IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Eric Franke
Sept 07, 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Used data collection, cleaning and analysis to determine rocket launch success factors
- Summary of all results
  - This was a useful exercise to learn and demonstrate skills acquired from this course

Introduction

- Project background and context
  - The context of this project is to determine the cost of a launch, in case an alternate company wants to bid against SpaceX for a rocket launch
- Problems you want to find answers
  - Chiefly want to demonstrate knowledge, but also determine rocket launch success factors

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - Data was collected using API calls to SpaceX free data
- Performed data wrangling
    - Data was cleaned from NaN occurrences using Python libraries
- Performed exploratory data analysis (EDA) using visualization and SQL
- Performed interactive visual analytics using Folium and Plotly Dash
- Performed predictive analysis using classification models
    - Data was analyzed

Data Collection

- Describe how data sets were collected.
  - Data sets were collected and created from the main Spacex Launch data
- You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

https://github.com/erfranke/IBM-Data-Science/blob/main/spacex-data-collection-api-lab1.ipynb

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 [35]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/A
```

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

In [36]:
```
response.status_code
```

Out[36]: 200

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

In [37]:
```
# 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 [38]:
```
# Get the head of the dataframe
data.head(5)
```

Out[38]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships
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	[]	[]

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts

https://github.com/erfranke/IBM-Data-Science/blob/main/spacex-webscraping.ipynb

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 external reference link towards the end of this lab

In [9]:

```
# 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')
```

Starting from the third table is our target table contains the actual launch records.

In [10]:

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
```

Data Wrangling

- Data was processed and a new Landing Outcome column label was created from Outcome Column
- Success rate was %67

https://github.com/erfranke/IBM-Data-Science/blob/main/spacex-data_wrangling_lab2.ipynb

EDA with Data Visualization

- Scatter plots, line plots and bar charts were used to display and categorize launch data and outcomes

https://github.com/erfranke/IBM-Data-Science/blob/main/spacex-eda-dataviz.ipynb

EDA with SQL

- Used SQL select calls to organize and arrange data for analysis

https://github.com/erfranke/IBM-Data-Science/blob/main/spacex-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Markers, circles, lines were added to a folium map to display distances and locations to geographical features that assist rocket launches, to gain insight as to how they help

https://github.com/erfranke/IBM-Data-Science/blob/main/spacex-launch_site_location.ipynb

Build a Dashboard with Plotly Dash

Pie charts and other selector items were organized to present launch characteristics and success factors

https://github.com/erfranke/IBM-Data-Science/blob/main/spacex_dash_app1.py

Predictive Analysis (Classification)

After comparing accuracy of above methods, they all preformed practically the same, except for tree which fit train data slightly better but test data worse.

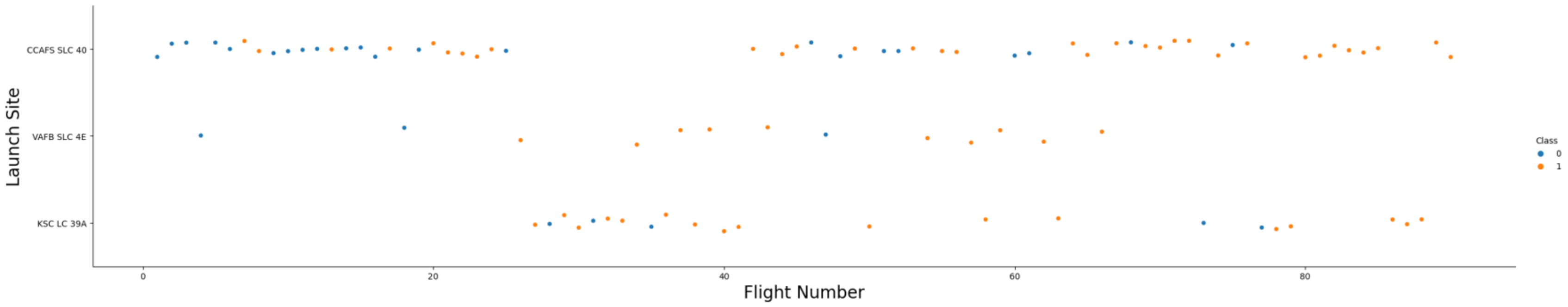
https://github.com/erfranke/IBM-Data-Science/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site



- Show the screenshot of the scatter plot with explanations

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

In [9]:

