IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Diego Vižintin

07.01.2023

Outline

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

Executive Summary

- Summary of methodologies

- Data Collection troughd API
- Dana Collection with Webscraping
- Data Wrangling
- Exploratory Data Analysis with Data visualisation
- Exploratory Data Analysis with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly
- Macheene Learning prediction
- Predictive Analysis (Classification)

- Summary of all results

- Exploratory Data Analysis result
- Interactive analytics in screenshots
- Predictive Analytics result

Introduction

- Project background and context

SpaceX is the most successful company of the commercial space age, making space travel affordable. The company 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 that the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Problems that we need to answer

What influences if the rocket will land successfully?

The effect each relationship with certain rocket variables will impact in determining the success rate of a successful landing.

What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected useing SpaceX API and webscraping from Wikipedia.
- Perform data wrangling
  - One-hot encodeing was applied to categorical features.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

The data collection process involved a combination of API requests from SpaceX REST API and Web Scraping data from a table in SpaceX's Wikipedia entry.

We had to use both of these data collection methods in order to get complete information about the launches for a more detailed analysis.

Data Columns are obtained by using SpaceX REST API:

Flight Number, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude, Latitude

Data Columns are obtained by using Wikipedia Web Scraping:

Flight Number, Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Version Booster, Booster landing, Date, Time

Data Collection – SpaceX API

- We used the get request to the Spacex API to collect data, clean the requested dana and did some basic dana wrangling and formatting.
- The link to the notebook is https://github.com/diegx/Coursera-IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

In [29]:
```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

In [30]:
```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

In [31]:
```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, wheter the core is reused, wheter legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

In [32]:
```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Now let's start requesting rocket launch data from SpaceX API with the following URL:

In [33]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [34]:
```
response = requests.get(spacex_url)
```

Data Collection - Scraping

- We applied web scrapping to webscrape Falcon 9 launch records with BeautifulSoup
- We parsed the launch record values and converted them into a pandas dataframe
- The link to this notebook is https://github.com/diegx/Coursera-IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

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]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

Out[5]:
```
200
```

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

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

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

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<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 external reference link towards the end of this lab

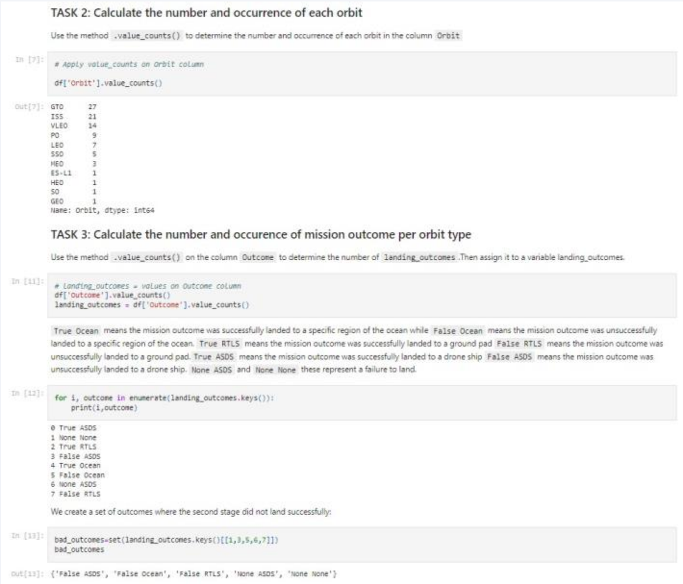
In [8]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

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

In [9]:
```
# 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%;">
<tbody><tr>
<th scope="col">Flight No.
```

Data Wrangling


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 [7]:
```
# Apply value_counts on Orbit column

df['Orbit'].value_counts()
```

Out[7]:
```
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 [11]:
```
# landing_outcomes = values on Outcome column
df['Outcome'].value_counts()
landing_outcomes = df['Outcome'].value_counts()
```

`True Ocean` means the mission outcome was successfully landed to a specific region of the ocean while `False Ocean` means the mission outcome was unsuccessfully landed to a specific region of the ocean. `True RTLS` means the mission outcome was successfully landed to a ground pad `False RTLS` means the mission outcome was unsuccessfully landed to a ground pad. `True ASDS` means the mission outcome was successfully landed to a drone ship `False ASDS` means the mission outcome was unsuccessfully landed to a drone ship. `None ASDS` and `None None` these represent a failure to land.

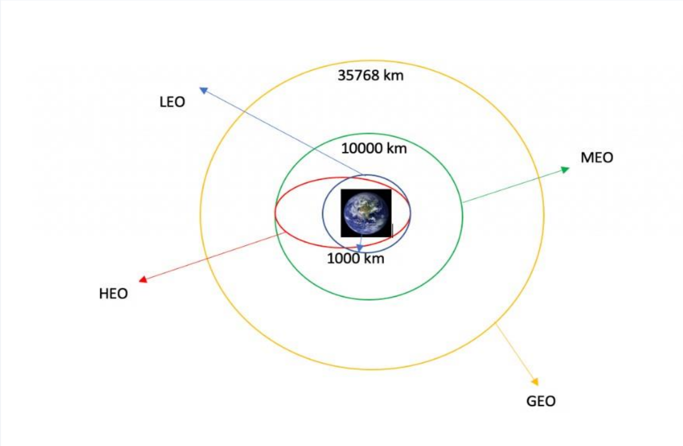
In [12]:
```
for i, outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

We create a set of outcomes where the second stage did not land successfully:

