The Future New Rival of SpaceX

Applied Data Science Capstone – IBM

Act. Aarón Loera

Outline

- Executive Summary
- Introduction
- Methodologies
  - Data Collection (API, Wrangling, Webscrapping)
  - Exploratory Data Analysis (EDA) with SQL and visualization (seaborn and matplotlib)
  - Locations Analysis with Folium
  - Analysis with interactive dashboard with Ploty Dash
  - Predictive Analysis (classification)
- Results
- Conclusion

Executive Summary

Summary of methods

Data Collection API	Data Wrangling	Data Webscraping	EDA with SQL
EDA with Visualization (matplotlib)	Locations Analysis with Folium	Anlysis with interactive dashboard with Ploty Dash	Predictive Analysis (classification)

Introduction

Context:

SpaceX has gained worldwide attention for a series of historic milestones.

It is the only private company ever to return a spacecraft from low-earth orbit, which it first accomplished in December 2010. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars whereas other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage.

Objective:

If we can determine if the first stage will land, we can determine the cost of a launch and be used to bid against SpaceX for a rocket launch.

Methodologies

Data Collection - API

1) Request and parse the SpaceX launch data using the GET request

```
static_json_url='https://cf-courses-data.s3.us.cloud
```

```
response.status_code
```

```
200
```

2) Decode the response content as a Json using .json() and turn it into a Pandas dataframe using .json_normalize()

```
data = pd.json_normalize(response.json())
```

3) Using custom functions to obtain data

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

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

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

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

4) Build our dataset using the data we have obtained. We combine the columns into a dictionary.

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

5) Then, we create a Pandas data frame for Falcon 9 launches from the dictionary.

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

Notebook – GitHub URL

Data Wrangling

Some Exploratory Data Analysis (EDA) was performed to find some patterns in the data and determine what would be the label for training supervised models.

The variable chosen was "Outcome" which contains the following valuable information:

- **True Ocean:** the mission outcome was successfully landed to a specific region of the ocean
- **False Ocean:** the mission outcome was unsuccessfully landed to a specific region of the ocean
- **True RTLS:** the mission outcome was successfully landed to a ground pad
- **False RTLS:** the mission outcome was unsuccessfully landed to a ground pad
- **True ASDS:** the mission outcome was successfully landed on a drone ship
- **False ASDS:** the mission outcome was unsuccessfully landed on a drone ship

In this part our efforts were to find some patterns in the data and determine what would be the label to train supervised models.

- A landing results column was created:

```
In [12]: # landing_outcomes = values on Outcome column
         landing_outcomes = df.value_counts('Outcome')
         landing_outcomes

Out[12]: Outcome
         True ASDS      41
         None None      19
         True RTLS      14
         False ASDS      6
         True Ocean      5
         False Ocean     2
         None ASDS       2
         False RTLS      1
         dtype: int64
```

```
In [14]: bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
         bad_outcomes

Out[14]: {'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0
None None	1	False	False	False	NaN	1.0	0	B1005	-80.577366	28.561857	0
True Ocean	1	False	False	True	NaN	1.0	0	B1006	-80.577366	28.561857	1
True Ocean	1	False	False	True	NaN	1.0	0	B1007	-80.577366	28.561857	1

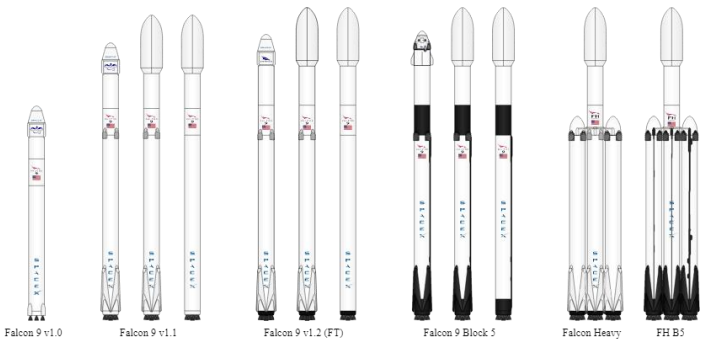
- And then a class column was assigned according to how favorable the landing was, which we will use as a label for our supervised models:

[Notebook - GitHub URL]

Data Webscraping

Web scrap Falcon 9 launch records with BeautifulSoup:

- Extract a Falcon 9 launch records HTML table from Wikipedia (flight no., launch site, payload, payload mass, orbit, customer, launch outcome, version booster, boster landing, date, time)
- Parse the table and convert it into a Pandas data frame



1. Perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response and create a BeautifulSoup object from the HTML response.

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_
response = requests.get(static_url)
data = response.text
soup = BeautifulSoup(data,'html.parser')
```

2. Find all tables on the wiki page first.

```
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]
```

3. Extract column name one by one

```
for j, table_header in enumerate(table_headers):
    name = extract_column_from_header(table_header)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

4. Create an empty dictionary with keys from the extracted column names

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each
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']=[]
```

5. Fill up the launch_dict with launch records extracted from table rows. (check notebook on te URL)

6. Create a dataframe from it

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

Notebook - GitHub

EDA with SQL

1. Load the SQL extension and establish a connection with the database.

```
%load_ext sql
```

```
import csv, sqlite3

con = sqlite3.connect("my_data1.db")
cur = con.cursor()
```

```
!pip install -q pandas==1.1.5
```

```
%sql sqlite:///my_data1.db
```

```
'Connected: @my_data1.db'
```

