Applied Data Science Capstone

Andrea D'Alcantara

08/01/2022

EXECUTIVE SUMMARY

- Summary of methodologies
  - Data collection with an API
  - Data collection with web scraping
  - Data wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with data visualization
  - Interactive visual analytics with Folium
  - Interactive Dashboard with Plotly Dash
  - Machine learning prediction
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive analytics
  - Machine learning predictive results

INTRODUCTION

SpaceX is a company that designs, manufactures and launches advanced rockets and spacecraft, advertising on its website that Falcon 9 rocket launches with a cost of 62 million dollars; when other providers cost upward of 165 million dollars each.

Much of the savings is because SpaceX can reuse the first stage on the next mission. Therefore if we can determine if the first stage will land, we can determine the cost of a launch.

The goal of this project as a data scientist is to predict the landing outcome of the first stage by gathering information about SpaceX and training a machine learning model in order to bid against it for a rocket launch.

METHODOLOGY :Data Collection with an API

Perform a **get request** to obtain the launch data from the **API**.

Results can be viewed by calling the **.json()** method.

Convert JSON to a dataframe by using the **json_normalize function**.

Transform raw data into a **clean dataset** providing meaningful data to be used.

Perform **data cleaning** to deal with missing and null values.

More in:
https://github.com/AndreaDAlcantara/AppliedDSCapstone/blob/e52987c366dff5048cecb2463156679296c6921e/Data%20Collection%20API.ipynb

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

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

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that
have multiple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the featu
re.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

```
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}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

```
# Create a data from launch_dict
launch_df = pd.DataFrame.from_dict(launch_dict)
```

METHODOLOGY :Data Collection with Web Scraping

Python BeautifulSoup package used to web scrape some HTML tables that contain valuable Falcon 9 launch records.

Parse the data from those tables and convert them into a Pandas data frame for further visualization and analysis.

Transform raw data into a **clean dataset** providing meaningful data to be used.

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

```
# use requests.get() method with the provided static_url
# assign the response to a object
data  = requests.get(static_url).text
```

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

```
# 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.

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

```
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.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)
            datatimelist=date_time(row[0])
```

```
df=pd.DataFrame(launch_dict)
df.head(100)
```

More in:
https://github.com/AndreaDAlcantara/AppliedDSCapstone/blob/0a7eeaebae236dfb54886d3d951e8b9903033786/Data%20Collection%20with%20Web%20Scraping.ipynb

METHODOLOGY :Data Wrangling

Find some patterns in the data and determine what would be the label for training supervised models.

```
# landing_outcomes = values on Outcome column
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.

```
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:

