payload scatter chart to show the correlation between payload and launch success
- And finally we created a pie chart so that users can see the success rate for the selected site

Predictive Analysis (Classification)

[Link to Machine Learning Prediction]

- We standardized the data using build in preprocessing transformations.
- We split the data into training and test data using the function train_test_split.
- Then we created a logistic regression, a support vector machine, a decision tree classifier, and k nearest neighbors objects using GridSearchCV objects.
- After that, we analyzed which was the method that performed best at predicting the landings based on the r^2 accuracy score of each method.

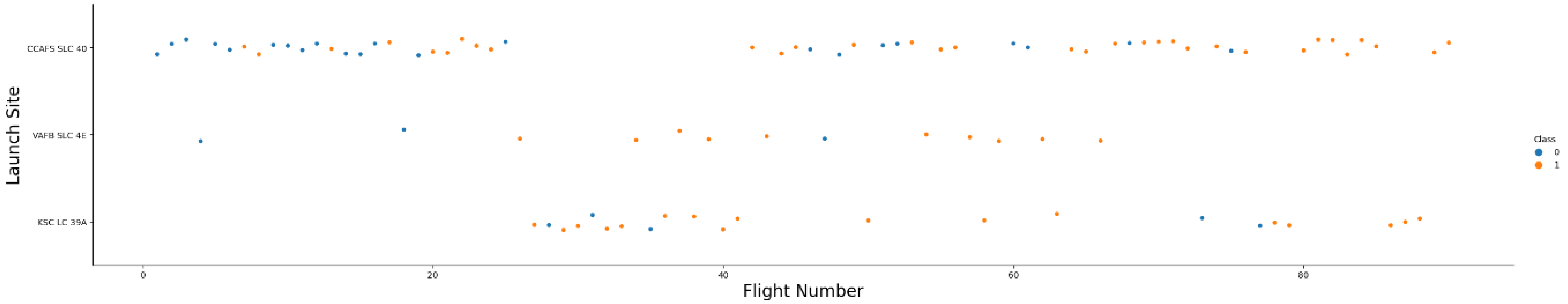
Results

- Based on the exploratory data analysis results, we can conclude the best methods to use for our data are the Logistic Regression, the SVM, and the KNN.
- We will further show analytics in screenshots