2.Import the database

```
import pandas as pd
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomai
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method="multi")
```

3. Run the next queries:

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

Notebook - GitHub

EDA with visualization

- Read the SpaceX dataset into a Pandas dataframe and print its summary
- Visualize with scatter and bar graphs between the following relationship:
  - Flight Number and Launch Site
  - Payload and Launch Site
  - Success rate of each orbit type
  - FlightNumber and Orbit type
  - Payload and Orbit type
  - Launch success yearly trend
- Features Engineering: obtain some preliminary insights (FlightNumber, PayloadMass, Orbit, LaunchSite, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial

Notebook - GitHub

Locations Analysis with Folium

- Create a folium Map object and added:
  - Mark all launch sites on a map
  - Mark the success/failed launches for each site on the map
  - Calculate the distances between a launch site to its proximities
- The following patterns are found at the launch sites:
  - Are all launch sites in proximity to the Equator line? **YES**
  - Are all launch sites in very close proximity to the coast? **YES**
  - Are launch sites in close proximity to railways? **YES**
  - Are launch sites in close proximity to highways? **YES**
  - Are launch sites in close proximity to coastline? **YES**
  - Do launch sites keep certain distance away from cities? **YES**

Notebook – IBM Watson Studio

Notebook - GitHub URL

Analysis with interactive dashboard with Ploty Dash

- The dashboard application contains input components such as:
  - Dropdown list
  - Range slider to interact with:
  - Pie chart
  - Scatter point chart.

- For this purpose, the following was carried out
  - Add a Launch Site Drop-down Input Component
  - Add a callback function to render success-pie-chart based on selected site dropdown
  - Add a Range Slider to Select Payload
  - Add a callback function to render the success-payload-scatter-chart scatter plot

Notebook - GitHub URL
(code only)

Predictive Analysis (classification)

- Perform exploratory Data Analysis and determine Training Labels
  - create a column for the class
  - Standardize the data
  - Split into training data and test data

- Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
  - Find the method performs best using test data

Notebook - GitHub

Results

EDA wit SQL

Launch Sites

Query:

```
%sql SELECT Launch_Site FROM SPACEXTBL GROUP BY Launch_Site
 * sqlite:///my_data1.db
Done.
```

Results:

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

- The names of the unique launch sites in the space misión are 4, which are: CCAFS LC-40, CCADS SLC-40, KSC LC-39A and VAFB SLC-4E

Launch Sites begin with CCA

Query:

```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE ('CCA%') LIMIT 5
```

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

- 5 records where launch site begin with the string ‘CCA’

Results:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload mass from NASA (CRS)

Query:

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER ='NASA (CRS)'
 * sqlite:///my_data1.db
Done.
```

Results:

SUM(PAYLOAD_MASS__KG_)
45596

- The total payload mass carried by NASA's rockets launched by NASA (CRS) is **45,596 Kg**

Average Payload mass from booster version F9 v1.1

Query:

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version ='F9 v1.1'
```

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

Results:

SUM(PAYLOAD_MASS__KG_)
14642

- The average payload mass carried by booster version F9 v1.1 is **14,642 Kg**

First successful landing outcome in ground pad

Query:

```
%%sql SELECT min([DATE]) AS Min_Date, Landing_Outcome FROM SPACEXTBL
WHERE Landing_Outcome = 'Success (ground pad)'
```

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

Results:

Min_Date	Landing_Outcome
2015-12-22 00:00:00	Success (ground pad)

- The date of the first successful landing on the runway was on **22/12/2015**

Successful Boosters in drone ship with specified payloads

Query:

```
%%sql SELECT Booster_Version, [Landing_Outcome], PAYLOAD_MASS__KG_ FROM SPACEXTBL
WHERE [Landing_Outcome] = 'Success (drone ship)'
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000
```

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

Results:

Booster_Version	Landing_Outcome	PAYLOAD_MASS__KG_
F9 FT B1022	Success (drone ship)	4696.0
F9 FT B1026	Success (drone ship)	4600.0
F9 FT B1021.2	Success (drone ship)	5300.0
F9 FT B1031.2	Success (drone ship)	5200.0

- There are **4** records of different booster version that have been successful in drone ship and have payload mass between **4,000 and 6,000 kg**

Total number of successful and failure mission outcomes

Query:

```
%%sql SELECT  Mission_Outcome, COUNT(Mission_Outcome)
FROM SPACEXTBL GROUP BY Mission_Outcome
```

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

Results:

Mission_Outcome	COUNT(Mission_Outcome)
None	0
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- According to the result, SpaceX seems to have successfully completed nearly **99%** of its missions

Booster versions which have carried the maximum payload mass

Query:

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

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

- According to the result, the booster version which have carried the maximum payload mass are the version ***F9 B5 10…***

Results:

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

Failure records of drone ship in 2015

Query:

```
%%sql SELECT substr(Date,6,2)MONTH, [Landing_Outcome], BOOSTER_VERSION, Launch_Site, DATE
FROM SPACEXTBL  WHERE [Landing_Outcome] = 'Failure (drone ship)' AND substr(Date,1,4)='2015'
-- Note: I used other "substr" because I have other date format
```

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

Results:

MONTH	Landing_Outcome	Booster_Version	Launch_Site	Date
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015-01-10 00:00:00
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015-04-14 00:00:00

- Records show the launch site with a failed landing in drone ship in 2015 was **CCAFS LC-40** in **January and April** of the same year

Rank Successful Landing Outcome

Query:

```
%%sql
SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS COUNT_ FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' AND LANDING_OUTCOME LIKE '%SUCCESS%'
GROUP BY LANDING_OUTCOME ORDER BY COUNT_ DESC
```

Results:

Landing_Outcome	COUNT_
Success (drone ship)	5
Success (ground pad)	3

- Records show the rank of count of successful landing_outcomes between the date **04-06-2010** and **20-03-2017** in descending order.
- As can be seen, there have been more successful drone ship landings than on ground pad.