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

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

Convert all possible landing outcomes into Training Labels with 1 meaning the booster successfully landed and 0 meaning it was unsuccessful.

```
# 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

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

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

More in:
https://github.com/AndreaDAlcantara/AppliedDSCapstone/blob/0ab8d04adebb99431e779d64289ea433f32a19c7/labs-jupyter-spacex-Data%20wrangling.ipynb

METHODOLOGY :EDA with SQL

SQL queries performed to have a better understanding of the dataset:

- *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 landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order*

More in:
https://github.com/AndreaDAlcantara/AppliedDSCapstone/blob/986aeb62e14933c02463cee070e17fd3f43ca233/EDA%20with%20SQL.ipynb

METHODOLOGY :EDA with Data Visualization

Performed Exploratory Data Analysis and Feature Engineering using Pandas and Matplotlib:

Data Analysis

- Scatter Graphs
  - FlightNumber vs. PayloadMass
  - FlightNumber vs LaunchSite
  - Payload Vs. Launch Site
  - FlightNumber vs Orbit type
  - Payload vs. Orbit
- Bar Chart
  - Success rate by orbit type
- Line Chart
  - Launch success yearly trend

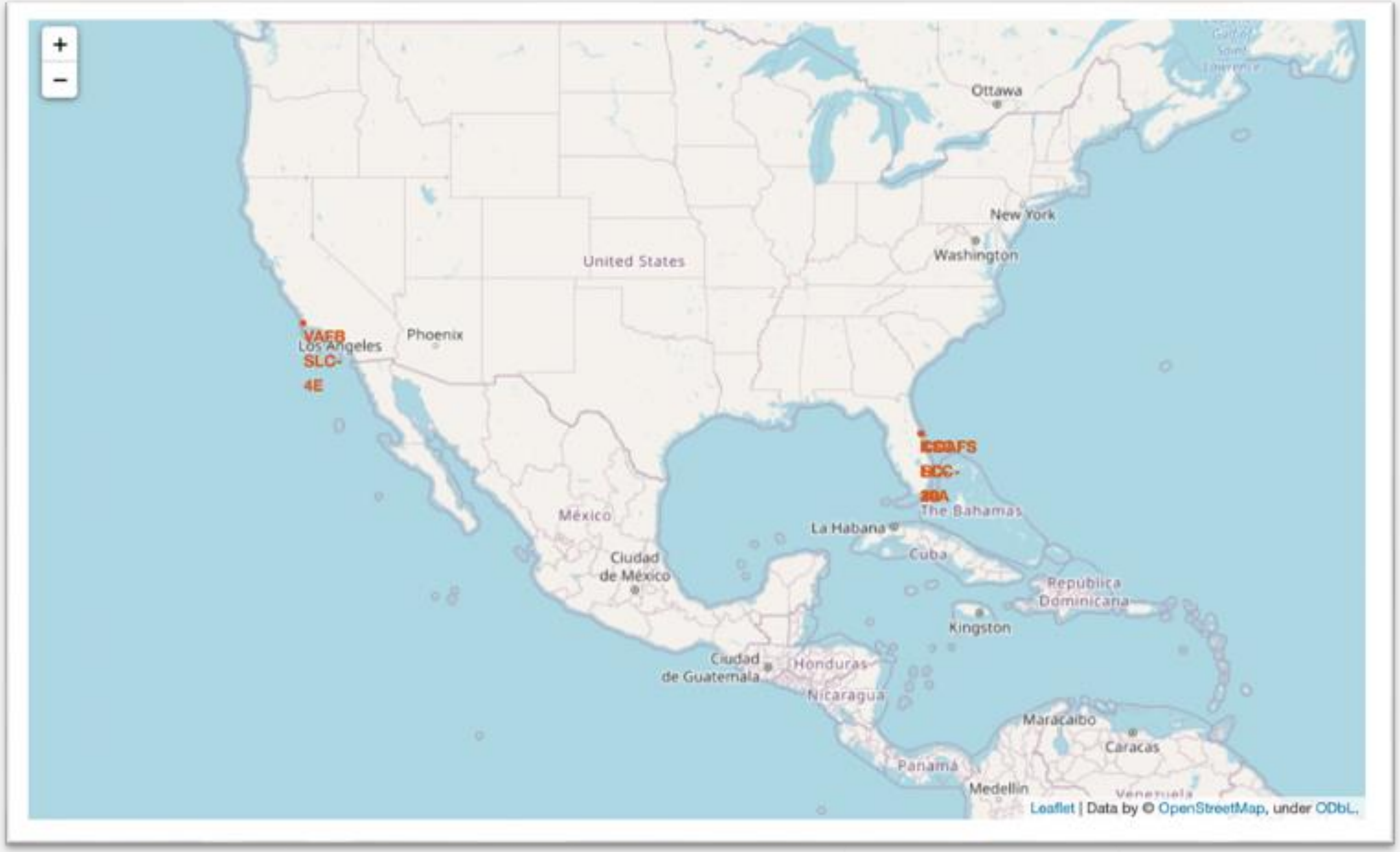
Feature Engineering

- Select the features and create categorical columns that will be used in success prediction

More in:
https://github.com/AndreaDAlcantara/AppliedDSCapstone/blob/42fcffe796237c05bb858bbd4d29bfbbfcd25e16/EDA%20with%20Data%20Visualization.ipynb

METHODOLOGY :Interactive Visual Analytics with Folium

Launch sites and their landing outcomes information were pinned, using the Folium library, to get visual insights about the location and proximities of them.

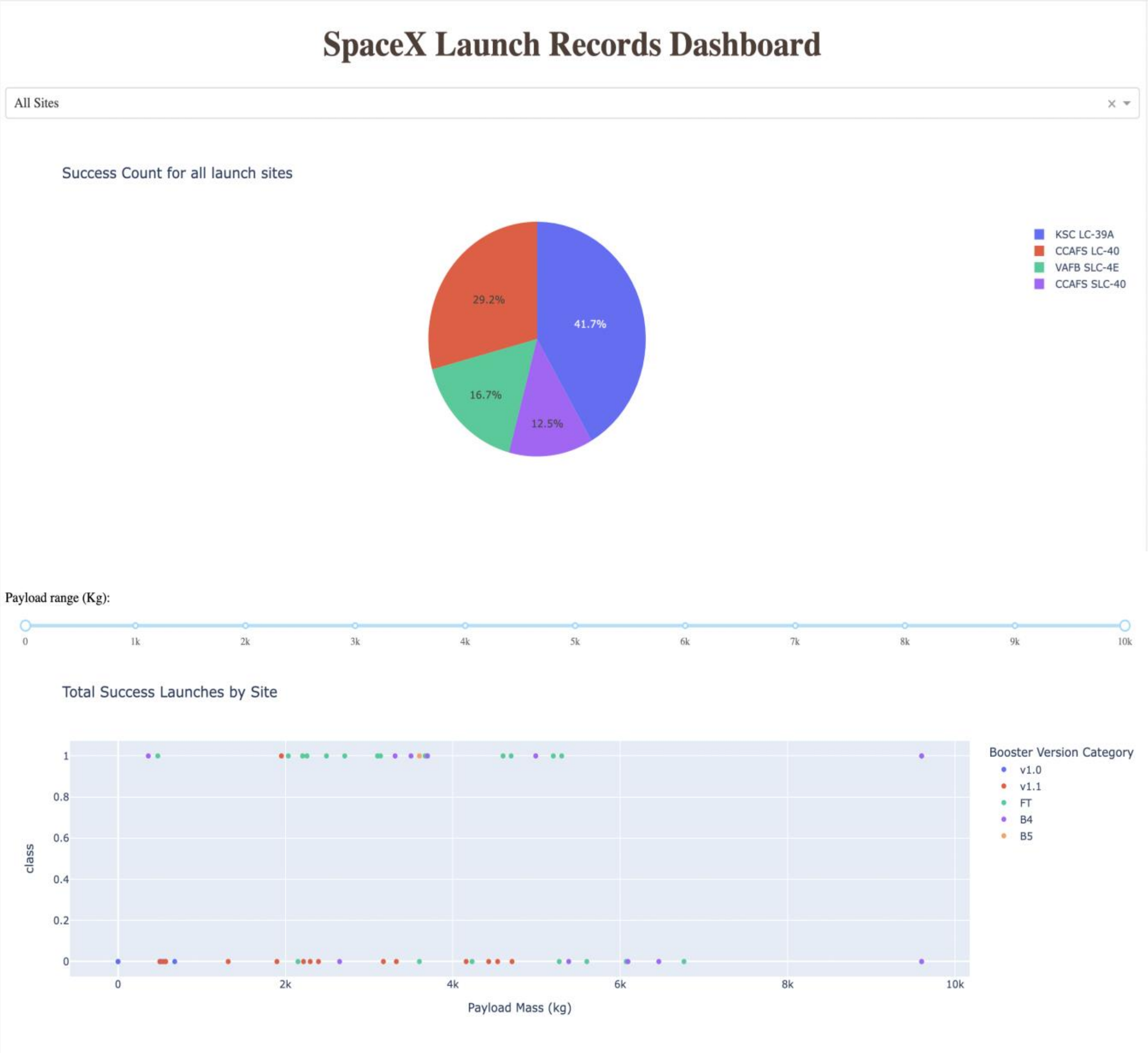


More in:
https://github.com/AndreaDAlcantara/AppliedDSCapstone/blob/42fcffe796237c05bb858bbd4d29bfbbfcd25e16/Interactive%20Visual%20Analytics%20with%20Folium.ipynb

METHODOLOGY :Interactive Dashboard with Plotly Dash

A Plotly Dash application was built to perform interactive visual analytics on SpaceX launch data in real-time.

The dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart to retrieve meaningful information about success launches by site and its correlation with the payload mass.



More in:
https://github.com/AndreaDAlcantara/AppliedDSCapstone/blob/42fcffe796237c05bb858bbd4d29bfbbfcd25e16/spacex_dash_app.py

METHODOLOGY :Machine Learning Prediction

The following Machine Learning methodology was used:

- Standardize the data.
- Split into training and test data.

Perform Exploratory Data Analysis and determine Training Labels

Evaluate and improve the model

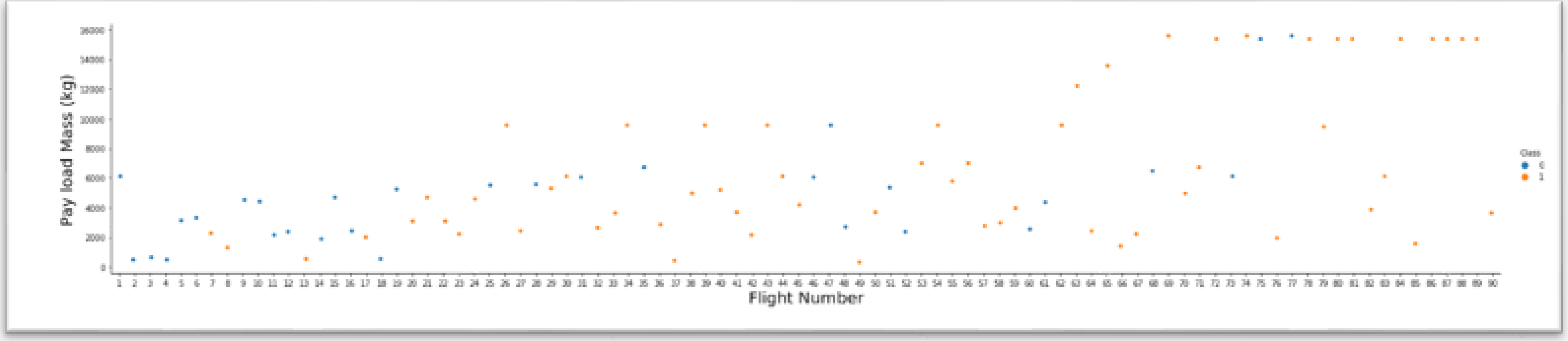
- Train the models and select **hyperparameters using** the function `GridSearchCV`.
- Find best Hyperparameter of: SVM, Classification Tree, KNN and Logistic Regression.

Find best method

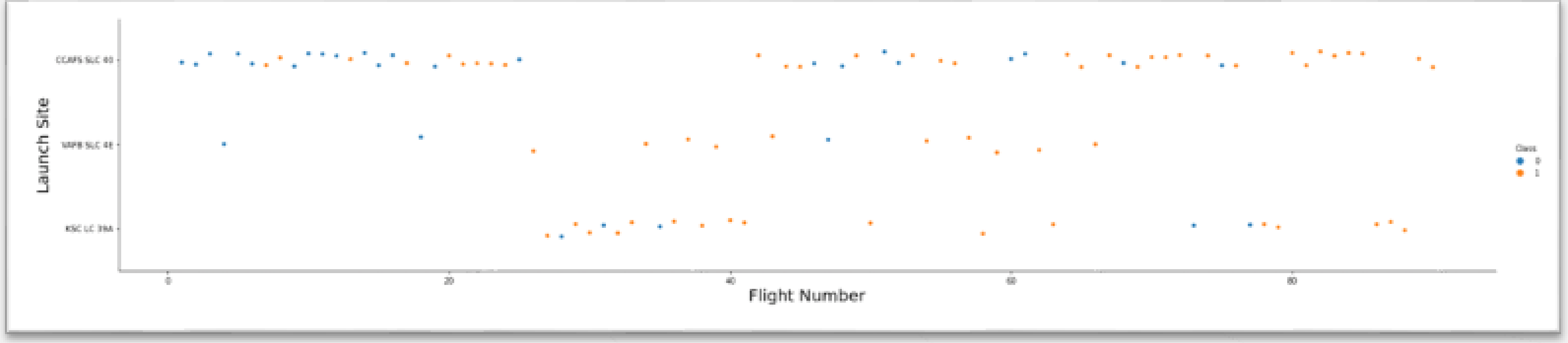
- Compare the accuracy of each method and find which performs best using test data.

More in:
https://github.com/AndreaDAlcantara/AppliedDSCapstone/blob/42fcffe796237c05bb858bbd4d29bfbbfcd25e16/Machine%20Learning%20Prediction.ipynb

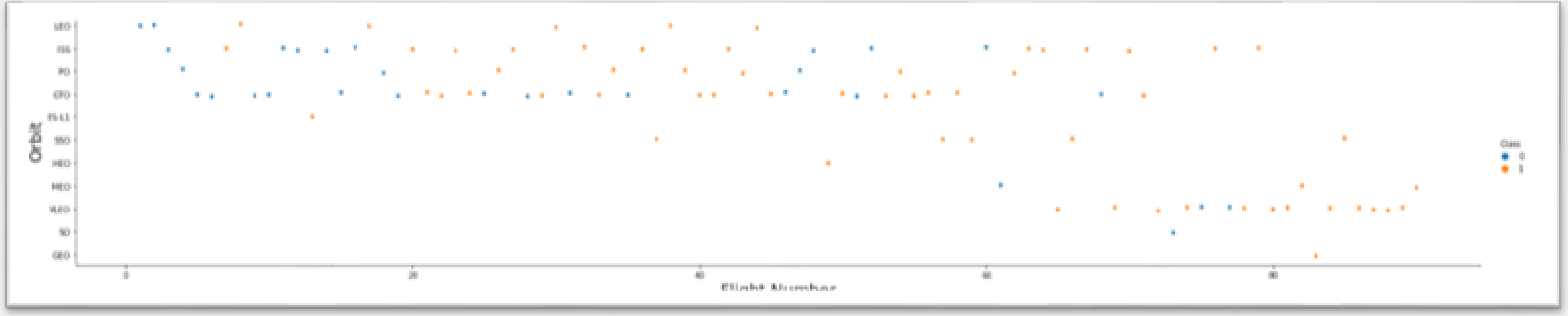
RESULTS :EDA with Data Visualization



We see that different launch sites have different success rates. **CCAFS LC-40**, has a success rate of **60%**, while **KSC LC-39A and VAFB SLC 4E** has a success rate of **77%**.

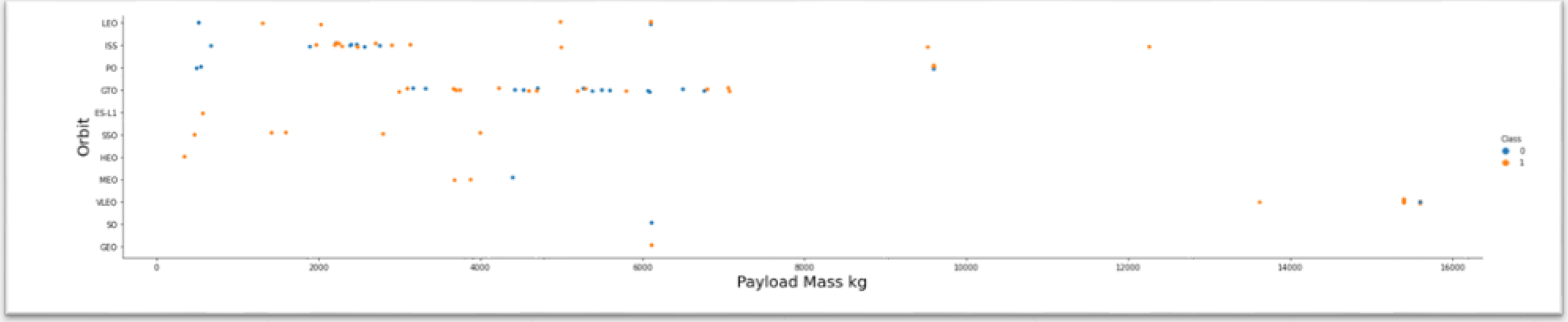


This graph shows that the success rates tend to increase with the number of flights of a site.

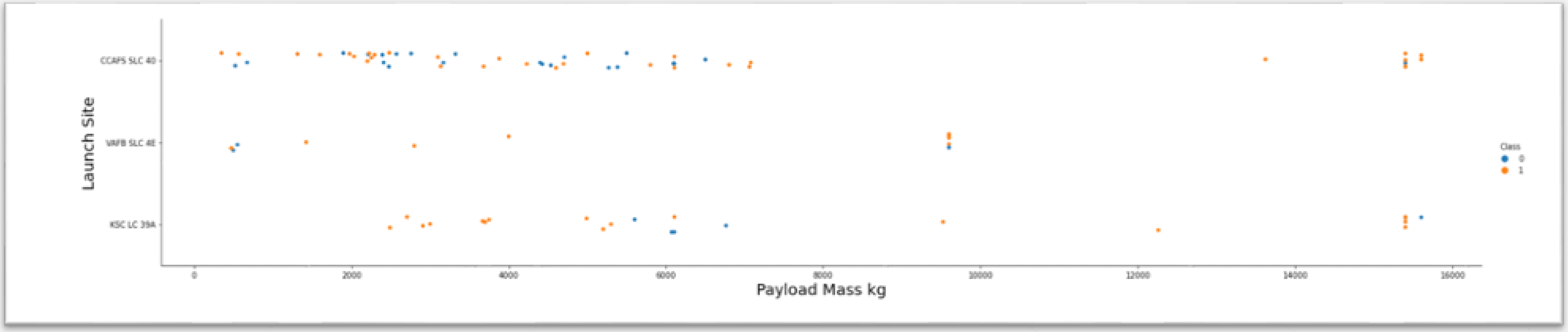


Here we can see that in the **LEO** orbit Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in **GTO** orbit.

RESULTS :EDA with Data Visualization

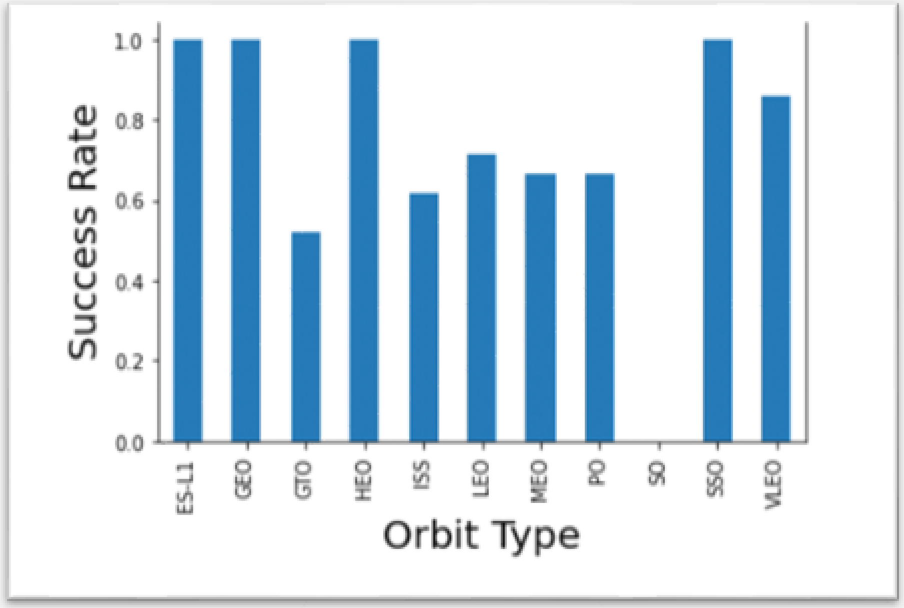


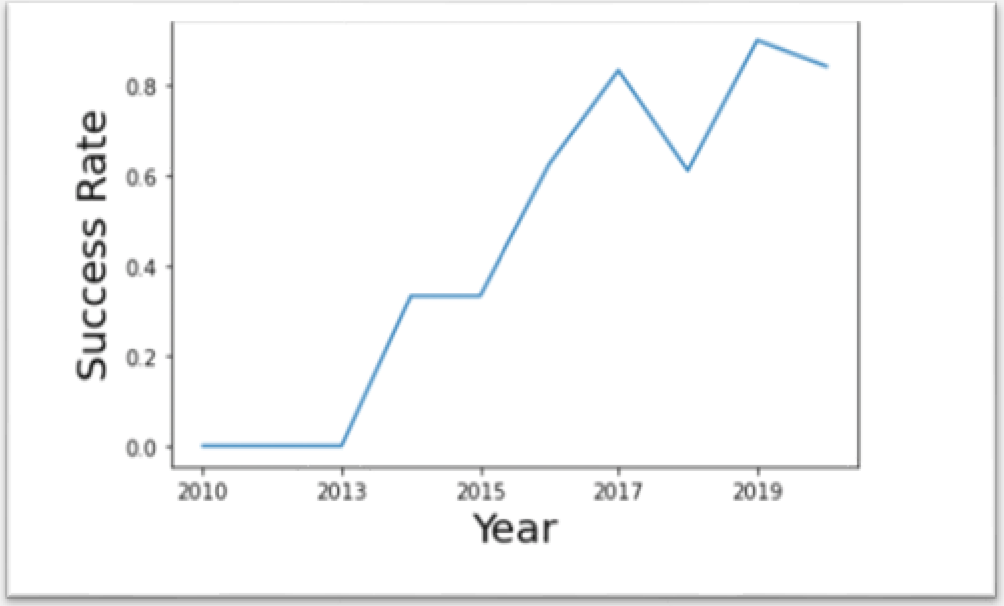
With heavy payloads the successful landing or positive landing rate are more for **Polar, LEO and ISS**. However for GTO it was not possible to distinguish this well as both positive landing rate and negative landing (unsuccessful mission) are both there here.



You will find for the **VAFB-SLC** launch site there are no rockets launched for heavy payload mass (greater than 10000).

RESULTS :EDA with Data Visualization





This bar chart shows that **ES-L1, GEO, HEO and SSO** have the best success rates.

The line chart shows that the sucess rate keeps increasing since **2013 till 2020**.

RESULTS :EDA with SQL

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