```
# Launch site SSAFS SLC 40 at higher flight numbers has a higher success rate
# KSC LC 39A and VAFB SLC 4E have proportianlly better success rates than CCAFS SLC 40
```

We also want to observe if there is any relationship between launch sites and their payload mass

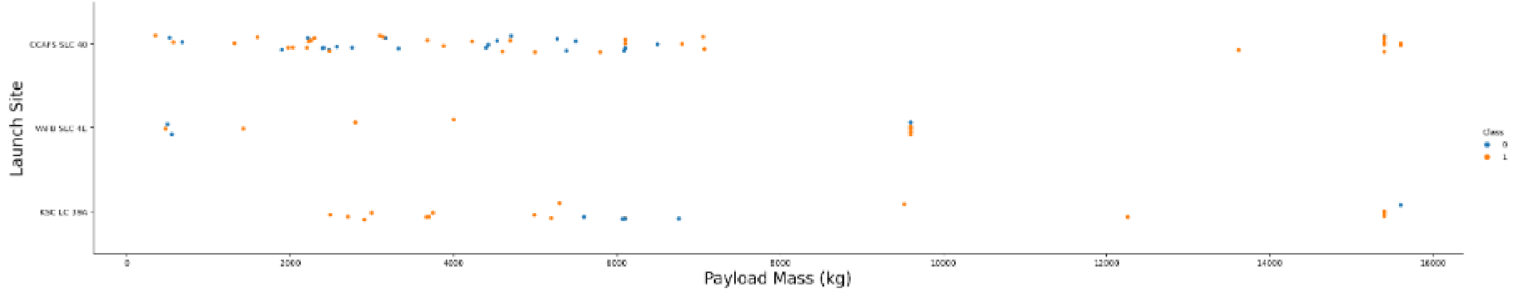
Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site

- Show the screenshot of the scatter plot with explanations

```
In [7]:
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class

sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Success Rate vs. Orbit Type

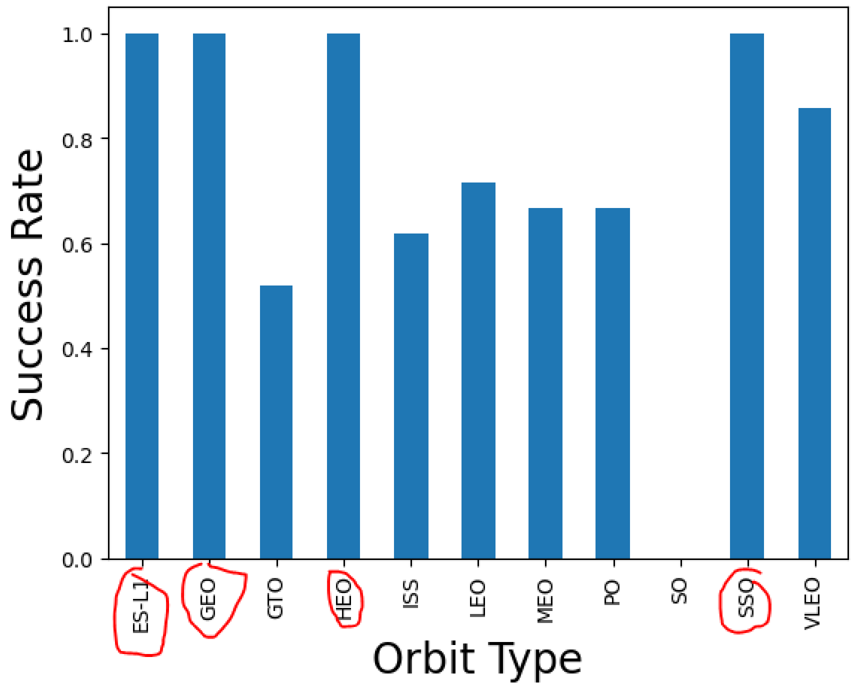
- Show a bar chart for the success rate of each orbit type