Section 2

Insights drawn from EDA

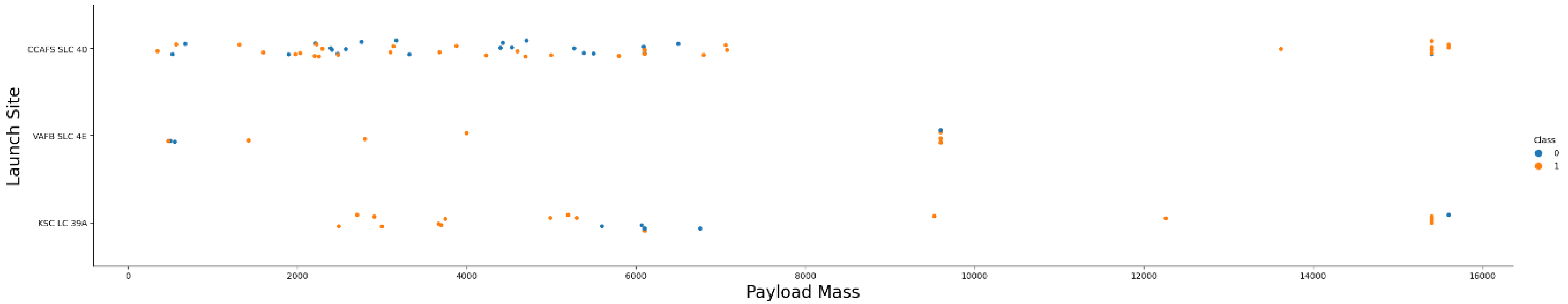
Flight Number vs. Launch Site



Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

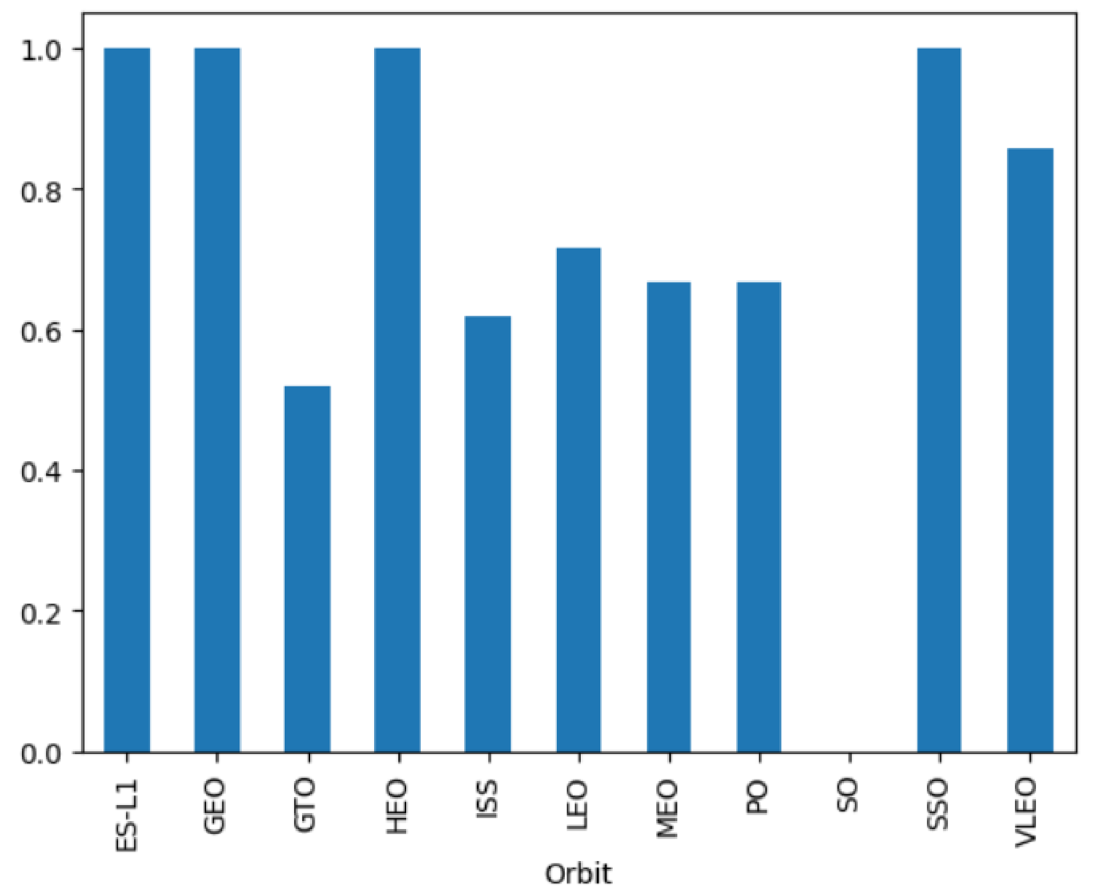
We can see that KSC LC-39A Launch Site has the majority of the success rate between approx 25% and approx 45%, VAFB SLC 4E has the majority of the success rate between approx 30% and 65%, and CCAFS LC-40 has the majority of success rate between 40% and 100%

Payload vs. Launch Site



Now if you observe Payload 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).

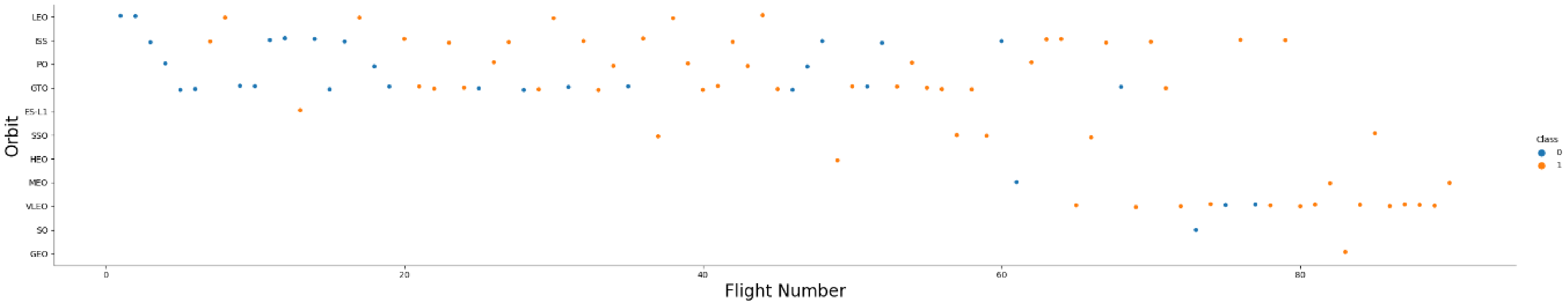
Success Rate vs. Orbit Type



Analyze the ploted bar chart try to find which orbits have high sucess rate.

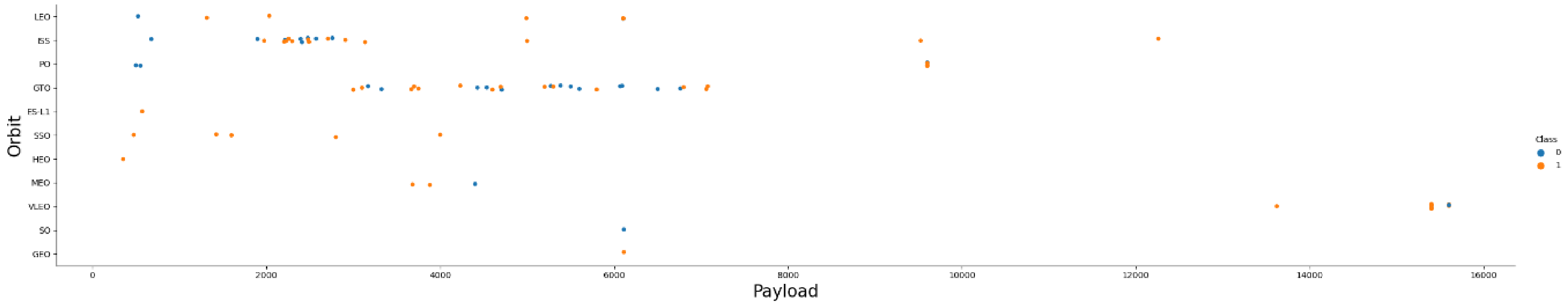
Orbits ES-L1, GEO, HEO, and SSO had the best success rate, followed closely by VLEO

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.