```
%sql select distinct (LAUNCH_SITE) from SPACEXTBL group by LAUNCH_SITE;
```

```
 * ibm_db_sa://wlc44326:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

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

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

```
%sql select * from SPACEXTBL where launch_site like 'CCA%' limit 5;
```

```
 * ibm_db_sa://wlc44326:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

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

Hint:Use min function

```
%sql SELECT * FROM SPACEXTBL WHERE DATE = (SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (ground pad)');
```

```
 * ibm_db_sa://wlc44326:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2015-12-22	01:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_MASS FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)' GROUP BY CUSTOMER;
```

```
 * ibm_db_sa://wlc44326:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

total_mass
45596

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

```
%sql SELECT * FROM (SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION LIKE '%F9 v1.1%');
```

```
 * ibm_db_sa://wlc44326:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

1
2534

RESULTS :EDA with SQL

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, PAYLOAD_MASS__KG_, LANDING__OUTCOME FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (drone ship)' AND PAYLO
```

* ibm_db_sa://wlc44326:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.

booster_version	payload_mass__kg_	landing__outcome
F9 FT B1022	4696	Success (drone ship)
F9 FT B1026	4600	Success (drone ship)
F9 FT B1021.2	5300	Success (drone ship)
F9 FT B1031.2	5200	Success (drone ship)

List the total number of successful and failure mission outcomes

```
%sql SELECT MISSION_OUTCOME, COUNT(*) FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE '%Success%' OR MISSION_OUTCOME LIKE '%Failure%' GROUP BY M
```

* ibm_db_sa://wlc44326:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.

mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

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

```
%sql SELECT DATE, BOOSTER_VERSION, LAUNCH_SITE, LANDING__OUTCOME FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND YEAR(D
```

* ibm_db_sa://wlc44326:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.

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)

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

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

* ibm_db_sa://wlc44326:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
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

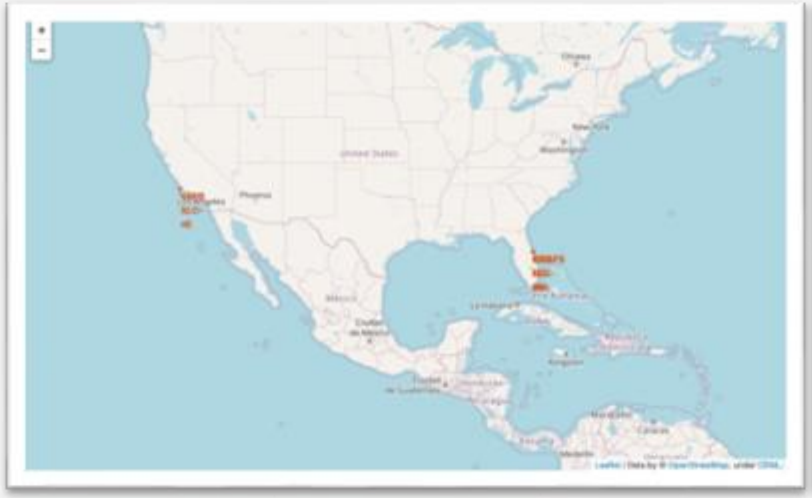
Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
%sql SELECT COUNT(*) AS RANK, LANDING__OUTCOME FROM SPACEXTBL WHERE DATE>'2010-06-04' AND DATE<'2017-03-20' GROUP BY LANDING__OUTCOME ORDE
```

* ibm_db_sa://wlc44326:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.

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

RESULTS :Interactive Visual Analytics with Folium



Map with marked launch sites

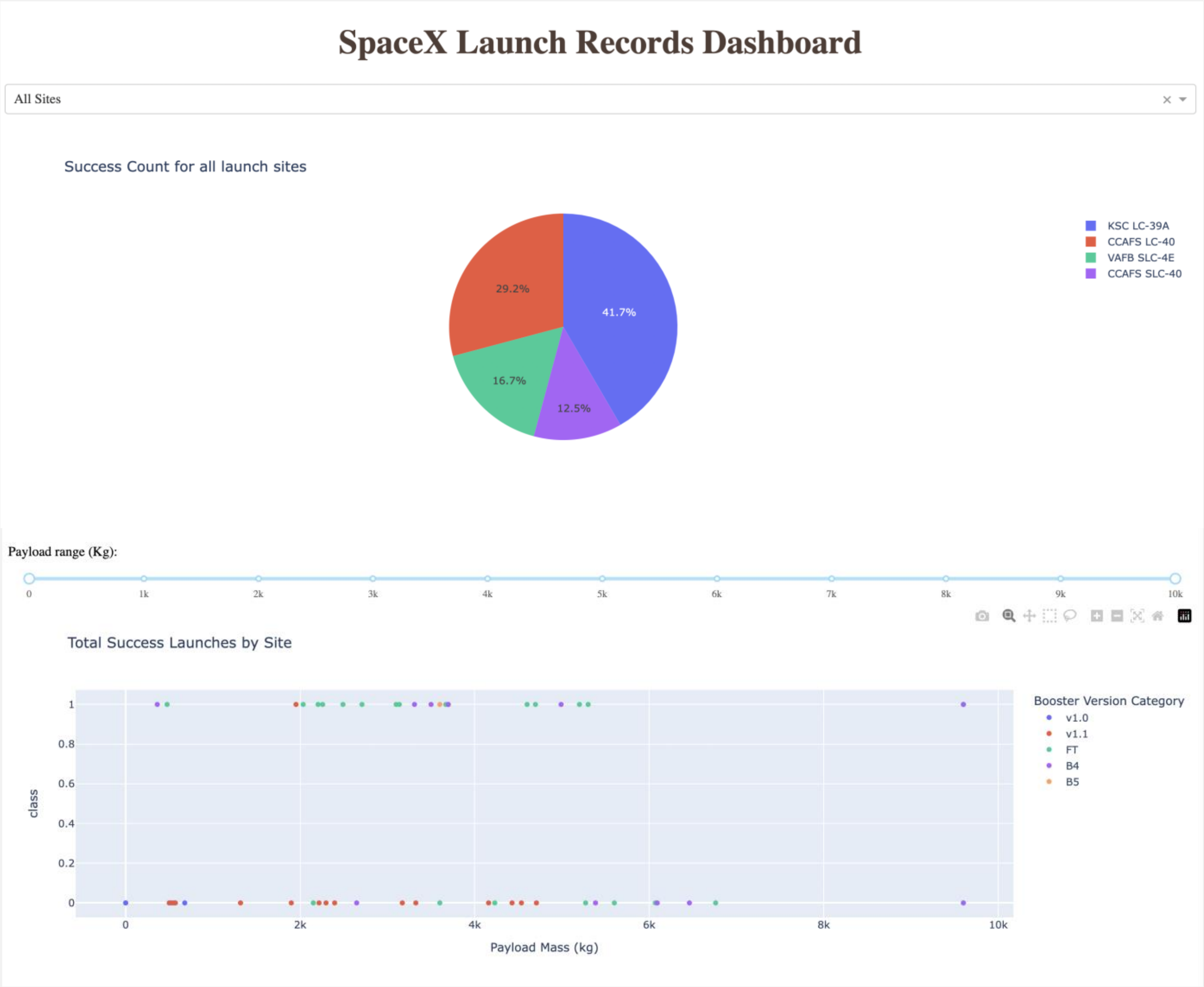
Map with color-labeled markers to easily identify which launch sites have relatively high success rates.

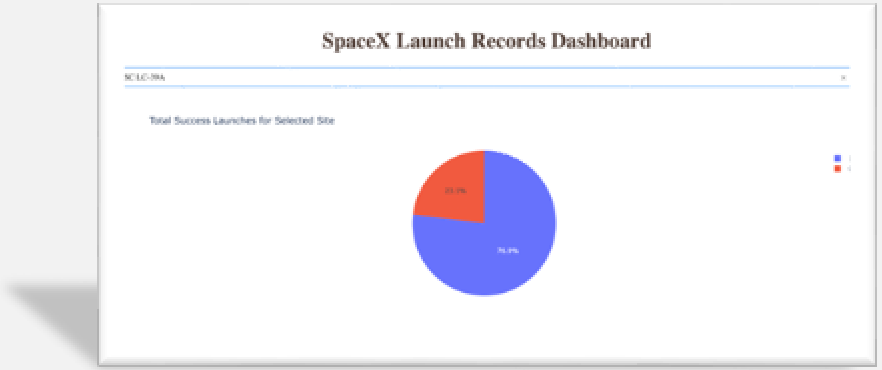
Map with plot distance lines to the proximities.

After this interactive visual analysis it was possible to answer the questions, as below:

- Are launch sites in close proximity to railways? No.
- Are launch sites in close proximity to highways? No.
- Are launch sites in close proximity to coastline? Yes.
- Do launch sites keep certain distance away from cities? Yes.

RESULTS :Interactive Visual Analytics with Plotly Dash





Some insights after visual analysis using the dashboard:

- **KSC LC-39A** has the **largest** successful launches AND **highest** launch success rate.
- **2-4k** payload range has the **highest** launch success rate.
- **6-10k** payload range has the **lowest** launch success rate.
- **FT F9 Booster version** has the **highest** launch success rate.

RESULTS :Machine Learning Prediction

```
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

```
Train set: (72, 83) (72,)
Test set: (18, 83) (18,)
```

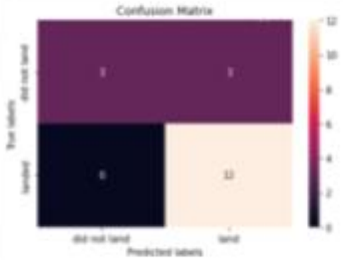
Calculate the accuracy on the test data using the method `score` :

```
y_pred = logreg_cv.predict(X_test)
r2 = logreg_cv.score(X_test, Y_test)
print('Accuracy on test: {}'.format(r2))
```

```
Accuracy on test: 0.8333333333333334
```

Lets look at the confusion matrix:

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



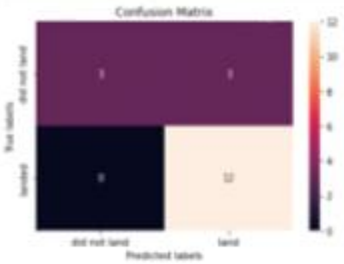
Calculate the accuracy on the test data using the method `score` :

```
acc2 = svm_cv.score(X_test, Y_test)
print('Accuracy on test: {}'.format(acc2))
```

```
Accuracy on test: 0.8333333333333334
```

We can plot the confusion matrix

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

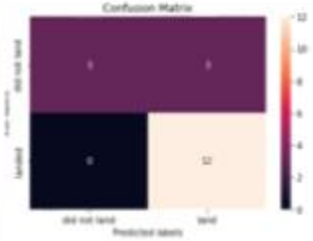



```
y_pred = tree_cv.predict(X_test)
acc3 = tree_cv.score(X_test, Y_test)
print('Accuracy on test: {}'.format(acc3))
```

```
ccuracy on test: 0.8333333333333334
```

We can plot the confusion matrix

```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



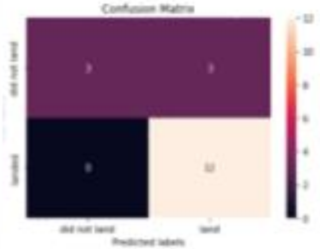
alculate the accuracy of tree_cv on the test data using the method `score` :

```
acc4 = knn_cv.score(X_test, Y_test)
print('Accuracy on test: {}'.format(acc4))
```

```
ccuracy on test: 0.8333333333333334
```

We can plot the confusion matrix

```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Find the method performs best:

```
print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
print( 'Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
print('Accuracy for K nearsdt neighbors method:', knn_cv.score(X_test, Y_test))
```

```
Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.8333333333333334
Accuracy for K nearsdt neighbors method: 0.8333333333333334
```

After calculating the accuracy and plotting the confusion matrix for each one of the following methods Logistic Regression, Decision Tree, KNN and SVM, all of them got the same score as shown in the pictures.

CONCLUSION

- Success landing rates tend to increase with the number of flights of a site.
- The launch sites KSC LC-39A and VAFB SLC 4E have better success rates (77%) than CCAFS LC-40 (60%).
- The orbits ES-L1, GEO, HEO and SSO have the best success rates.
- Heavy payloads tend not to have positive landing rate except for the orbits Polar, LEO and ISS.
- Sucess rate keeps increasing since 2013 till 2020.


Thank You!