- Show the screenshot of the scatter plot with explanations

In [12]:

```
# HINT use groupby method on Orbit column and get the mean of Class column
df.groupby("Orbit").mean()['Class'].plot(kind='bar')
plt.xlabel("Orbit Type",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```

```
<ipython-input-12-41a5b23f1f94>:2: FutureWarning: The default value of numeric_only in DataFrameGr
a future version, numeric_only will default to False. Either specify numeric_only or select only c
for the function.
  df.groupby("Orbit").mean()['Class'].plot(kind='bar')
```



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

Flight Number vs. Orbit Type

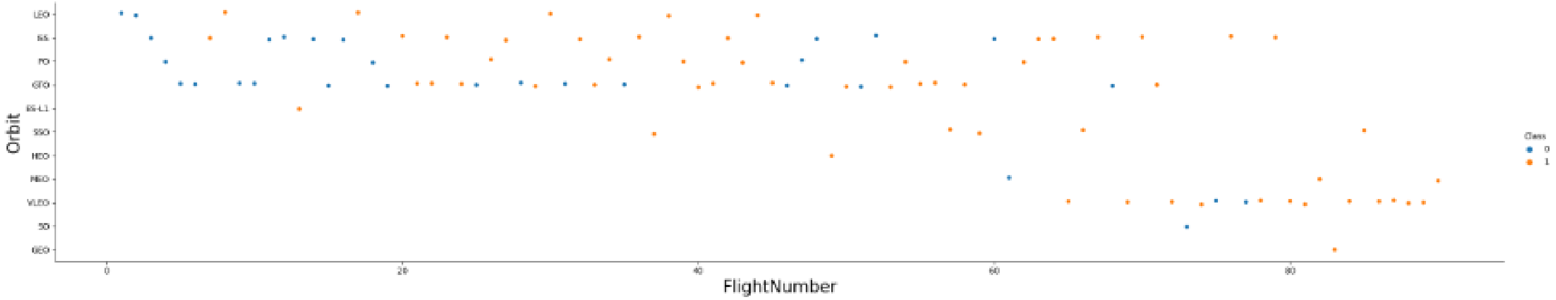
- Show a scatter point of Flight number vs. Orbit type
- Show the screenshot of the scatter plot with explanations

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

In [ ]:
```
### TASK  4: Visualize the relationship between FlightNumber and Orbit type
```

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

In [13]:
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



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.

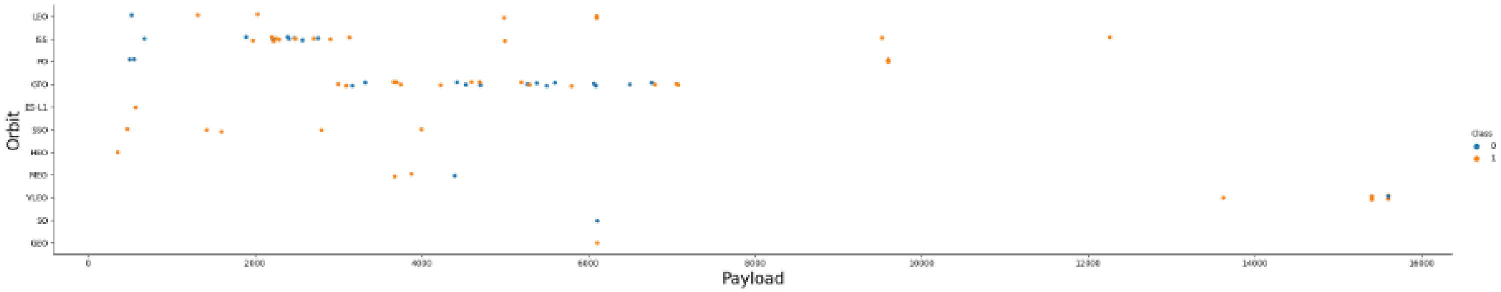
Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type

- Show the screenshot of the scatter plot with explanations

In [14]:

```python
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



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.