Results

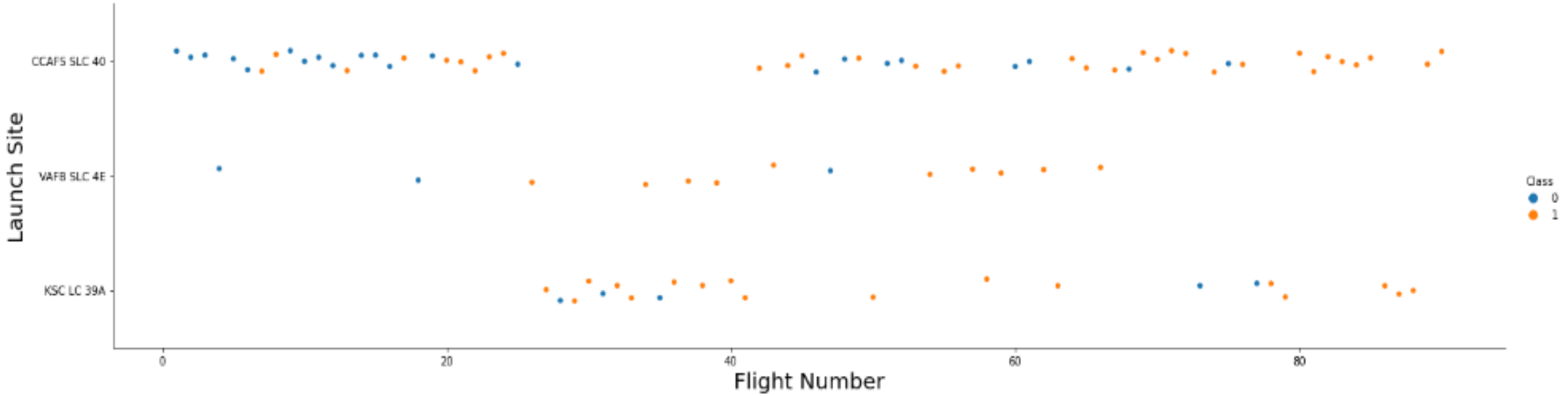
EDA with visualization

Flight Number vs Launch Site

Code:

```
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 4)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

Result:



Class 0 means unsuccessful launch and **Class 1** represent successful one

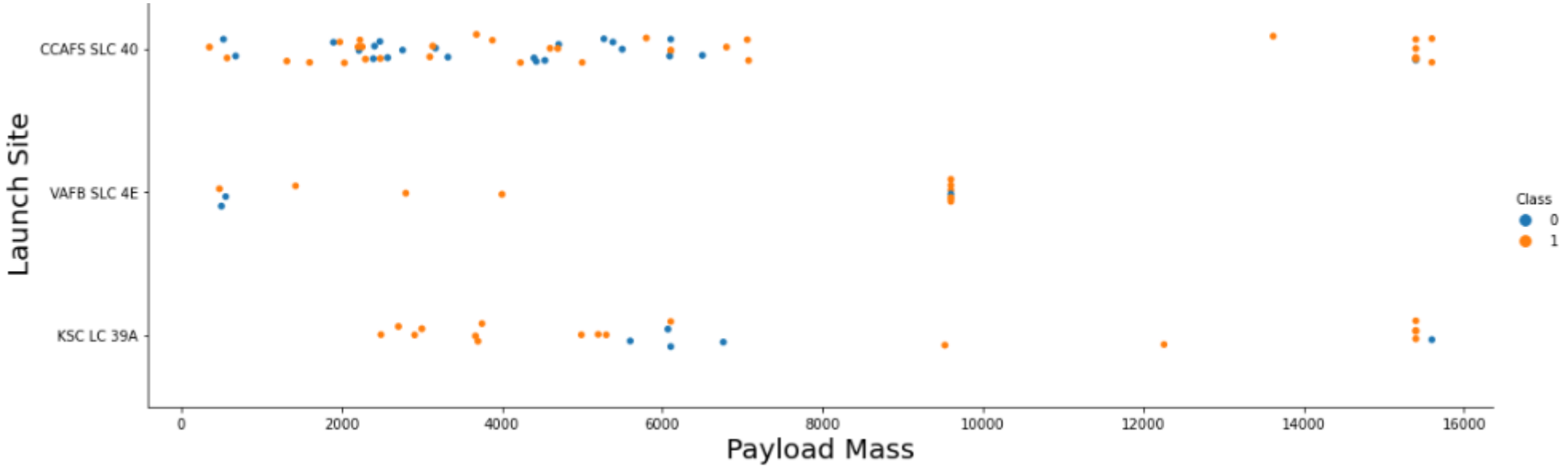
The graph shows that as the time progresses and the number of flights increases, the **success rate will grow**, also the most used launch site since the beginning is the CCAFS SLC 40.

Pay Load vs Launch Site

Code:

```
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 3)
plt.xlabel("Payload Mass",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

Result:



Class 0 means unsuccessful launch and **Class 1** represent successful one

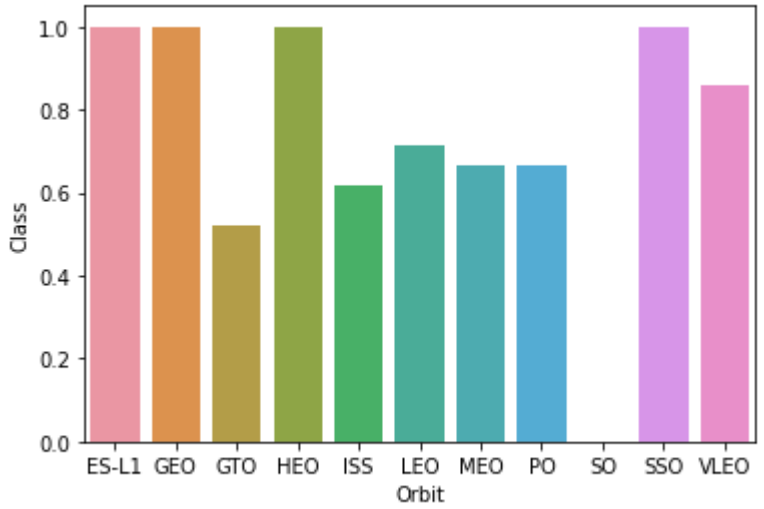
The graph shows the **VAFB-SLC** launch site don´t have rockets launched for heavy payload mass(greater than 10000)

Success rate of each orbit type

Code:

```
orbit_success = df.groupby('Orbit').mean()
#orbit_success
orbit_success.reset_index(inplace=True)
sns.barplot(x="Orbit", y="Class", data=orbit_success)
```

Result:



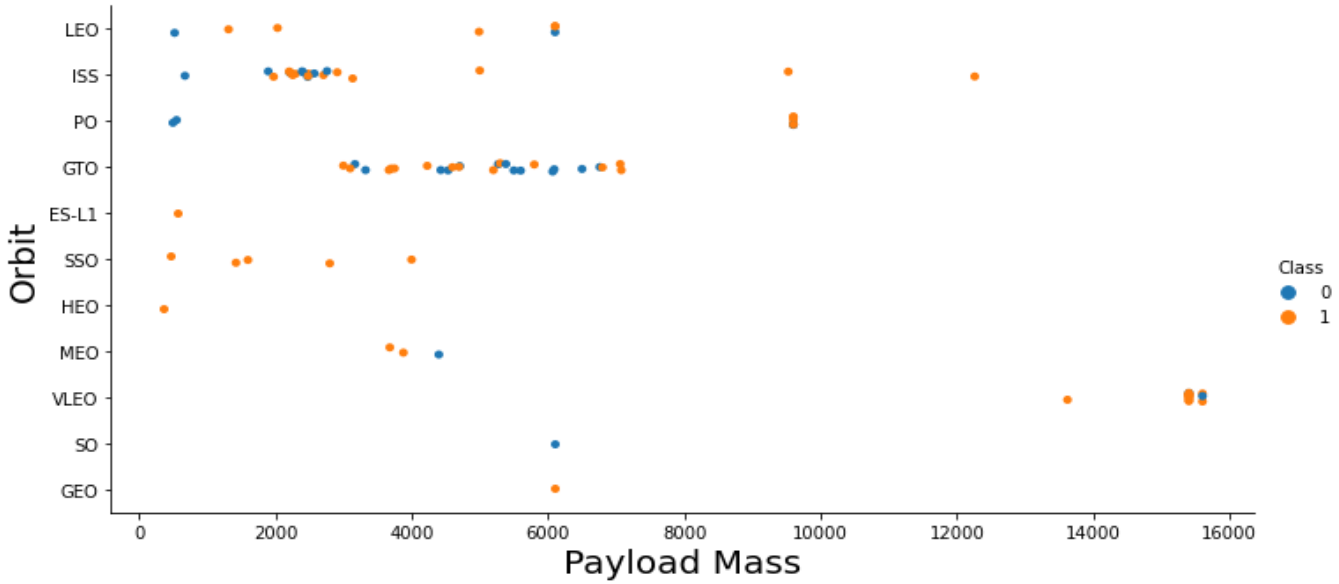
We can see that the orbits with the highest success rate are ES-L1, GEO, HEO and SSO (100%). While GTO, ISS, MEO and PO are the lowest.