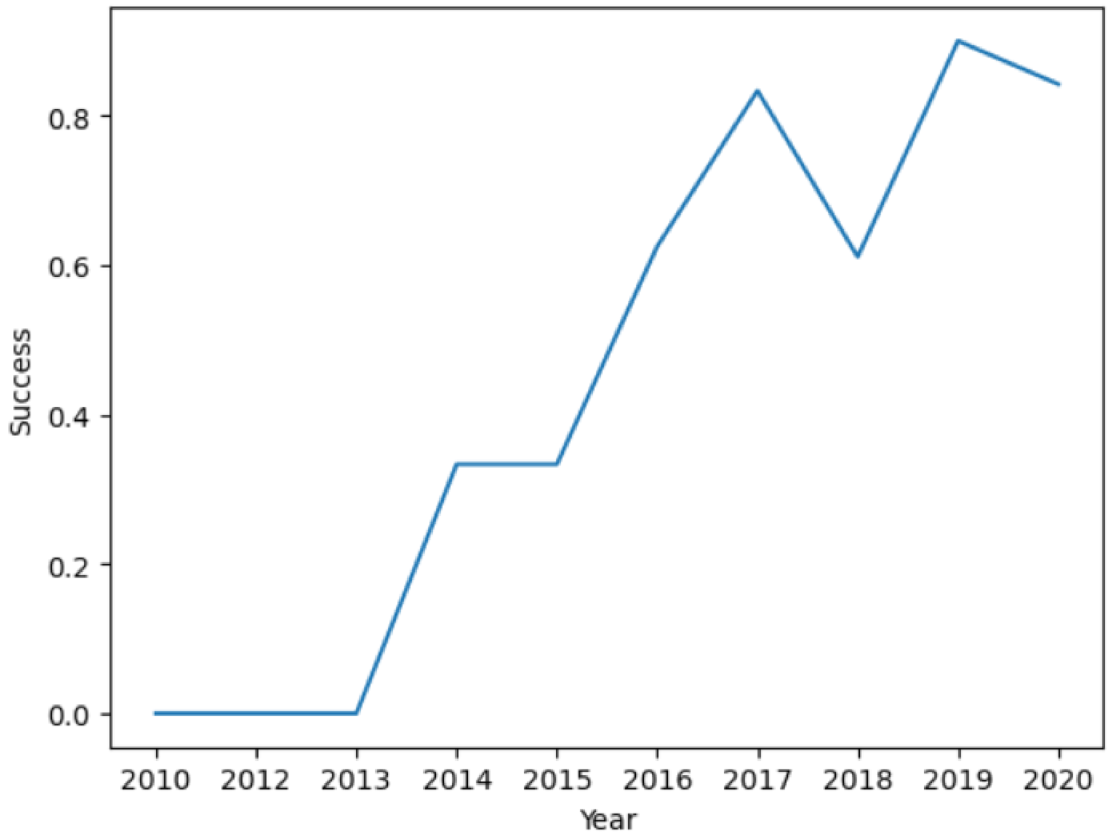
Payload vs. Orbit Type



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



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

All Launch Site Names

Task 1

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

In [11]:
```
%%sql

SELECT DISTINCT launch_site
FROM spacex;
```

```
 * ibm_db_sa://swx90196:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[11]:

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

We used DISTINCT to display unique names of sites

Launch Site Names Begin with 'CCA'

Task 2

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

In [12]:
```
%%sql

SELECT *
FROM spacex
WHERE launch_site LIKE '%CCA%'
LIMIT 5;
```

```
 * ibm_db_sa://swx90196:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[12]:

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

We used LIKE and “%...%” to find out the names that begin with ‘CCA’

Total Payload Mass

Task 3

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

In [31]:
```
%%sql
SELECT SUM(payload_mass_kg_) AS sum_of_payload_mass_NASA_CRS
FROM spacex
WHERE customer = 'NASA (CRS)';
```

```
 * ibm_db_sa://swx90196:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[31]:

sum_of_payload_mass_nasa_crs
45596

We used SUM to find out the total payload mass

Average Payload Mass by F9 v1.1

Task 4

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

In [32]:
```
%%sql
SELECT AVG(payload_mass_kg_) AS avg_payload_mass_F9v11
FROM spacex
WHERE booster_version = 'F9 v1.1';
```

```
 * ibm_db_sa://swx90196:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[32]:

avg_payload_mass_f9v11
2928

We used AVG to find out the average payload mass carried by booster F9 v1.1

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

In [33]:
```
%%sql
SELECT MIN(DATE) AS first_successful_ground_pad_landing
FROM spacex
WHERE landing_outcome = 'Success (ground pad)';
```

```
 * ibm_db_sa://swx90196:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[33]:

first_successful_ground_pad_landing
2015-12-22

We used MIN to find the dates of the first successful landing outcome on ground pad

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

In [34]:
```
%%sql
SELECT booster_version, payload_mass_kg_
FROM spacex
WHERE landing_outcome = 'Success (drone ship)'
AND payload_mass_kg_ BETWEEN 4000 AND 6000;
```