In [13]:
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

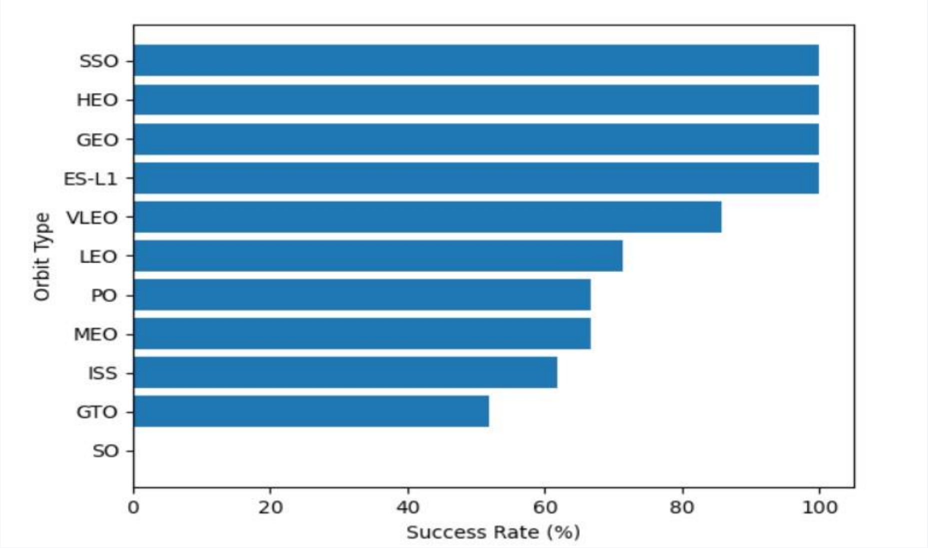
Out[13]: {'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}

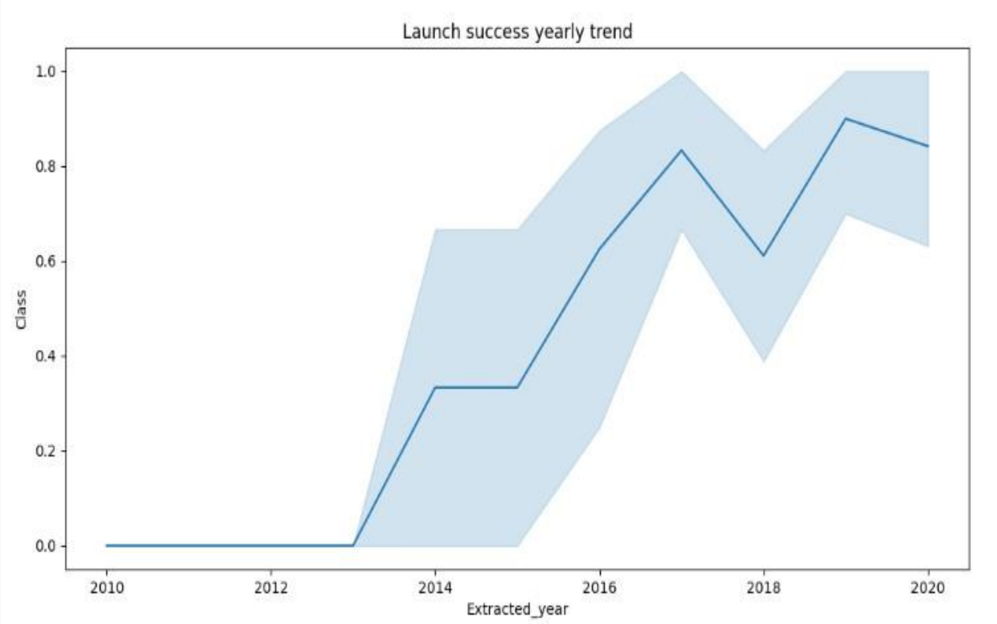


- Exploratory data analysis was performed to determine the training labels
- We calculated the number of launches at each site and the number of orbits and ocurances of each orbit
- We than created a landing outcome label from the outcome column.
- The link to this notebook is https://github.com/diegx/Coursera-IBM-Applied-Data-Science-Capstone/blob/main/Lab%202%20Data%20wrangling.ipynb

EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type and the yearly launch success trend.





- The link to this notebook is https://github.com/diegx/Coursera-IBM-Applied-Data-Science-Capstone/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The names of the booster versions which have carried the maximum payload mass.
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to this notebook is https://github.com/diegx/Coursera-IBM-Applied-Data-Science-Capstone/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.i.e., 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 question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.
- The link to this notebook is https://github.com/diegx/Coursera-IBM-Applied-Data-Science-Capstone/blob/main/Foliuma%20lab.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 graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- We added range sliders.
- Explain why you added those plots and interactions

different launch sites and we would like to first see which one has the largest success count. Then, we would like to select one specific site

to find if variable payload is correlated to mission outcome. From a dashboard point of view, we want to be able to easily select different payload range and see if we can identify some visual patterns.

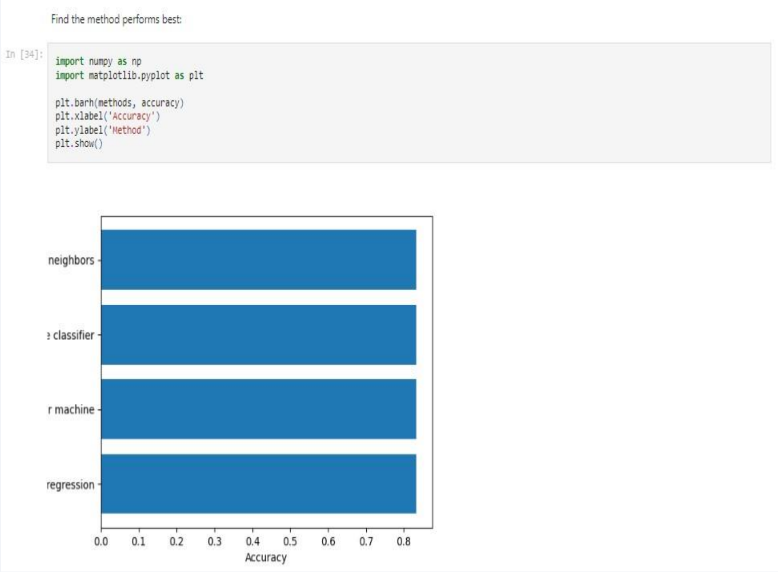
As such, we can visually observe how payload may be correlated with mission outcomes for selected site(s).