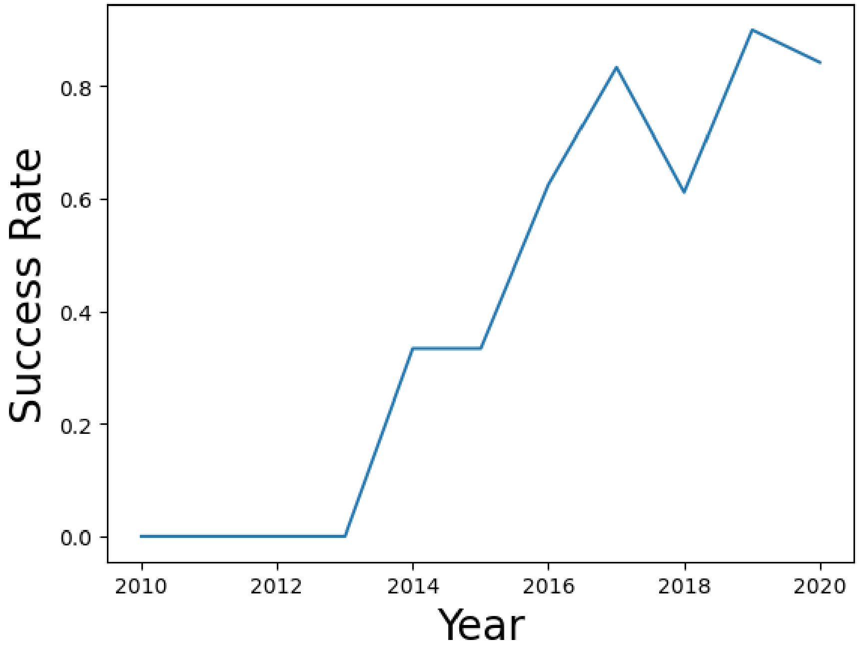
Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- Show the screenshot of the scatter plot with explanations

```
In [52]:
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate

df["Year"] = pd.DatetimeIndex(df["Date"]).year.astype(int)

df_year = df.groupby(df['Year'], as_index=False).agg({"Class": "mean"})
#df_orbit
sns.lineplot(y="Class", x="Year", data=df_year)
plt.xlabel("Year", fontsize=20)
plt.ylabel("Success Rate", fontsize=20)
plt.show()
```



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

Task 1

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

[17]:

```
%%sql
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXTBL;
```

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

t[17]:

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`

Task 2

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

In [29]:
```
%%sql
SELECT "Date", PAYLOAD, LAUNCH_SITE
FROM SPACEXTBL
WHERE LAUNCH_SITE LIKE 'CCA%'
LIMIT 5;
```

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

Out[29]:

Date	Payload	Launch_Site
2010-04-06	Dragon Spacecraft Qualification Unit	CCAFS LC-40
2010-08-12	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	CCAFS LC-40
2012-05-22	Dragon demo flight C2	CCAFS LC-40
2012-08-10	SpaceX CRS-1	CCAFS LC-40
2013-01-03	SpaceX CRS-2	CCAFS LC-40

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

Task 3

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

[30]:
```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)';
```

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

t[30]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

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

Task 4

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

n [31]:

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Booster_Version LIKE 'F9 v1.0%';
```

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

ut[31]:

AVG(PAYLOAD_MASS__KG_)
340.4

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

Task 5

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

Hint:Use min function

```
]:  %%sql
    SELECT MIN(Date)
    FROM SPACEXTBL
    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

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 [51]:

```
%%sql
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE LANDING_OUTCOME = "Success (drone ship)"
    AND 4000 < PAYLOAD_MASS__KG_ < 6000;
```

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

Out[51]:

Booster_Version
F9 FT B1021.1
F9 FT B1022
F9 FT B1023.1
F9 FT B1026
F9 FT B1029.1
F9 FT B1021.2
F9 FT B1029.2
F9 FT B1036.1
F9 FT B1038.1
F9 B4 B1041.1
F9 FT B1031.2
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

Task 7

List the total number of successful and failure mission outcomes

In [52]:

```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

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

Out[52]:

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

Task 8

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

```
%%sql
SELECT DISTINCT BOOSTER_VERSION
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL);
```

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

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

Task 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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
%sql select "Landing_Outcome", substr(Date,1,4), substr(Date,6,2),  "Booster_Version", "Launch_Site" from SPACEXTABLE where
```

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

Landing_Outcome	substr(Date,1,4)	substr(Date,6,2)	Booster_Version	Launch_Site
Failure (drone ship)	2015	10	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	2015	04	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

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.

[61]:

```
%%sql SELECT "LANDING_OUTCOME", COUNT(*) as 'COUNT' FROM SPACEXTBL

WHERE substr(Date,1,4) || substr(Date,6,2) || substr(Date,9,2)

between '20100604' and '20170320' GROUP BY "Landing_Outcome" ORDER BY "COUNT" DESC;
```

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

[61]:

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

Section 3

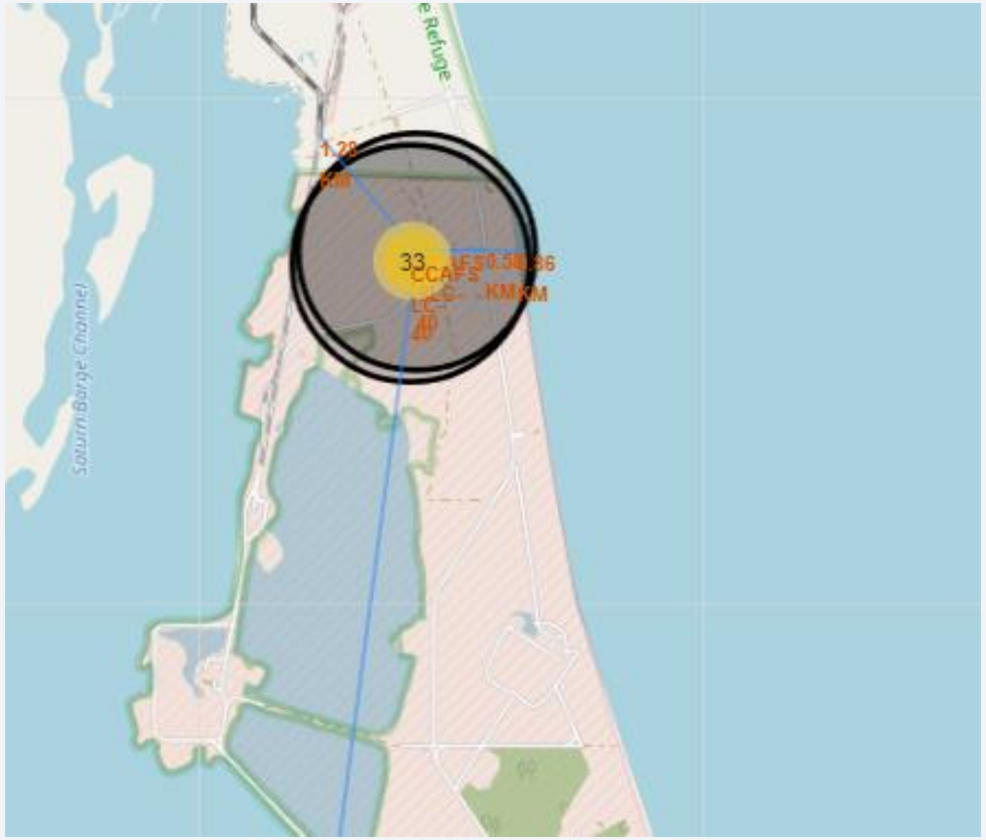
Launch Sites Proximities Analysis

US Launch Sites

Launch sites we are considering below

Marked Success/Fail Launches per site

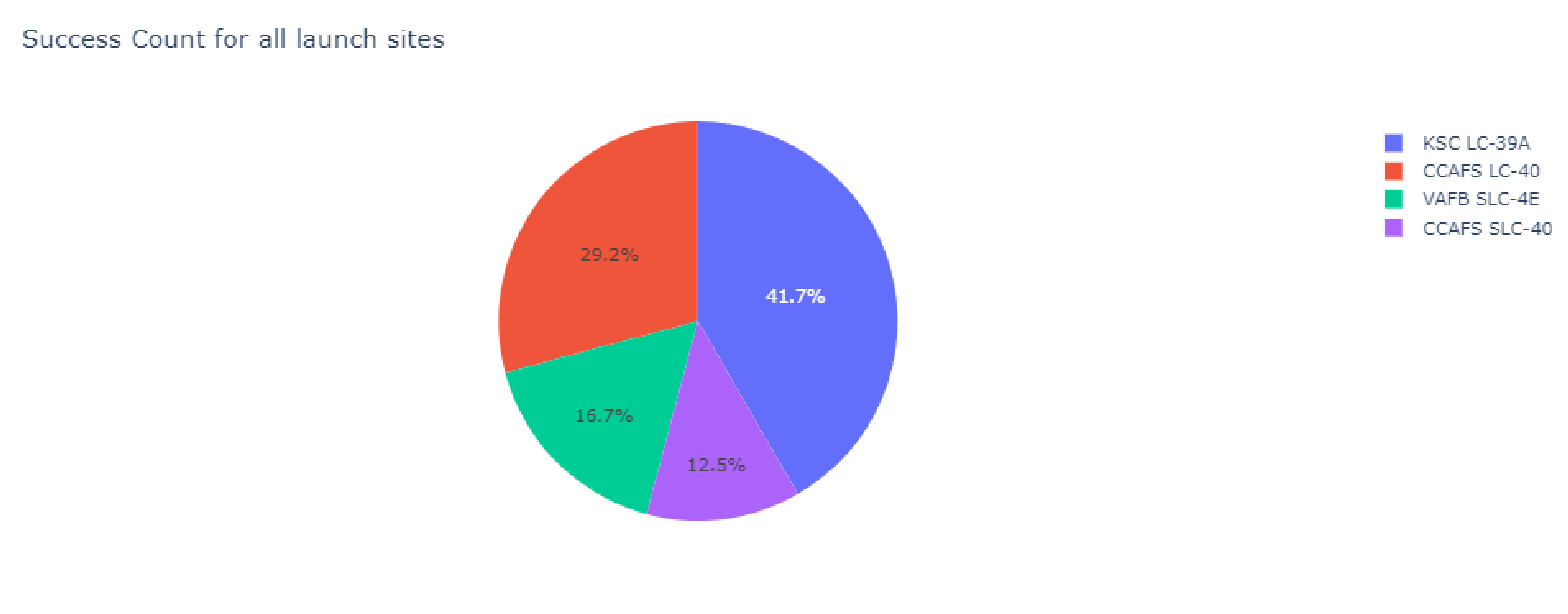
Proximity of launch site to success factors (rail, sea, etc)



Section 4

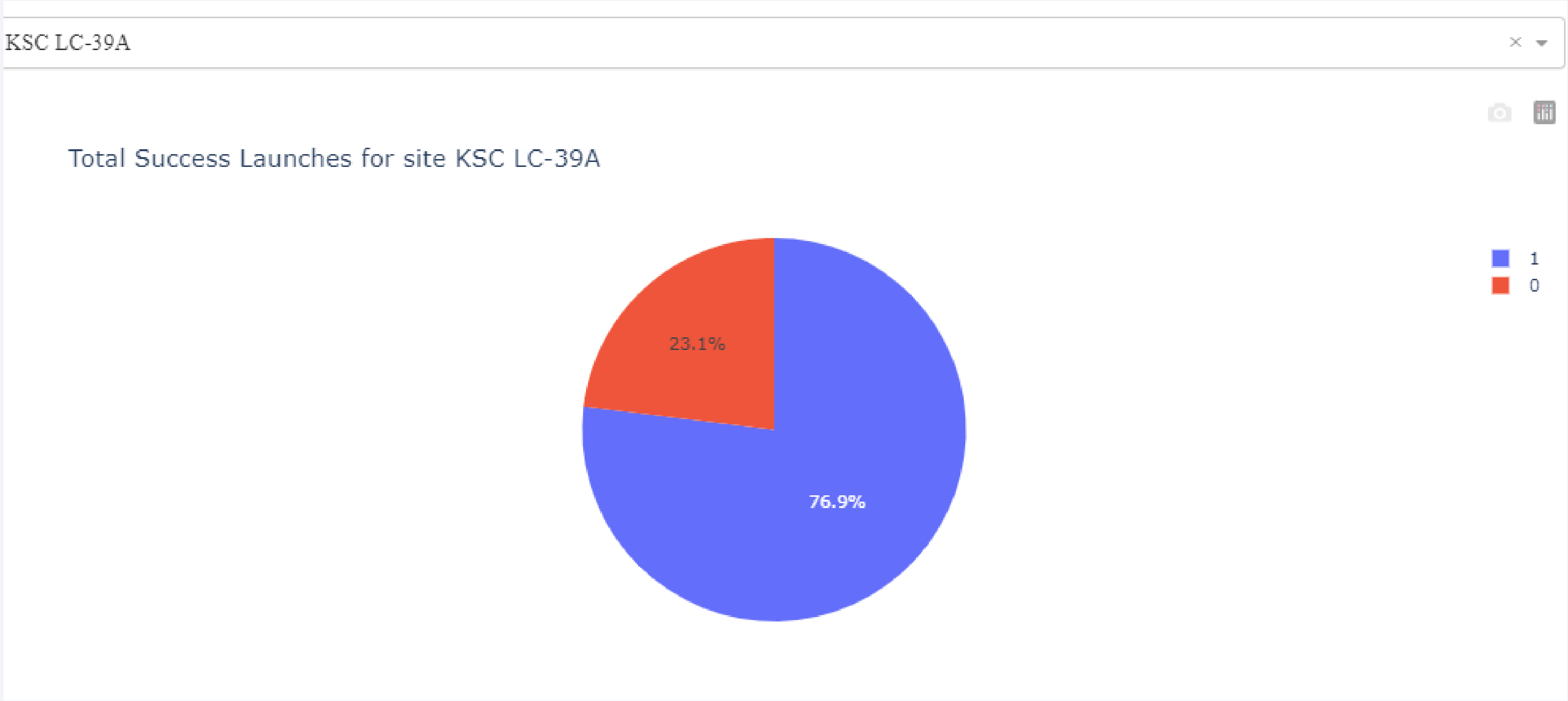
Build a Dashboard with Plotly Dash

Launch Success Count, All Sites

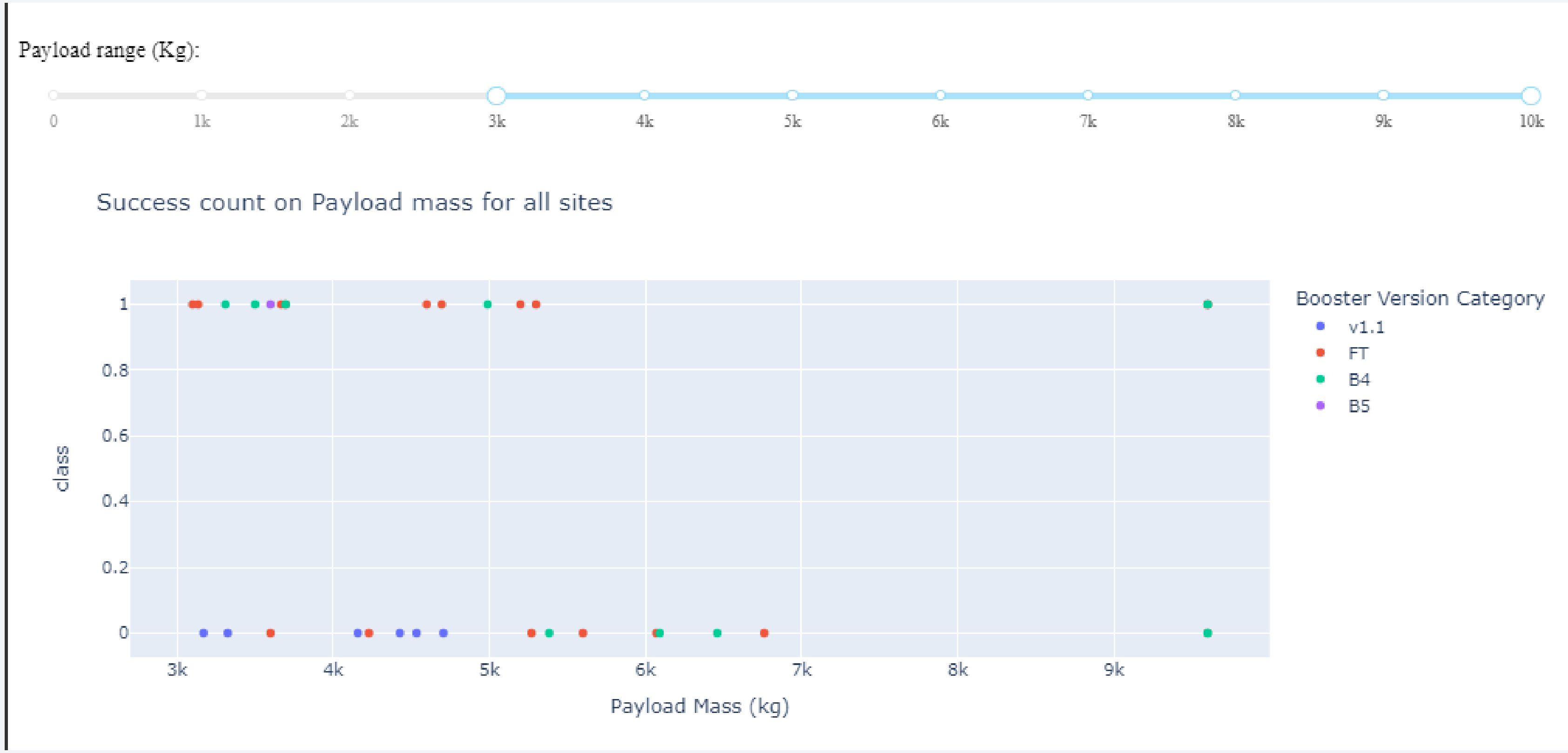


- Most of the successful launches come from KSC LC-39A

KSC LC-39A launch success ratio



All Sites Success count on Payload Mass



- At 3000 kg payload mass, the most successful booster version appears to be FT, but that could also be because it's the most frequent