Pay Load vs Orbit Type

Code:

```
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 2)
plt.xlabel("Payload Mass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

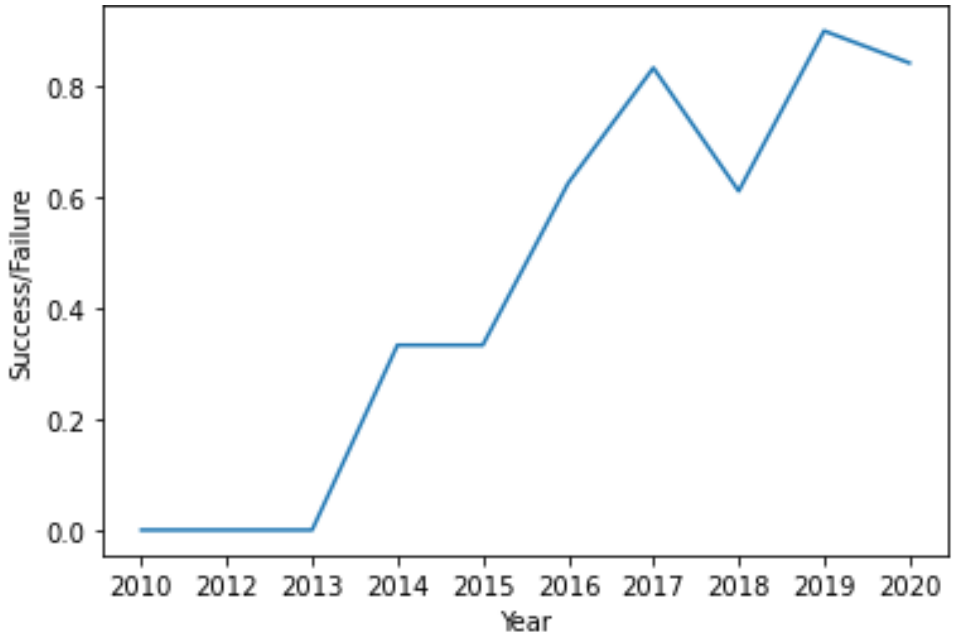
Result:



Class 0 means unsuccessful launch and **Class 1** represent successful one

In the **LEO, PO** and **ISS** orbits the success landing appears related to the payload mass; on the other hand, there seems to be no relationship between flight number when in **GTO** orbit.

Launch success yearly trend



We can observe that the first years of launches there was not much success in the landings, but from 2013 onwards the success increased exponentially. In 2018 there was a small recession but in 2019 it recovered.

Results

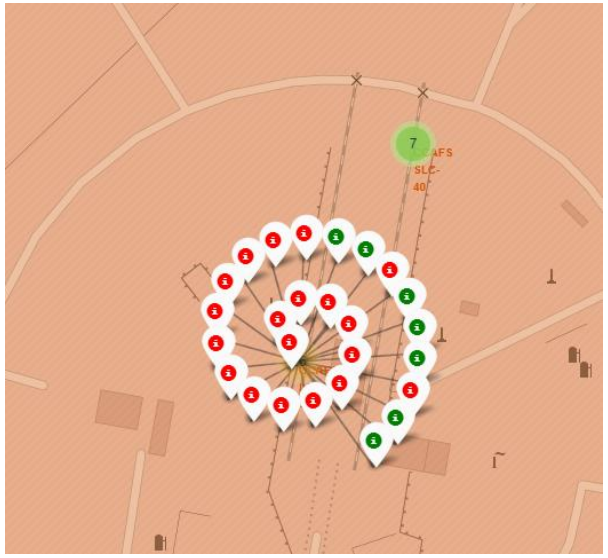
Locations Analysis with Folium

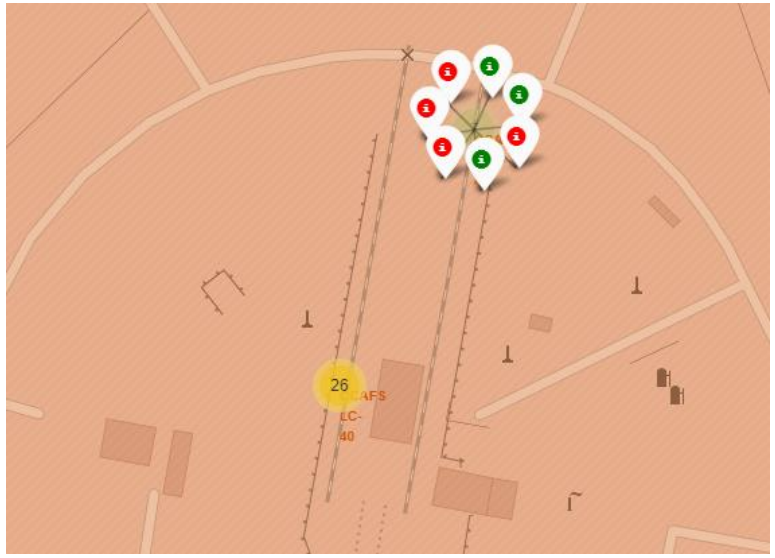
Launch Sites on a map

The generated map is shown with the launching points marked.