In addition, we want to color-label the Booster version on each scatter point so that we may observe mission outcomes with different boosters.

- The link to the notebook is
  https://github.com/diegx/Coursera-IBM-Applied-Data-Science-Capstone/blob/1b1abcdf59ae41bbce94169c8202eca1a9345087/spacex_dash_app.py

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook
  https://github.com/diegx/Coursera-IBM-Applied-Data-Science-Capstone/blob/main/SpaceX_Machine_Learning.jupyterlite.ipynb



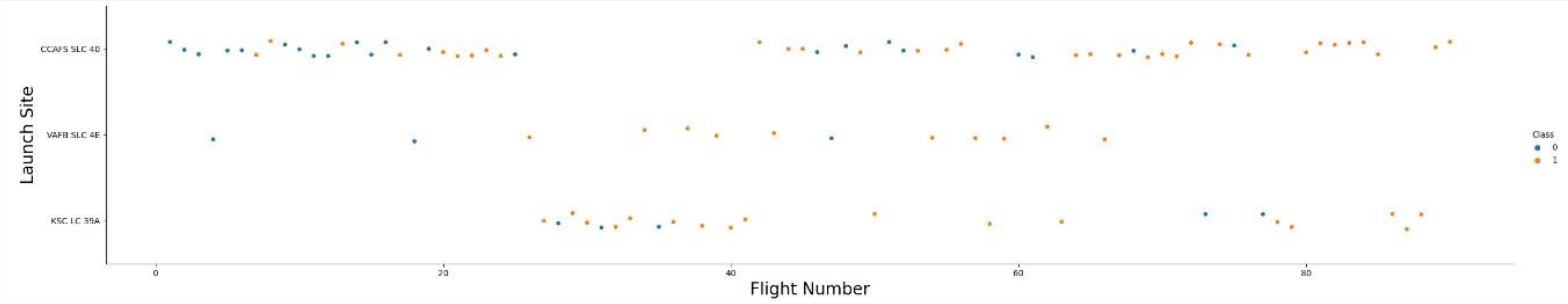
Results

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

Section 2

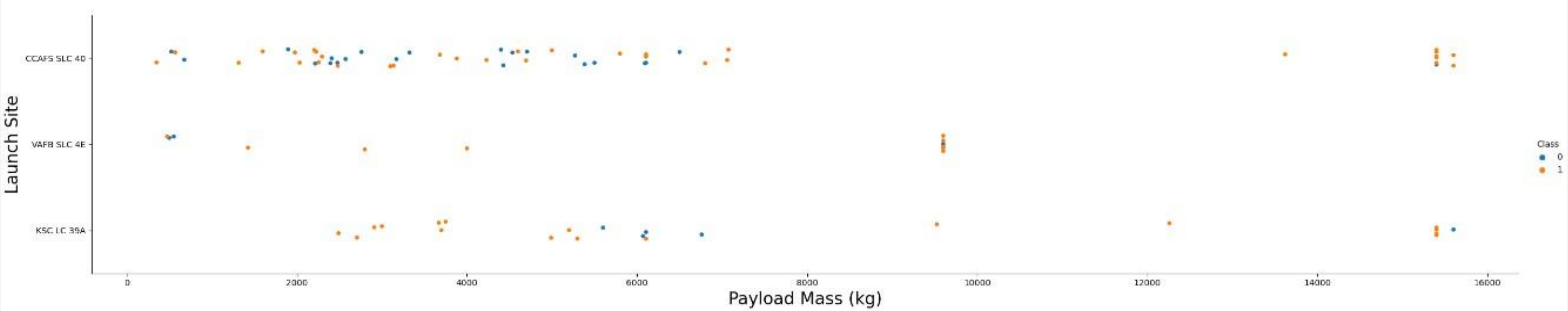
Insights drawn from EDA

Flight Number vs. Launch Site



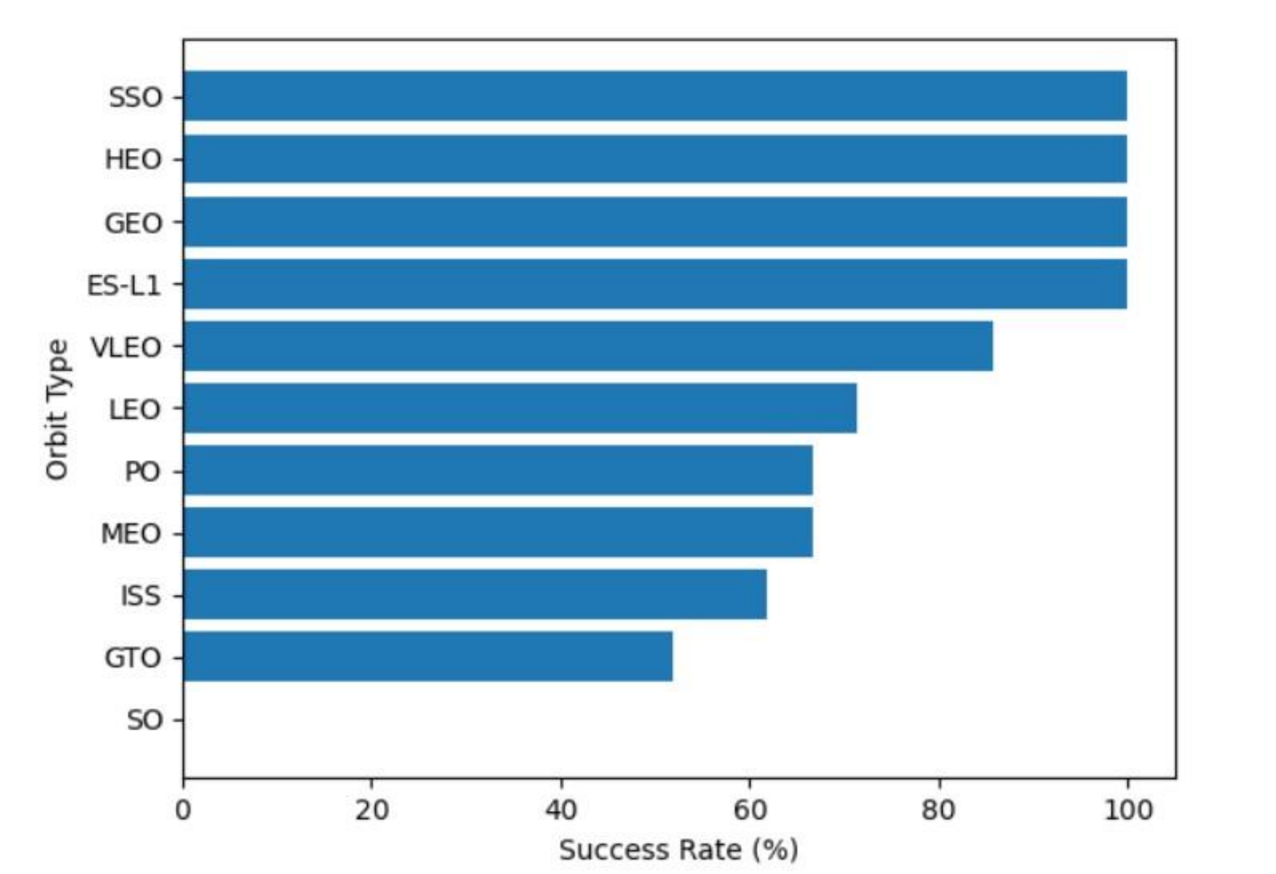