Section 5

Predictive Analysis (Classification)

Confusion Matrix

TASK 7

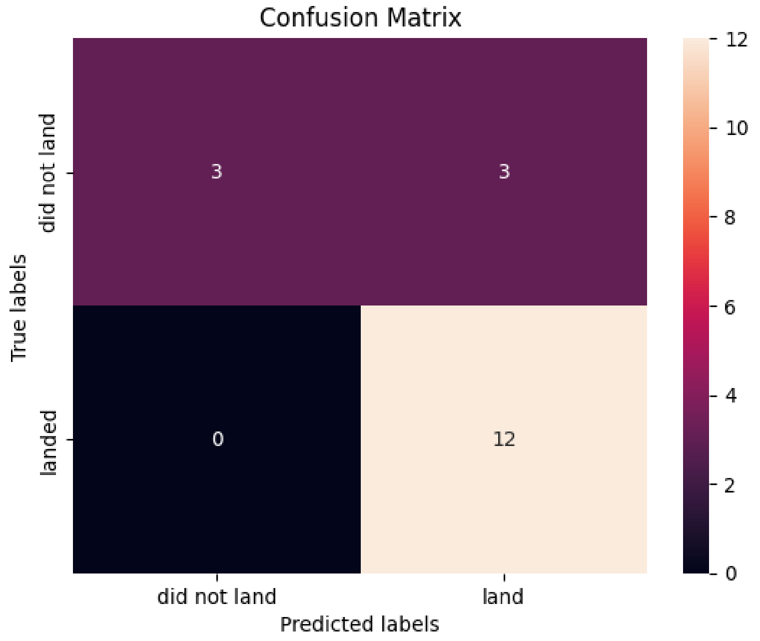
Calculate the accuracy on the test data using the method `score` :

In [24]:
```
print("test set accuracy :",svm_cv.score(X_test, Y_test))
```

```
test set accuracy : 0.8333333333333334
```

We can plot the confusion matrix

In [25]:
```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Thank you!