```
 * ibm_db_sa://swx90196:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[34]:

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

We used 2 conditions (WHERE … = … AND … BETWEEN … AND …) to list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

In [35]:
```
%%sql
SELECT mission_outcome, COUNT(mission_outcome) AS nr_mission_outcomes
FROM spacex
GROUP BY mission_outcome;
```

```
 * ibm_db_sa://swx90196:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[35]:

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

We used COUNT and GROUP BY to calculate the total number of successful and to show the failure mission outcomes

Boosters Carried Maximum Payload

Task 8

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

In [45]:
```
%%sql

SELECT booster_version AS booster_versions, payload_mass_kg_ AS max_payload_mass
FROM spacex
WHERE payload_mass_kg_= (SELECT MAX(payload_mass_kg_) FROM spacex);
```

```
 * ibm_db_sa://swx90196:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[45]:

booster_versions	max_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

We used a subquery to list the names of the booster which have carried the maximum payload mass

2015 Launch Records

Task 9

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [112…

```
%%sql
SELECT DATE, booster_version, launch_site, landing_outcome
FROM spacex
WHERE YEAR(DATE) = '2015'
AND landing_outcome = 'Failure (drone ship)';
```

```
 * ibm_db_sa://swx90196:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

Out[112]:

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

We used YEAR(column) to list the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

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

Task 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

In [148…

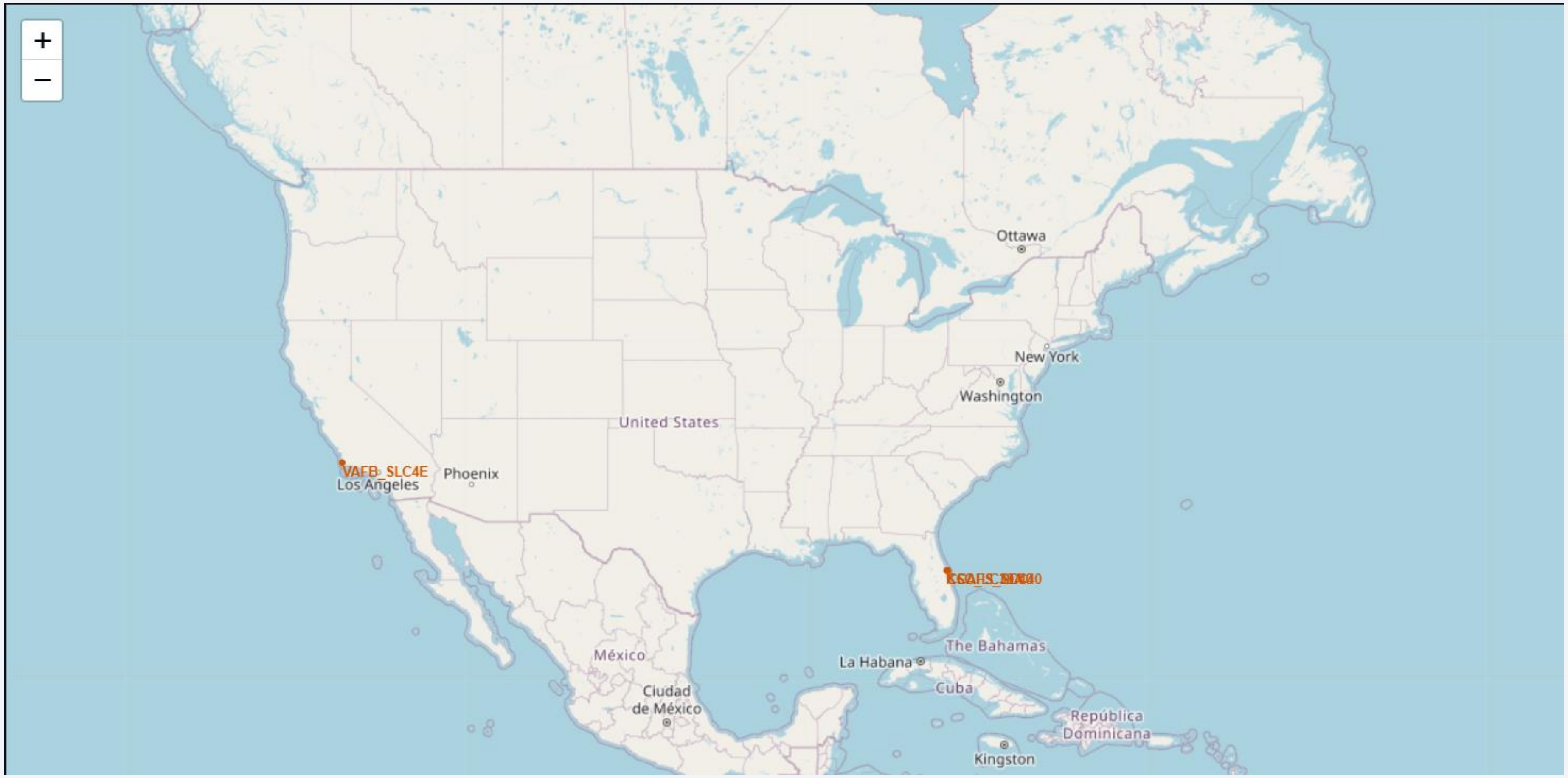
```
%%sql
SELECT landing_outcome, COUNT(landing_outcome) AS counted_landings,
FROM spacex
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY landing_outcome
ORDER BY COUNT(landing_outcome) DESC;
```

We used COUNT, GROUP BY and ORDER BY to 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

Section 3

Launch Sites Proximities Analysis

Location of all the Launch Sites



We can see that all the locations of the launch sites are close to the coastlines.

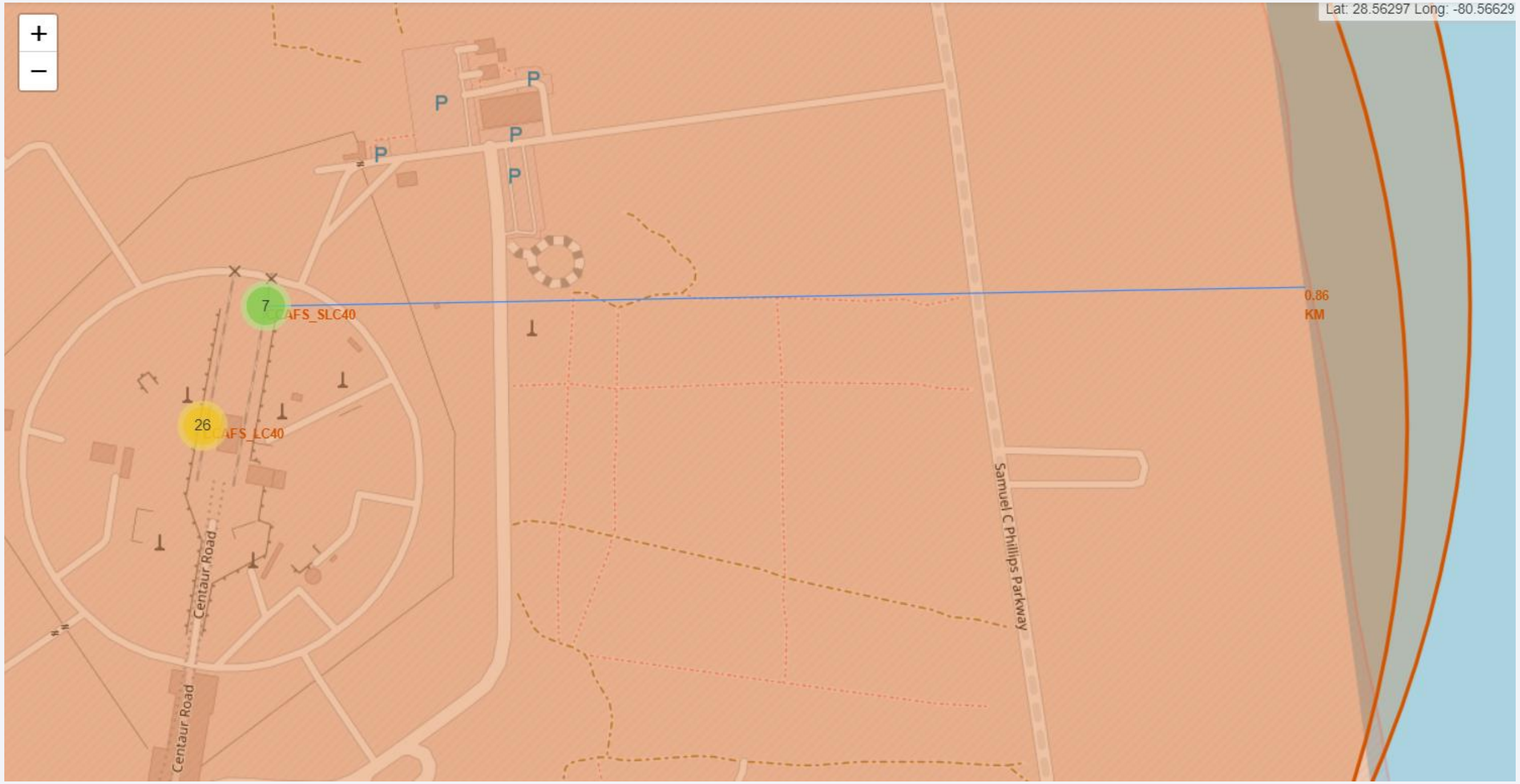
Launch Sites with color labels

We can see that the launch sites have color labels now.

Green represents successful launches.

Red represents unsuccessful launches.

Launch Sites Distance to Coastline



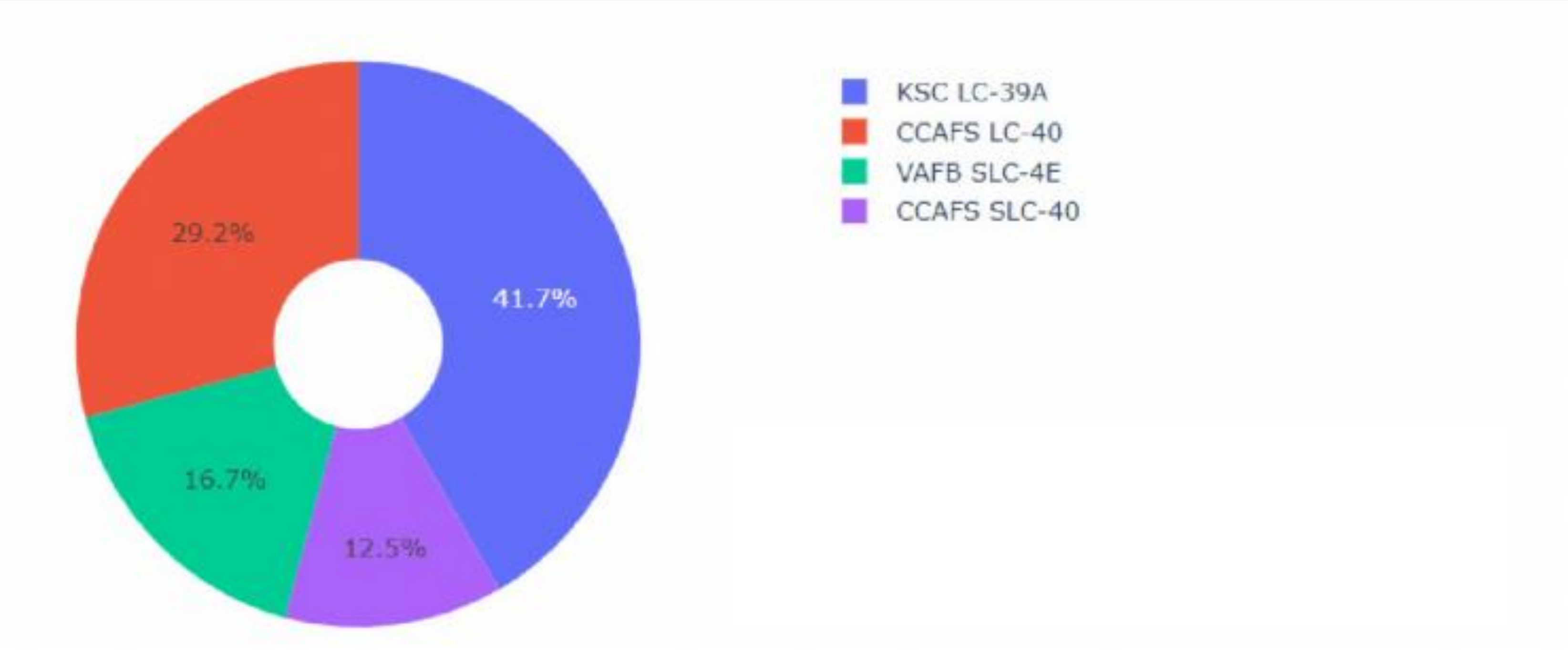
We can see the blue line drawn from the launching site to the coastline.

Where the line ends, near the coastline, we can see with red a number that represents the number of km from the launch site to the coast line.

Section 4

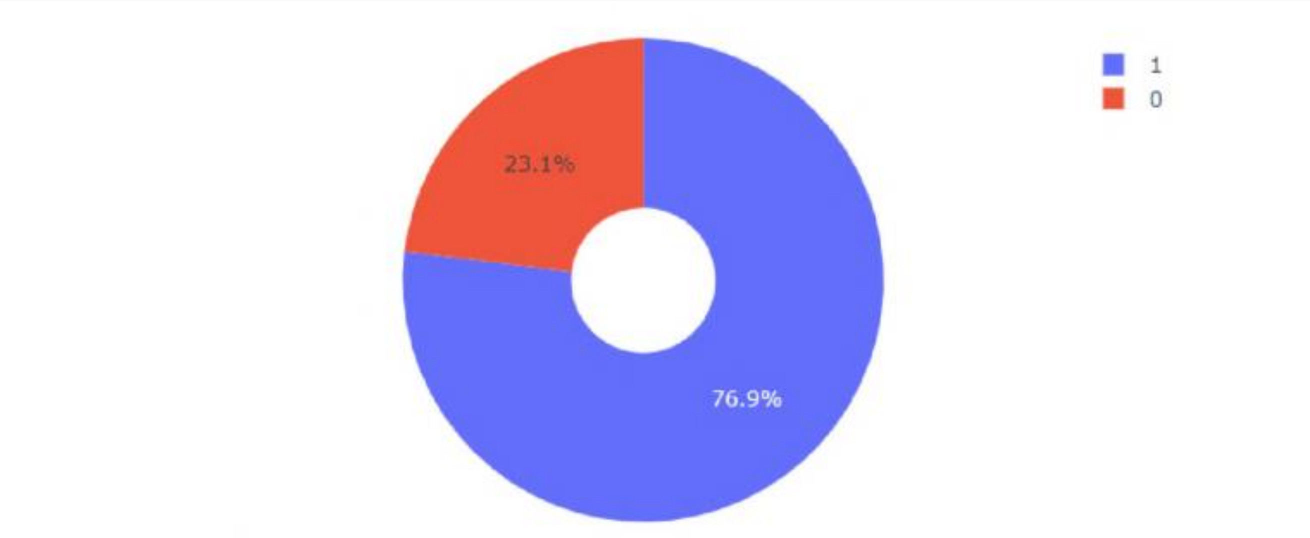
Build a Dashboard with Plotly Dash

Launch Sites Success



We created a pie chart so visitors can see the differences between all the launch sites

Success Rate

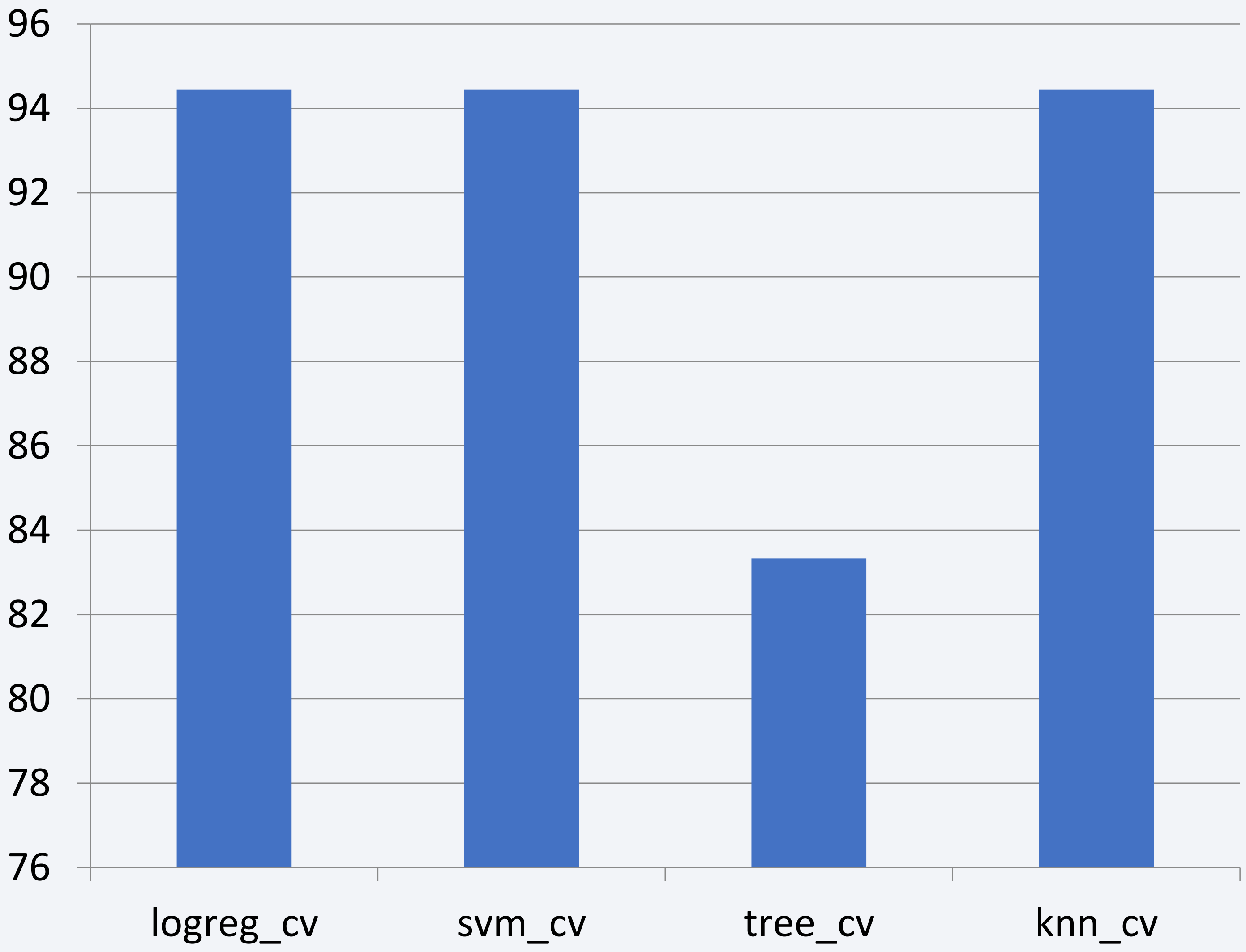


We created a success pie chart so visitors can see the difference between successful (blue) and failure (red) for a selected site

Section 5

Predictive Analysis (Classification)

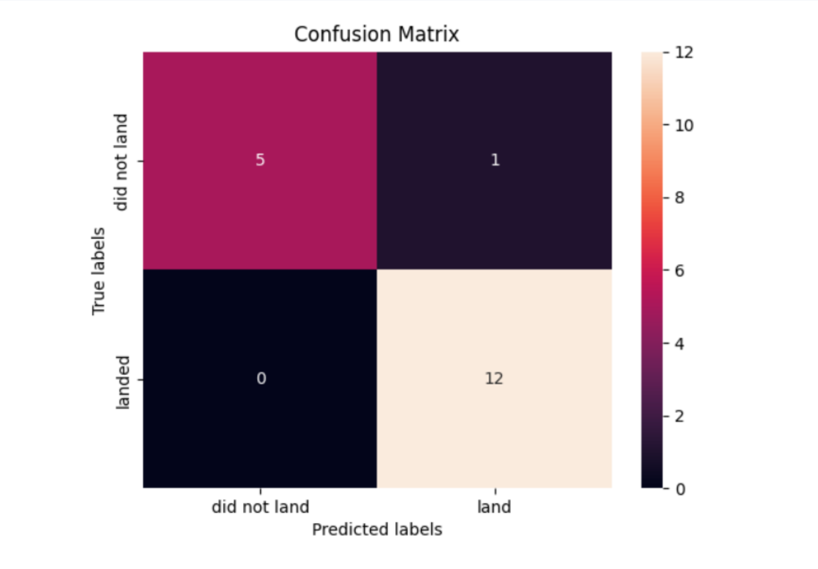
Classification Accuracy



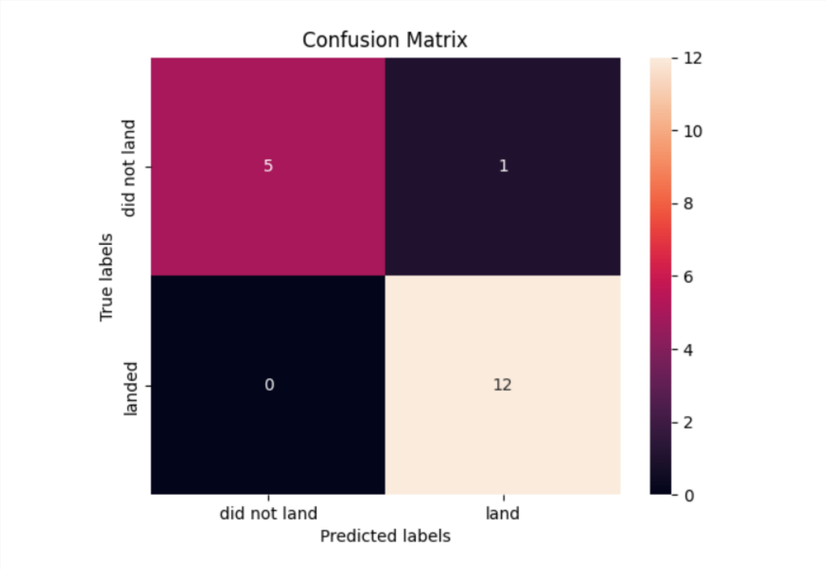
Based on the exploratory data analysis results, we can conclude the best methods to use for our data are the Logistic Regression, the SVM, and the KNN.

Confusion Matrix

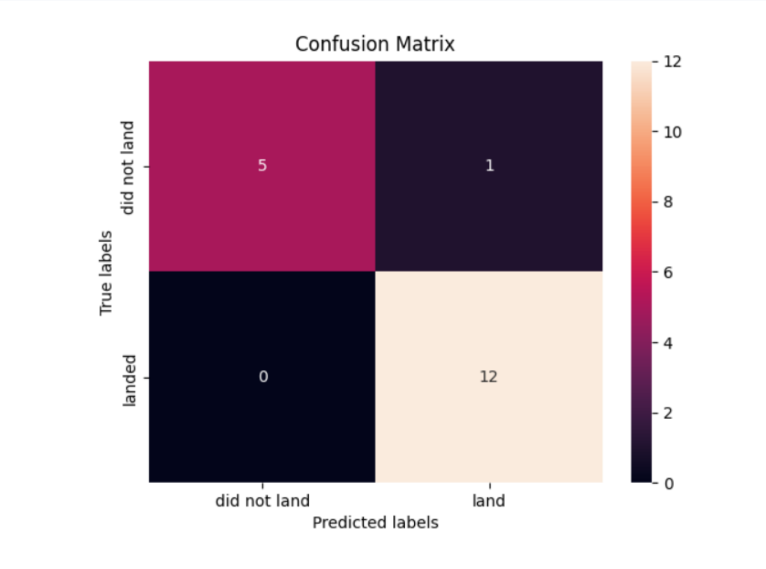
Logistic Regression

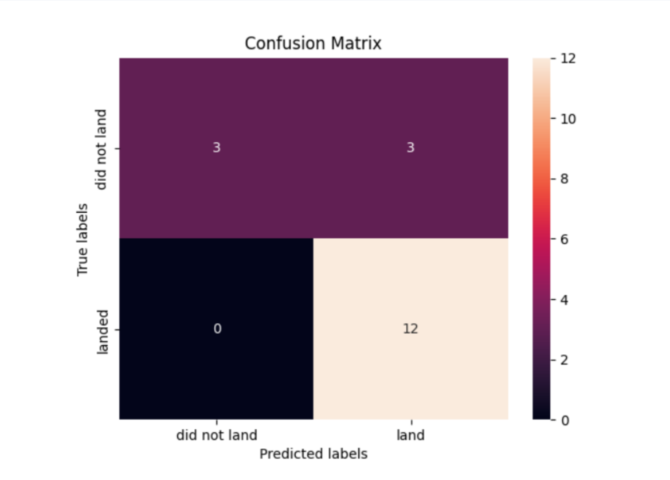


Support Vector Machine



K Nearest Neighbors





Decision Tree

As we can see, best methods for predictions are Logistic Regression, Support Vector Machine, and KNN

Conclusions

- We can conclude the best methods to use for our data are the Logistic Regression, the SVM, and the KNN.
- The first successful landing outcome in ground pad was achieved in 2015, on 22 of December
- Orbits ES-L1, GEO, HEO, and SSO had the best success rate, followed closely by VLEO
- Between all launch sites, the KSC LC 39A site had the most success

Thank you!