All launch sites are near the equator, at the southernmost point of the United States and near the coast.

Success/failed launches marks

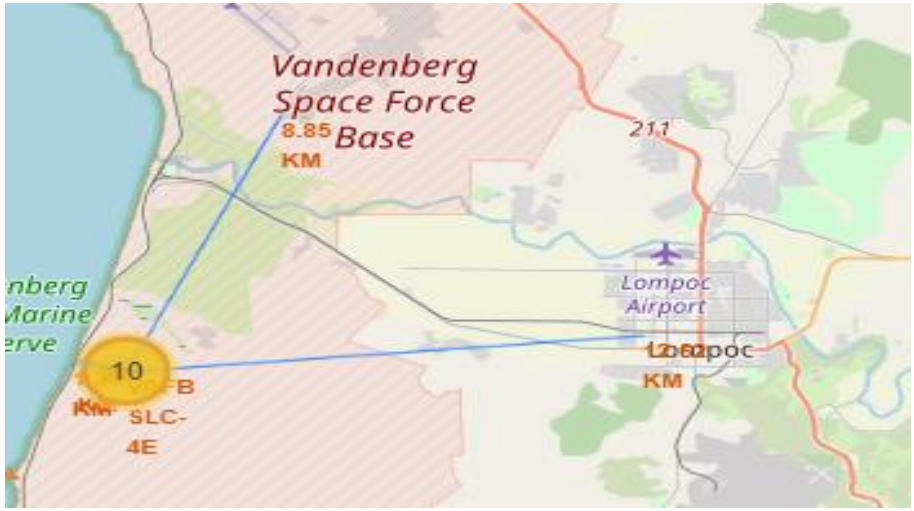




With the cluster mark you can see that in Florida there have been more launches (46) than in California (10).

We can also see in green color the successful launches compared to the unsuccessful ones.

Distances between a launch site to its proximities

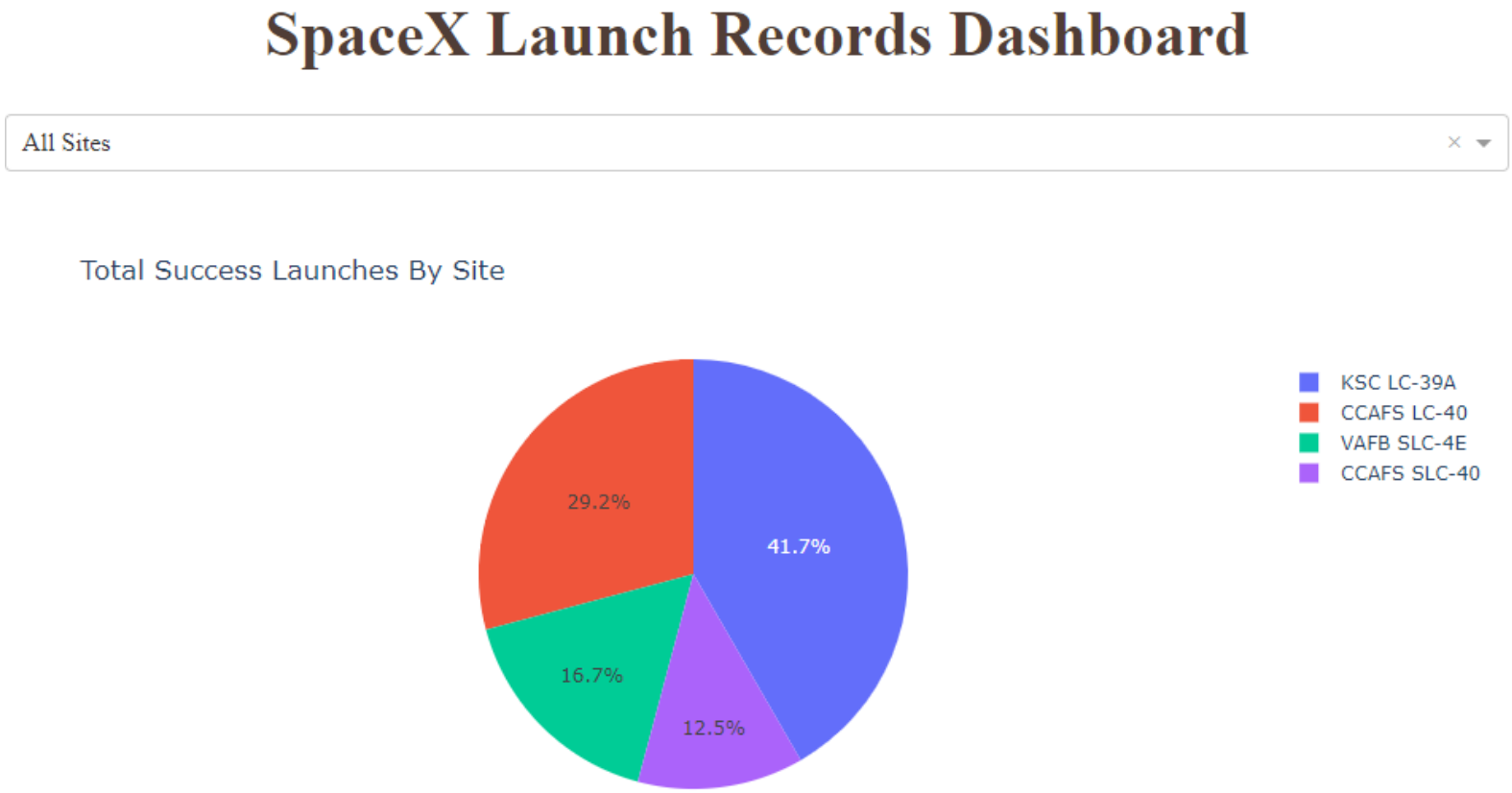


We can observe that the launch sites are located very close to railways, highways and the coast, probably to supply supplies efficiently, and they are also kept at a certain distance from the cities for security reasons.

Results

Interactive dashboard with Ploty Dash

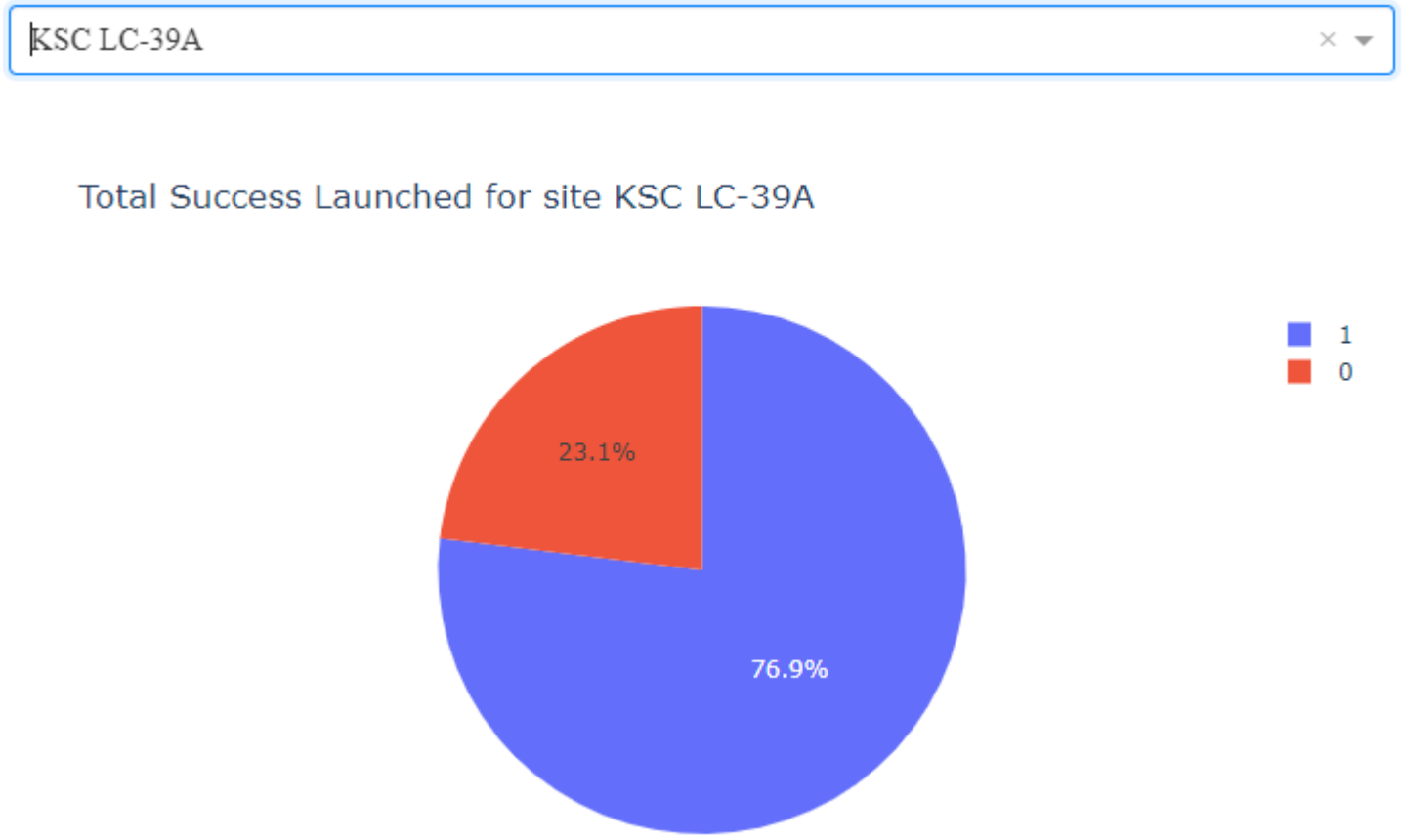
Total Success Launches By Site



As we can see, **KSC LC-39A** contains the records with the highest percentage of successful landings (**41.7%**) .

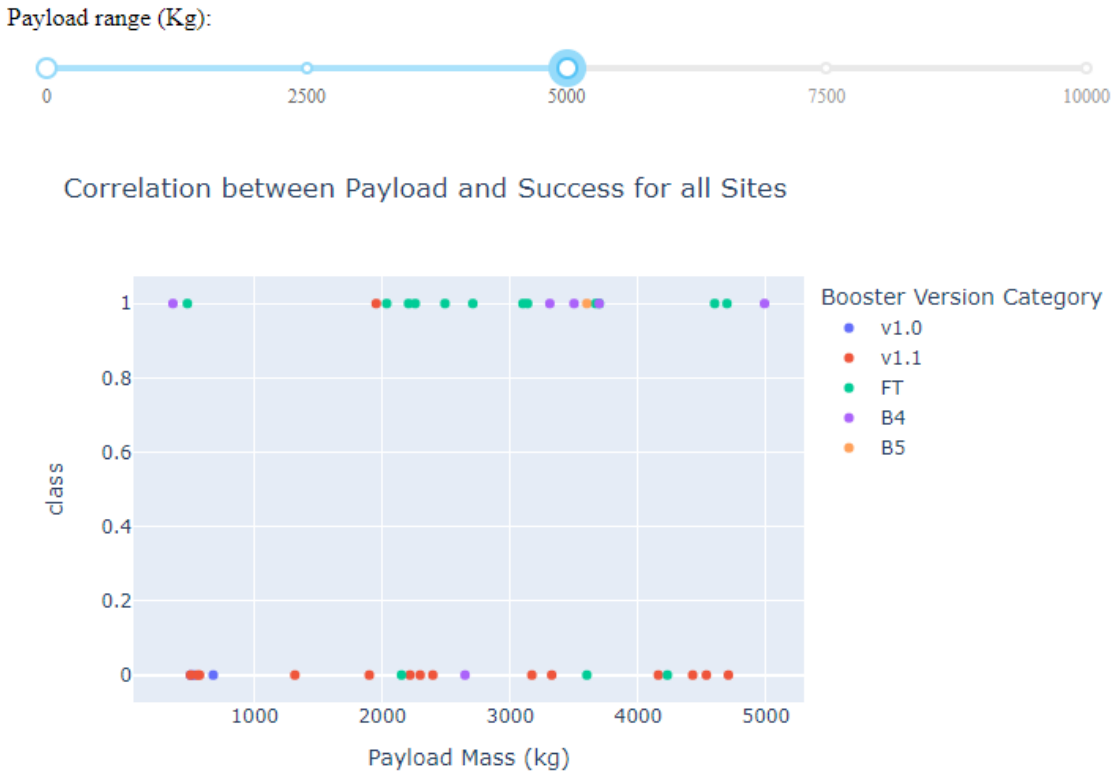