- VAFB SLC 4E and KSC LC 39A have higher success rates
- The CCAFS SLC 40 launch site has about a half of all launches.
- The number of launches varies between each launch site
- It can be assumed that wit each new launch at a launch site there is a higher rate of success.

Payload vs. Launch Site



- The greater the payload mass the greater the succes at a launch site.
- At the VAFB SLC 4E launch site there were no launches above 10000 kg.
- KSC LC 39A has a 100% success rate for payload mass under 5500 kg.
- Most lunches with a payload mass greater than 6000kg were sucessfull.

Success Rate vs. Orbit Type



- Orbits with a success rate of 100%

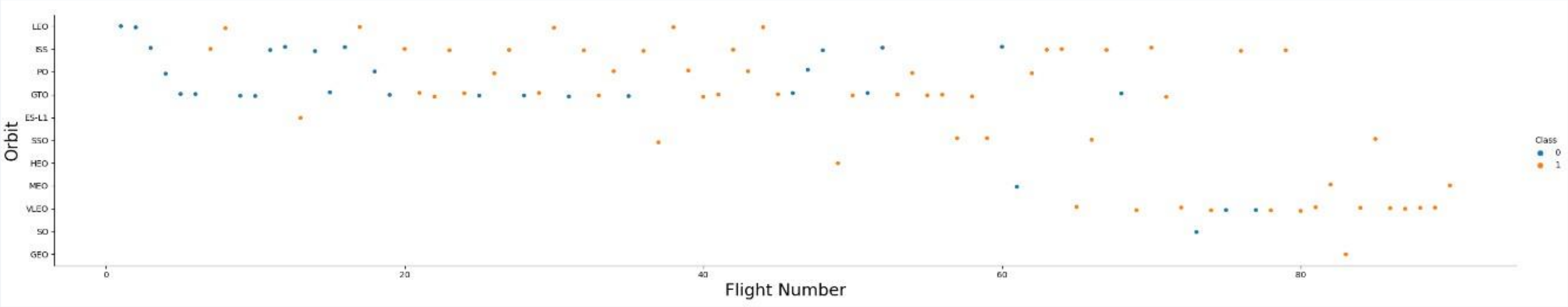
  SSO, HEO, GEO, ES-L1
- Orbits with a success rate above 50%

  VLEO, LEO, PO, MEO, ISS
- Orbits wit a success of 50%

  GTO
- Orbits with a 0% success rate

  SO

Flight Number vs. Orbit Type



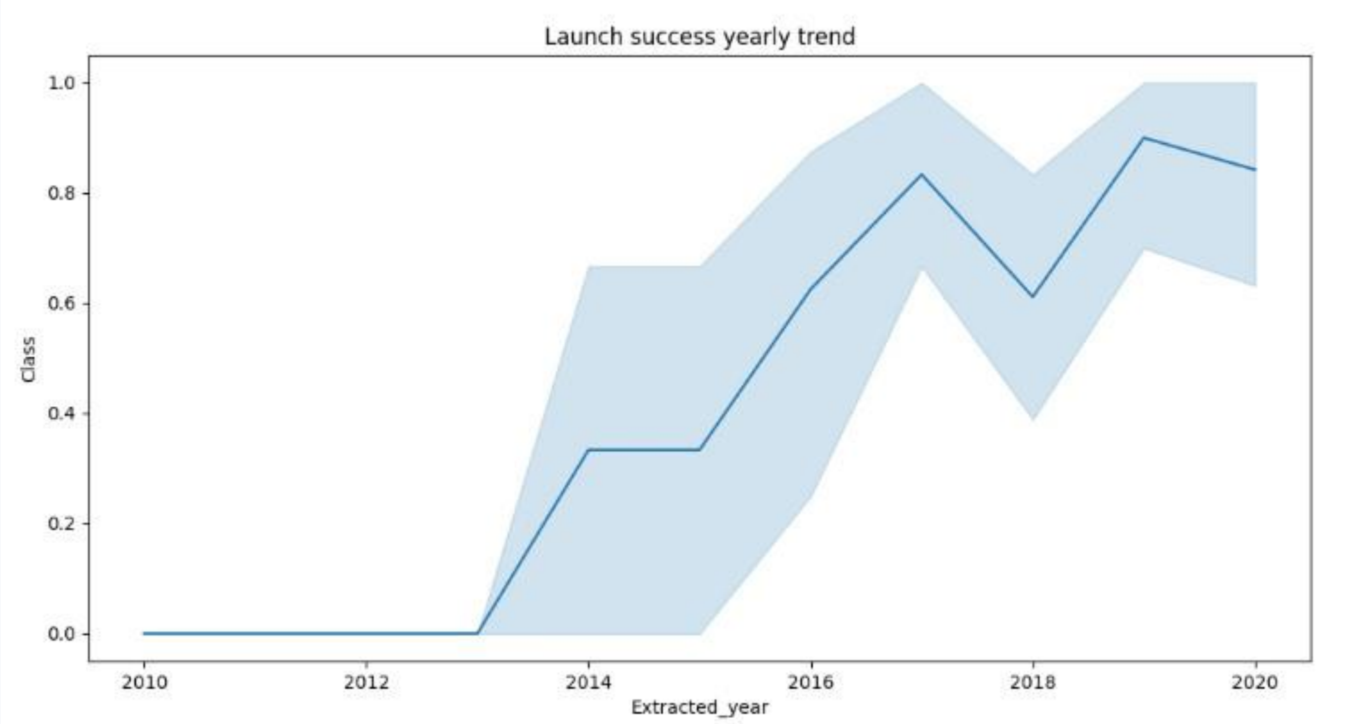
- In the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

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 here.

Launch Success Yearly Trend

- The success rate increased in the period from 2013 to 2020



All Launch Site Names

- Displaying the names of the unique launch sites in the space mission.

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1;
```

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

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

Launch Site Names Begin with 'CCA'

- Displaying the launchsite names that begin with ‘CCA’

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

In [16]:
```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

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

Out[16]:

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

- We used the query below to display the total payload mass carried by the boosters launched by NASA (CRS)

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

In [17]:
```
sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD FROM SPACEXTBL WHERE PAYLOAD  LIKE '%CRS%';
```

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

Out[17]:

TOTAL_PAYLOAD
111268

Average Payload Mass by F9 v1.1

- We used the query below to display the average payload mass carried by booster version F9 v1.1

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

```
In [18]: sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_PAYLOAD FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';

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

Out[18]:

AVG_PAYLOAD
2928.4

First Successful Ground Landing Date

- We used the query below to display the first successful ground landing date

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

Hint:Use min function

```
[44]: %sql SELECT MIN(DATE) AS FIRST_SUCCESSFUL_LANDING_OUTCOME FROM SPACEXTBL WHERE "Landing _Outcome" = 'Success (ground pad)'
```

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

[44]:

FIRST_SUCCESSFUL_LANDING_OUTCOME
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

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

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

```
%%sql
SELECT Booster_Version FROM SPACEXTBL WHERE "Landing _Outcome" ='Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4001 and 5999
```

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

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- The total number of successful and failure mission outcomes.
- 100 successful mission outcomes and 1 faliure

List the total number of successful and failure mission outcomes

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

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

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

Boosters Carried Maximum Payload

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

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

[14]:
```
%sql SELECT DISTINCT(BOOSTER_VERSION) FROM SPACEXTBL \
  WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

[14]:

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

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 substr(Date, 4, 2) as month,booster_version,"Landing _Outcome",launch_site
from SPACEXTBL where "Landing _Outcome"='Failure (drone ship)' and substr(Date,7,4)='2015';
```

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

month	Booster_Version	Landing _Outcome	Launch_Site
01	F9 v1.1 B1012	Failure (drone ship)	CCAFS LC-40
04	F9 v1.1 B1015	Failure (drone ship)	CCAFS LC-40

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

We counted the sucessful outcomes between 04.06.2010 and 20.03.2017

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
%sql SELECT "Date", "LANDING _OUTCOME", COUNT ("LANDING _OUTCOME")  FROM SPACEXTBL where "Date"  between '04-06-2010' and '20-03-2017' and "
```

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

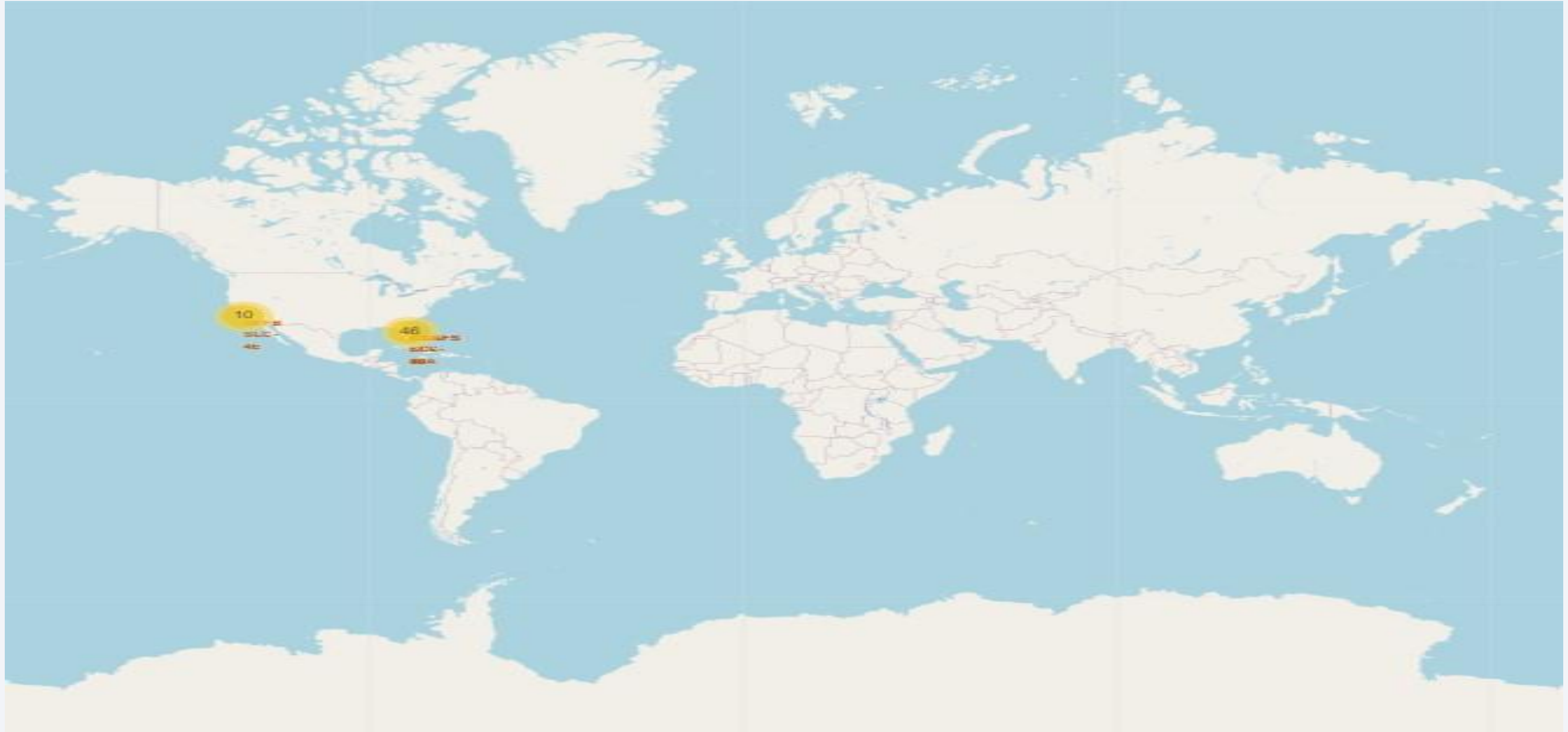
Date	Landing _Outcome	COUNT ("LANDING _OUTCOME")
07-08-2018	Success	20
08-04-2016	Success (drone ship)	8
18-07-2016	Success (ground pad)	6

Section 3

Launch Sites Proximities Analysis

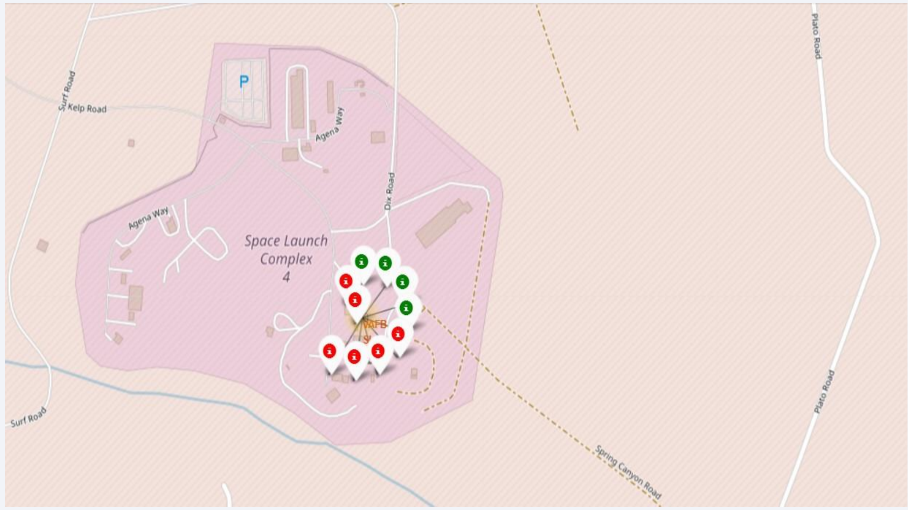
All launch sites global map markers

- All SpaceX sites are found in the United States of America.



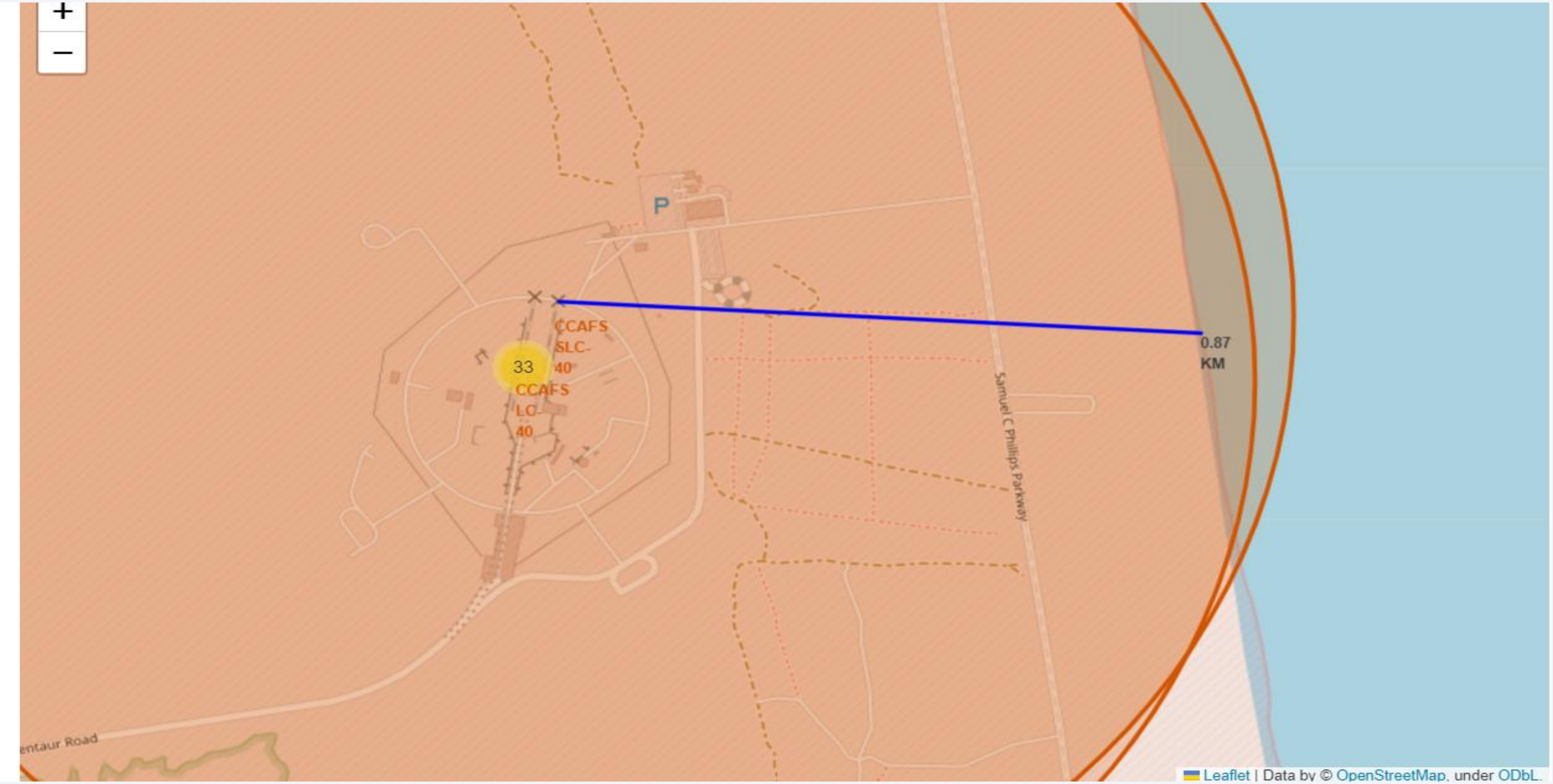
Markers showing launch sites with color labels

- Green markers show successful launches red show failed ones



Launch Site distance to coastline

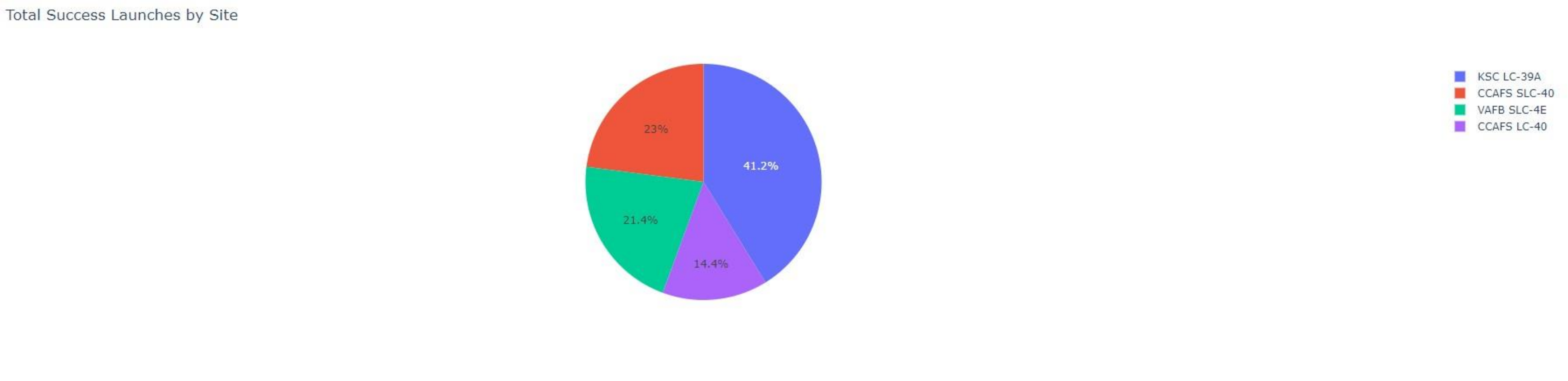
The distance from the CCAFC-SLC-40 is 0.87 km



Section 4

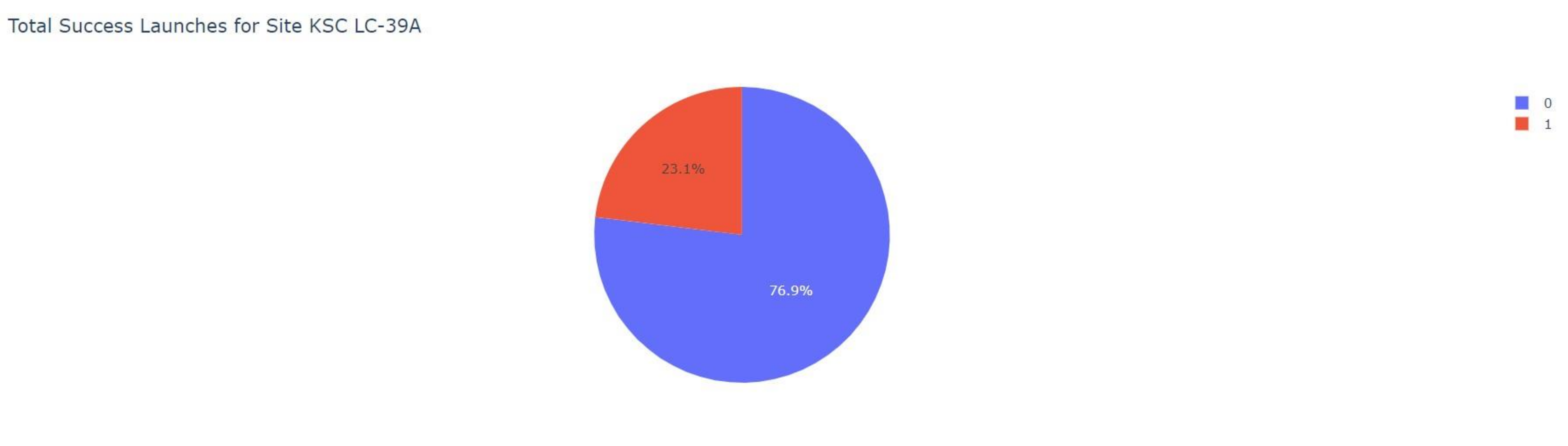
Build a Dashboard with Plotly Dash

Pie chart showing the success count for all sites



- The chart clearly shows that from all the sites, KSC LC-39A has the most successful launches.

Pie chart showing the Launch site with the highest launch success ratio



KSC LC-39A has the highest launch success rate (76.9%) with 10 successful and only 3 failed landings.

Payload vs. Launch Outcome for all sites

- The charts show that payloads between 2000 and 5500 kg have the highest success rate.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
[22]: parameters = {'criterion': ['gini', 'entropy'],
           'splitter': ['best', 'random'],
           'max_depth': [2*n for n in range(1,10)],
           'max_features': ['auto', 'sqrt'],
           'min_samples_leaf': [1, 2, 4],
           'min_samples_split': [2, 5, 10]}

      tree = DecisionTreeClassifier()
```

```
[23]: tree_cv = GridSearchCV(tree,parameters,cv=10)
      tree_cv.fit(X_train, Y_train)
```

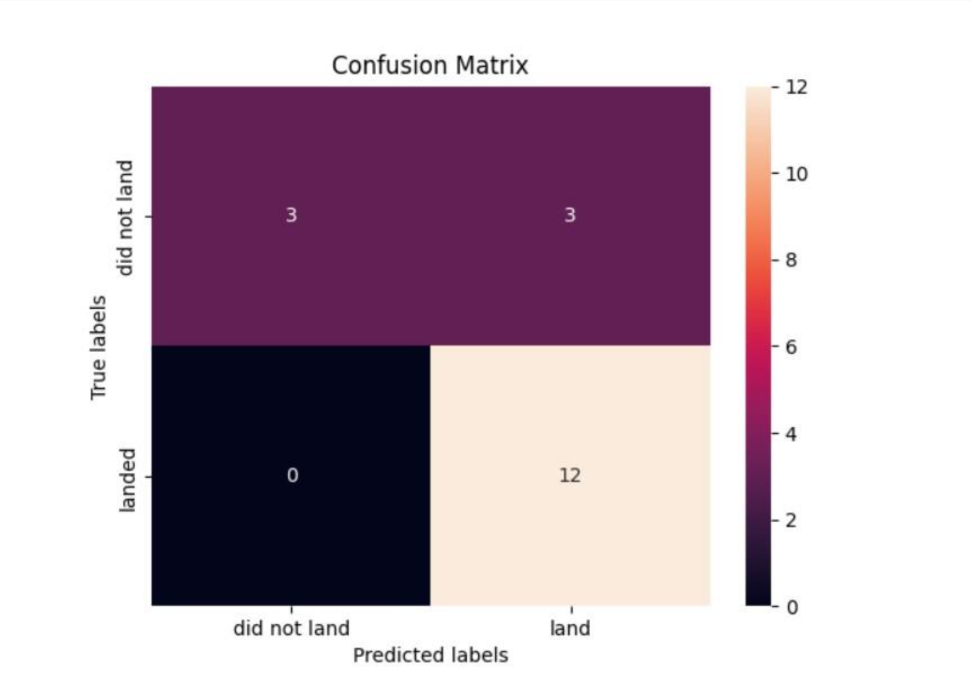
```
[23]: GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
                   param_grid={'criterion': ['gini', 'entropy'],
                               'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                               'max_features': ['auto', 'sqrt'],
                               'min_samples_leaf': [1, 2, 4],
                               'min_samples_split': [2, 5, 10],
                               'splitter': ['best', 'random']})
```

```
[24]: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
      print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 1
0, 'splitter': 'random'}
accuracy : 0.875
```

Confusion Matrix

- Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the major problem is false positives.



Conclusions

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- 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 of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!