It is followed by **CCAFS LC-40** with a **29.2%** success rate in its launches.

Total Success Launched for site KSC LC-39A



For **KSC LC-39A** who has the highest percentage of successful landings, we see that it is divided into 10 (**76.9%**) good landings versus 3 (**23.1%**) unsuccessful landings.

Payload Success for all sites

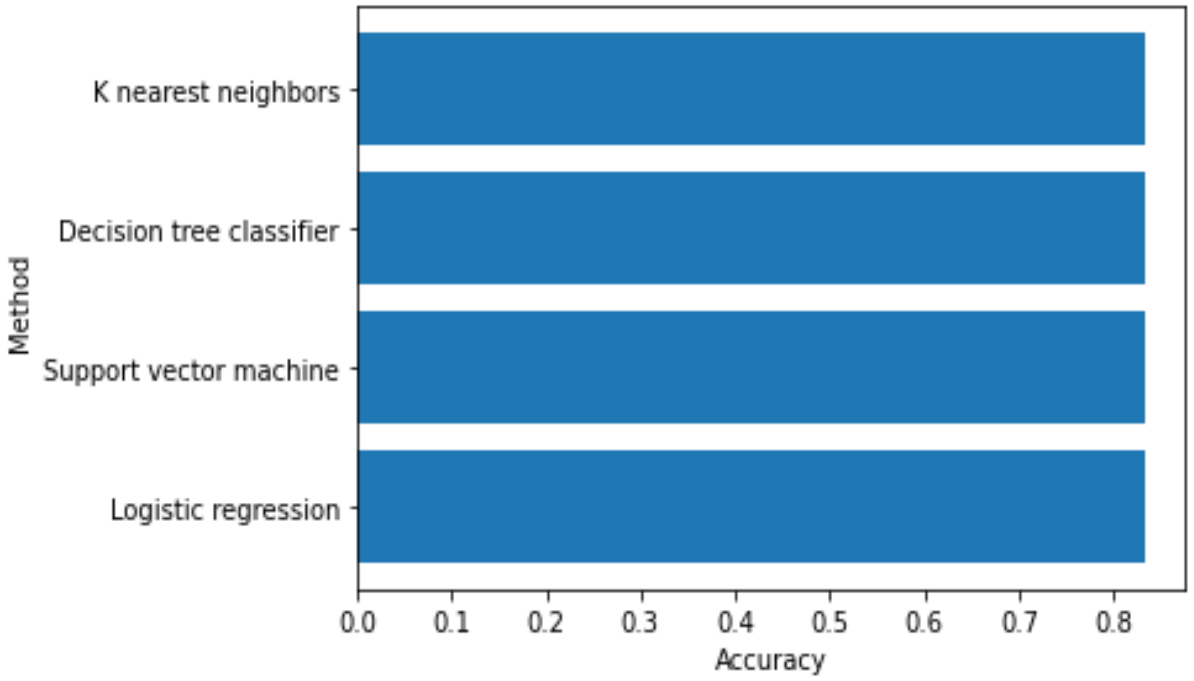


If we move the range slider to select the payload in two parts: **light** (0-5,000 kg) and **heavy** (5,000-10,000 kg) to notice that its success rate for the **lighter ones** is much higher.

Results

Predictive Analysis - Classification

Accuracy of Classification Models



Four models have been selected to compare their accuracy: **KNN, Decision tree classifier, SPV and Logistic regression**.

Using the method score for each of them we can see that it gives the same which is **83.34%**.

We believe that even the database is still small to obtain the optimal model.

Confusion Matrix

The models predicted **12** true positives (**TP**) and **0** false negatives (**FN**). Which is good.

While the model predicted **3** true positives (**TP**), there were also **3** predictions that said successful landings when the true label was failure (**FP**).

Conclusions

- As the number of flights increased, the success rate increased, and recently it has exceeded 80%.
- Orbital types SSO, HEO, GEO, and ES-L1 have the highest success rate (100%).
- The launch site is close to railways, highways, and coastline, but far from cities.
- KSLC-39A has the highest number of launch successes and the highest success rate among all sites.
- The launch success rate of low weighted payloads is higher than that of heavy weighted payloads.
- In this dataset, all models have the same accuracy (83.33%), but it seems that more data is needed to determine the optimal model due to the small data size.

and so as the professor says: