Winning Space Race with Data Science

Carl Che
March 7, 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Collecting the data through API and Web Scraping
  - Data Wrangling
  - Exploratory data analysis
  - Data visualization
  - Model development
  - Model evaluation
- Summary of all results

Introduction

- Project background and context

we will predict if the Falcon 9 first stage will land successfully. SpaceX 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 if the first stage will land, we can determine the cost of a launch.

Problems you want to find answers

- Factors that determine the rocket launch success rate

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - We use standard Python API request package to make HTTP request to spaceX restful endpoint to query falcon launch data in JSON format and load it into python panda data frame for further analysis
  - Use Web Scraping to load launch data into BeautifulSoup object, and parse the HTML page to create a panda data frame
- Perform data wrangling
  - For numerical data if missing, we replace it with the mean value of the column.
- 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

- Request and parse the SpaceX launch data using GET request
  - Use panda read_json query spaceX launch data and store in the panda data frame
  - Manipulate the data fram, keep the the interested column
  - Filter the data, keep date before Nov 13, 2020, core and playload only with one element
  - Use the helper function to query the launch data details with the identification number in the launch data
  - Use the result to form a python dictionary and then create panda data from to hold the spaceX launch data details
- Web Scraping Wikipedia HTML table for the SpaceX launch data with Beautifulsoup
  - Get HTML page content with python request API and create BeautifulSoup object
  - Find the tables in the page with soup object, extract the column name into a list for the python dictionary object
  - Parse the html table and fill in the date into a python dictionary
  - Create a panda data frame from the dictionary

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts
  - Use panda read_json query spaceX launch history
  - Do the data cleaning
  - Use python request API to query launch details per the identification of the launch data
- Add the GitHub URL of the completed SpaceX API calls notebook:
  https://github.com/CarlChe/datascience/blob/master/Data%20Collection%20API.ipynb

To make the requested JSON results more consistent, we will use the following static response object for this project:

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

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

```
response.status_code
```

```
200
```

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

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

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

```
# 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 payloa
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 feature.
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)]
```

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

the list has now been update

```
BoosterVersion[0:5]
```

```
['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

we can apply the rest of the functions here:

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

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

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

Data Collection - Scraping

- Present your web scraping process:
  - Extract a Falcon 9 launch records HTML table from Wikipedia
  - Parse the table and convert it into a Pandas data frame
- Add the GitHub URL of the completed web scraping notebook:
  https://github.com/CarlChe/datascience/blo/master/Data%20Collection%20with%20Web%20Scraping.ipynb

```
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])

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

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

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

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

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

            # Payload Mass
            # TODO: Append the payload_mass into launch_dict with key `Payload mass`
            payload_mass = get_mass(row[4])
            launch_dict['Payload mass'].append(payload_mass)
            #print(payload)

            # Orbit
            # TODO: Append the orbit into launch_dict with key `Orbit`
            orbit = row[5].a.string
            launch_dict['Orbit'].append(orbit)
            #print(orbit)

            # Customer
            # TODO: Append the customer into launch_dict with key `Customer`
            customer = row[6].text.strip()
            launch_dict['Customer'].append(customer)
            #print(customer)

            # Launch outcome
            # TODO: Append the launch_outcome into launch_dict with key `Launch outcome`
            launch_outcome = list(row[7].strings)[0]
            launch_dict['Launch outcome'].append(launch_outcome)
            #print(launch_outcome)

            # Booster landing
            # TODO: Append the launch_outcome into launch_dict with key `Booster landing`
            booster_landing = landing_status(row[8])
            launch_dict['Booster landing'].append(booster_landing)
            #print(booster_landing)
```

```
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']=[]
```

```
# 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)
```

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

falcon_launch = requests.get(static_url)
```

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a respons
soup = BeautifulSoup(falcon_launch.content, "html.parser")
```

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

```
# Use soup.title attribute
soup.title
```

Data Wrangling

- Identify the missing values in the dataset
- Fill the missing number value with mean value of the column
- Exploratory data analysis
  - Check column data type
  - Calculate the numbers of launches on each sites.
  - Calculate the number and occurrence of each orbit
  - Calculate the number and occurrence of mission outcome per orbit type
- Create landing outcome label in the data frame.
  - Create landing outcome label from data frame “Outcome” column
  - Add new column in the data frame to mark the launch status per outcome label

- GitHub notebook link
  https://github.com/CarlChe/datascience/blob/master/Data%20wrangling.ipynb

EDA with Data Visualization

- We explore the different factors related to the launch outcome

  Categorical plot for factors impact on the launch outcome: FlightNumber and Payload, LaunchStie and FlightNumber, LaunchStie and PayloadMass, Oribt and FlightNumber, Orbit and PayloadMass

  Barchar plot for factor impact on launch outcome: Orbit

  Regression plot for yearly launch success rate.

- Add the GitHub URL of your completed EDA with data visualization notebook: https://github.com/CarlChe/datascience/blob/master/EDA%20with%20Data%20Visualization.ipynb

EDA with SQL

- Query unique launch sites from space mission
- Query 5 records where launch site begin with string “CCA”
- Query total payload mass carried by boosters launched by “NASA (CRS)”
- Query average payload mass carried by booster version “F9 v1.1”
- Query data when the first successful landing outcome is ground pad was achived
- Query names of the booster which have success in drone ship and have payload with 4000 and 6000 range
- Query total number of successful and failure mission outcomes
- Query name of the booster_version which have carried the maximum payload mass
- Query bootster_version and launch site name with the failed landing_outcomes in drone ship in year 2015
- Query landing outcome between 2010-06-04 and 2017-03-20 in descending order

- Add the GitHub URL of your completed EDA with SQL notebook:
  https://github.com/CarlChe/datascience/blob/master/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- Use folium.Circle and folium.Marker to lable each launch site, mark the success/failed launches for each site
- Explain why you added those objects.

  Mark the launch site and its name on the map, so it is visual label on the map

  To have direct visual label for each launch outcome at each launch site.

  To label the distance that the launch site from the neareast costline, city, and highway.
- Add the GitHub URL of your completed interactive map with Folium map: https://github.com/CarlChe/datascience/blob/master/Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

- I add HTML component in dashboard: a dropdown list, a Pie chart, a range slider, a pie chart and scatter plot callback function
- Dropdown list valued is used as Pie chart input, and the slider value range and dropdown list value is used to control data used to render the scatter chart

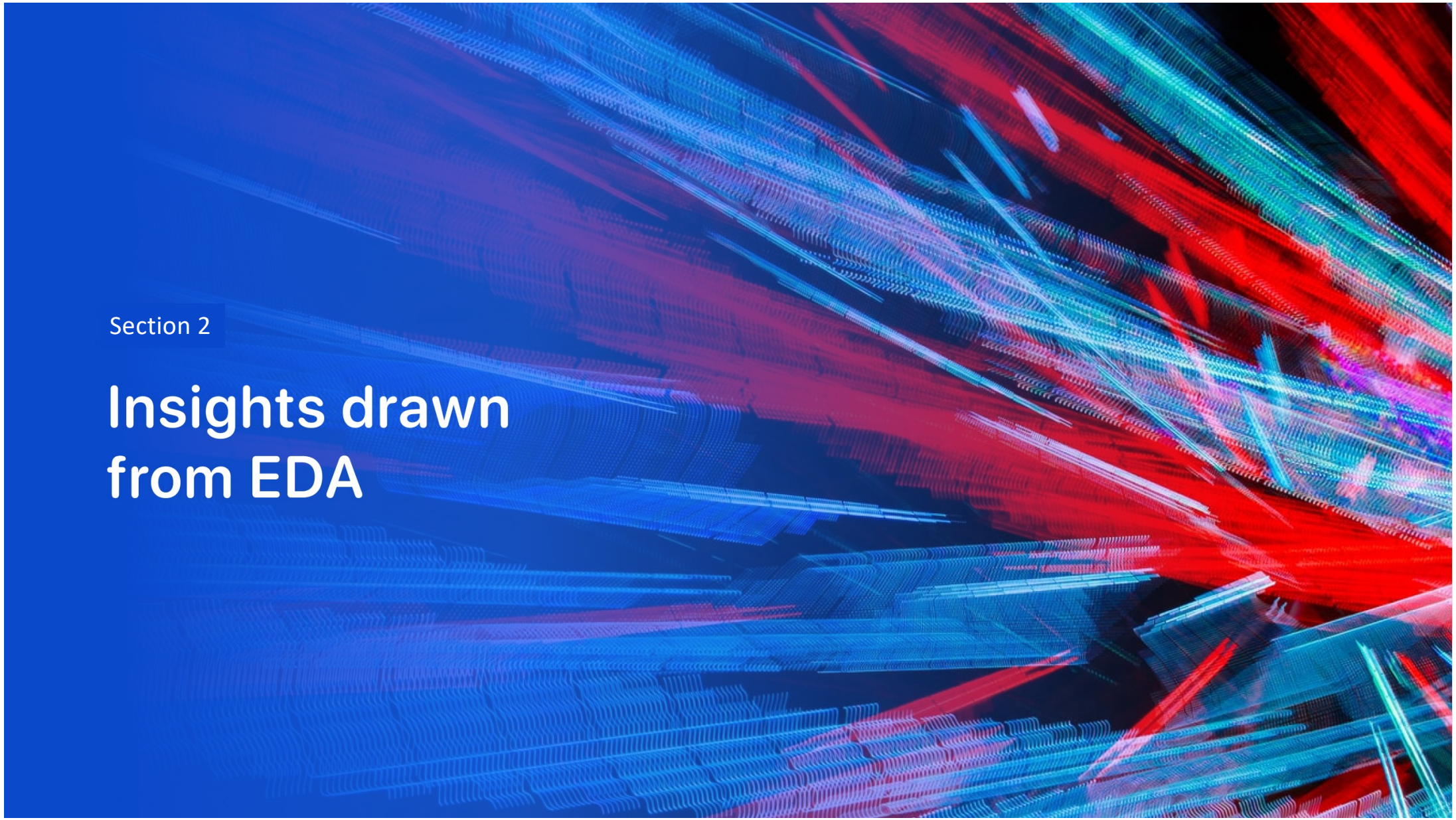
- Add the GitHub URL of your completed Plotly Dash lab: https://github.com/CarlChe/datascience/blob/master/sapcex_dash_app.py

Predictive Analysis (Classification)

- Classification model
    1. Have data frame, dummy variable for column category (X), and class array for the landing outcomes (Y)
    2. Standardize the dummy variable for column category data
    3. Use the function train_test_split to split the data X and Y into training and test data
    4. Create logistic object and fit the training data in and get the accuracy score
    5. Get the score with the training model on the test data with model in step 4, and plot the confusion matrix
    6. Create a support vector machine object and fit training data with best parameter
    7. Get the score with the training model on the test data with model in step 5, and plot the confusion matrix
    8. Create decision tree classifier object and fit training data with best parameter
    9. Get the score with the training model on the test data with model in step 8, and plot the confusion matrix
    10. Create k nearest neighbors object and fit training data with best parameter
    11. Get the score with the training model on the test data with model in step 10, and plot the confusion matrix
    12. Print the algorithm that perform the best with highest core
- You need present your model development process using key phrases and flowchart
- Add the GitHub URL of your completed predictive analysis lab: http://localhost:8888/notebooks/BScUoL/2022-2/CM3005-DS/IBM%20Data%20Science%20Professional%20Certificate/10.%20Applied%20Data%20Science%20Capstone/datascience/Machine%20Learning%20Prediction.ipynb

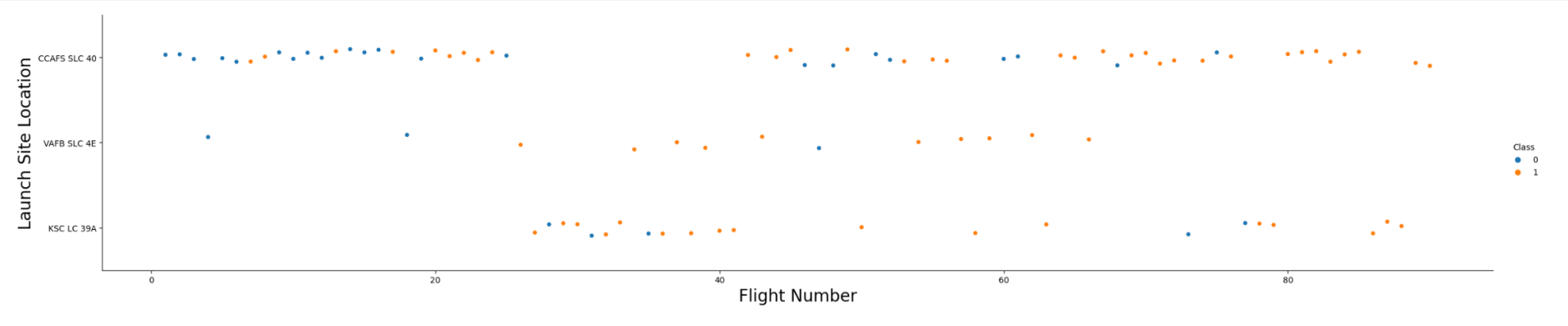
Results

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


Section 2
Insights drawn from EDA

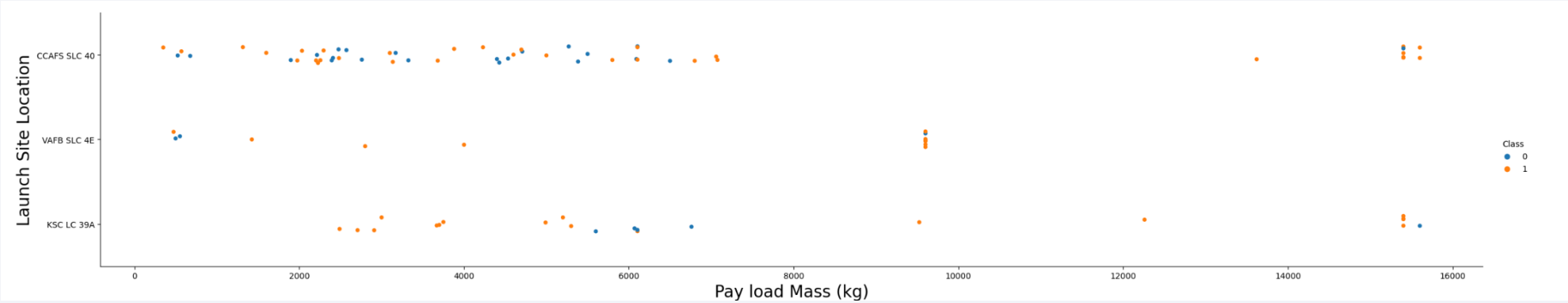
Flight Number vs. Launch Site

- As the Flight Number increase, the successful landing rate increase, CCAFS SLC 40 lauch site are take most the launch mission and it stopped used for a while and move to KSC LC 39A site, VAFB SLC 4E not in use at later stage of launch mission



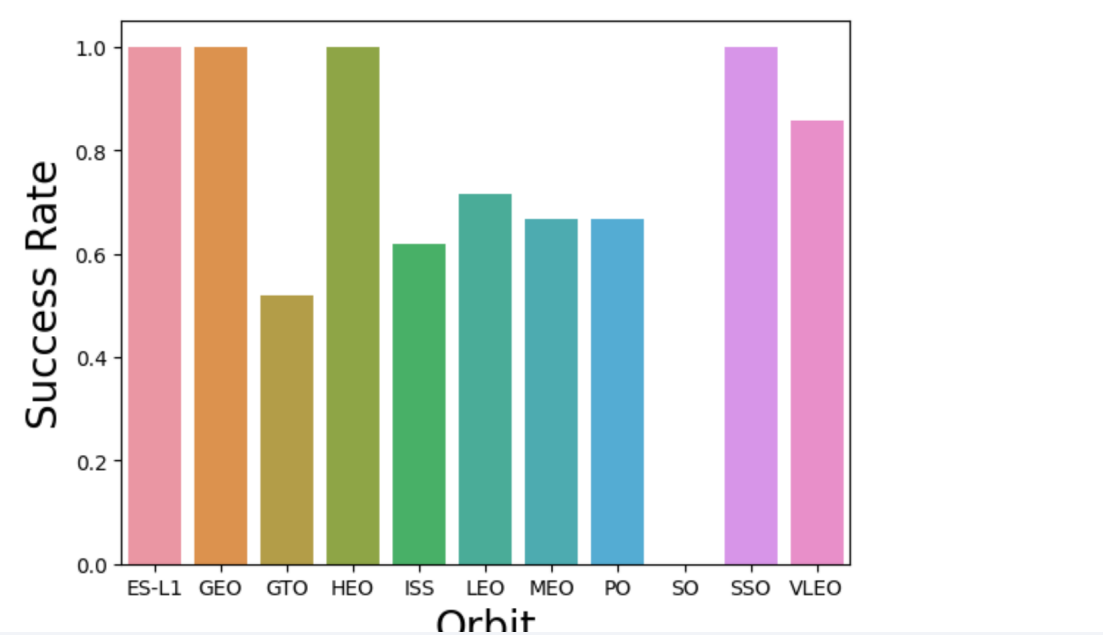
Payload vs. Launch Site

- VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).



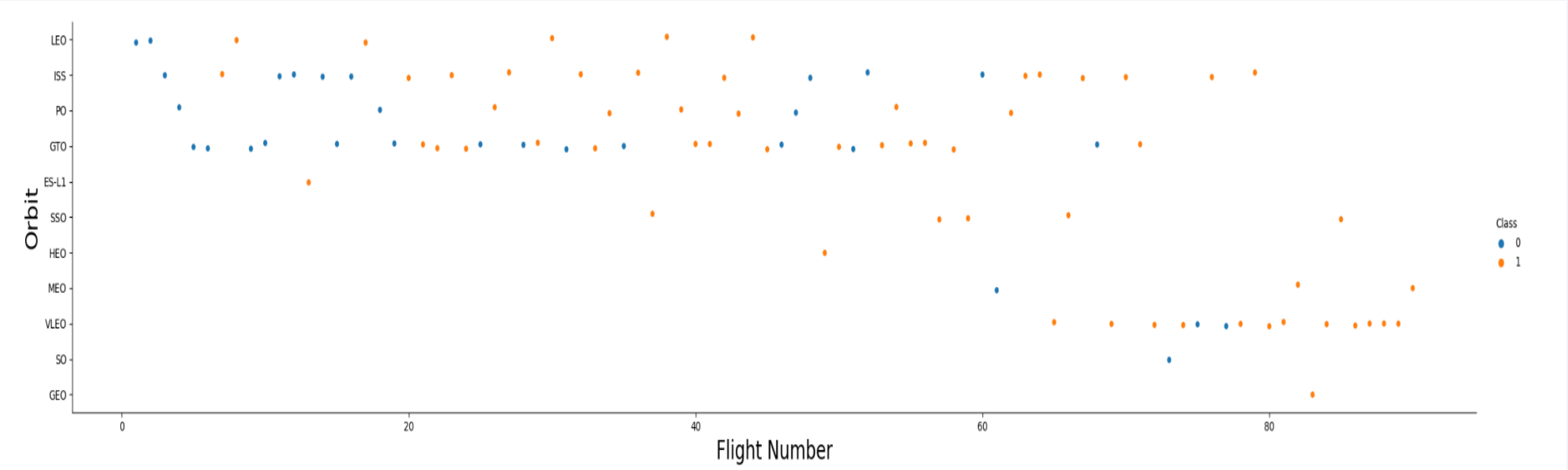
Success Rate vs. Orbit Type

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



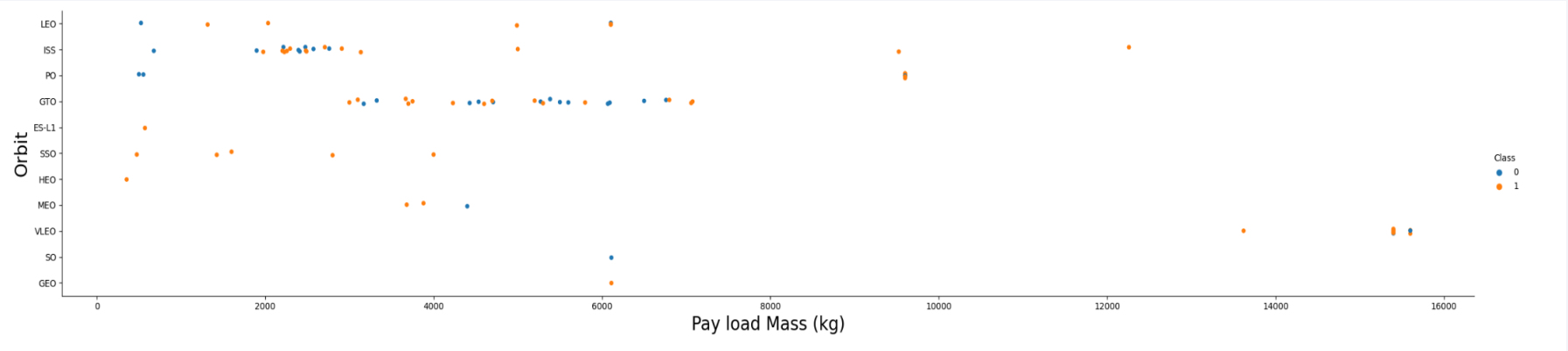
Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type



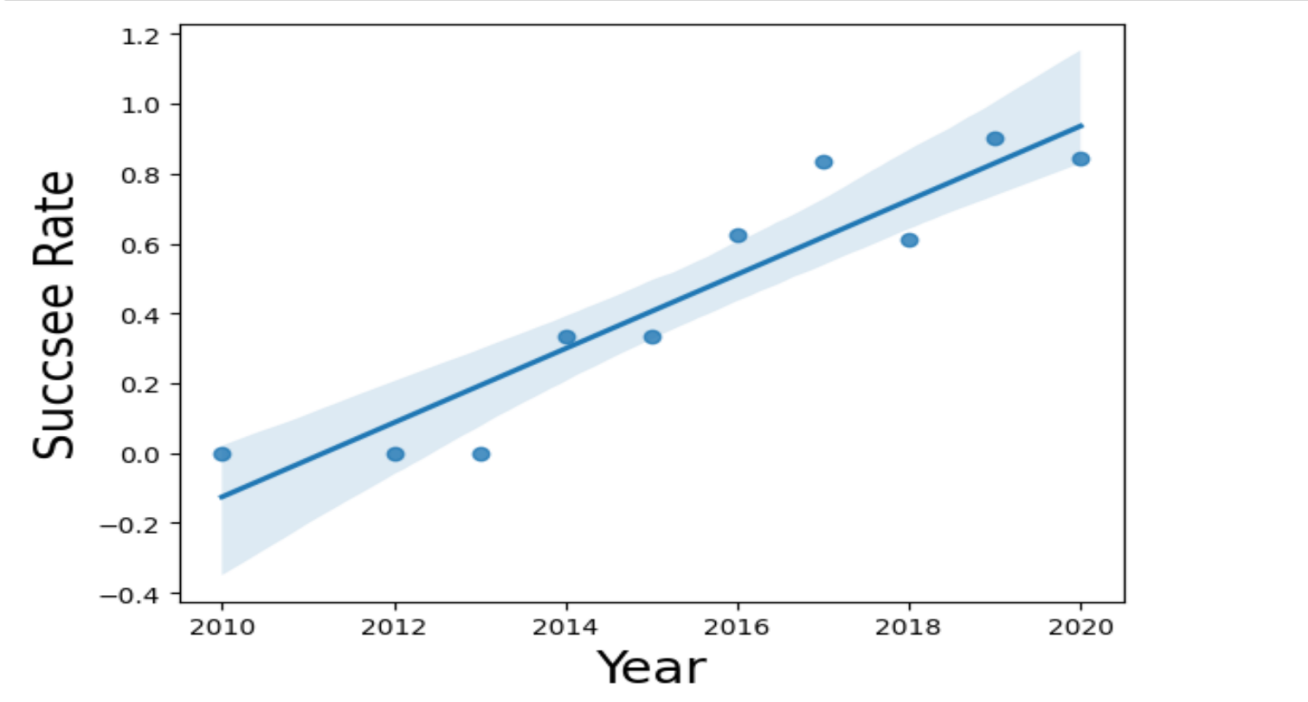
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there.
- ES-L1, SSO, HEO launch all success when payload less then 4000



Launch Success Yearly Trend

- Sucess rate since 2013 kept increasing till 2020



All Launch Site Names

- Use the SQL disctinct key word to find the unique value in the table column

```
1 %sql select distinct Launch_Site from SPACEXTBL
2
```

```
 * ibm_db_sa://vbs79701:***@54a2f15b-5c0f-46df-8954-7e38
3/bludb
Done.
```

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

Launch Site Names Begin with 'CCA'

- Query the string column with condition like with pattern
- Limit the raw number with SQL limit key word

```
1 %sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

```
 * ibm_db_sa://vbs79701:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain
3/bludb
Done.
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	la
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	F
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	F
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	

Total Payload Mass

- Use SQL sum function on a number column
- Limit the participate row with condition on the customer column

```
1 %sql select sum(payload_mass__kg_) from SPACEXTBL WHERE customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://vbs79701:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.d
3/bludb
Done.
```

1
45596

Average Payload Mass by F9 v1.1

- Use SQL avg function on a number column
- Limit the participate row with condition on the booster_version column

```
1 %sql select avg(payload_mass__kg_) from SPACEXTBL WHERE booster_version = 'F9 v1.1'
```

```
 * ibm_db_sa://vbs79701:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.dat
3/bludb
Done.
```

1
2928

First Successful Ground Landing Date

- Use SQL min function on a date column
- Limit the participate row with condition on the landing_outcome column

```
1 %sql select min(DATE) from SPACEXTBL WHERE landing__outcome = 'Success (ground pad)'
```

```
 * ibm_db_sa://vbs79701:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.dat
3/bludb
Done.
```

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Limit the participate row with two condition condition connect with ”and” on the landing_outcome column, limit the number column value with “between…and…”

```
%sql select booster_version from SPACEXTBL where landing__outcome = 'Success (drone ship)'\
and payload_mass__kg_ between 4000 and 6000
```

```
 * ibm_db_sa://vbs79701:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases
3/bludb
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

- Aggregation data with "GROUP BY" SQL key word

```
1 %sql select mission_outcome, count(mission_outcome) from SPACEXTBL GROUP BY mission_outcome
```

```
 * ibm_db_sa://vbs79701:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.
3/bludb
Done.
```

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

Boosters Carried Maximum Payload

- Get the max payload from all the launch mission.
- Query the booster_version that payload mass equal the max playload from above

```
%sql select booster_version, payload_mass__kg_ \
from SPACEXTBL \
where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTBL)
```

```
 * ibm_db_sa://vbs79701:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0
3/bludb
Done.
```

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600

expand output; double click to hide output

2015 Launch Records

- Limit the result set with condition on string column and date column

```
%sql select booster_version, launch_site \
from SPACEXTBL \
where landing__outcome = 'Failure (drone ship)' and year(DATE) = 2015
```

```
* ibm_db_sa://vbs79701:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tg
/bludb
one.
```

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

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

- Limit data result set by apply condition on date column
- Aggregate the data with “GROUP BY”
- Sort query result with “ORDER BY”

```
%sql select count(landing__outcome), landing__outcome \
from SPACEXTBL \
where DATE between '2010-06-04' and '2017-03-20' \
group by landing__outcome\
order by count(landing__outcome) desc
```

```
* ibm_db_sa://vbs79701:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c
/bludb
one.
```

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

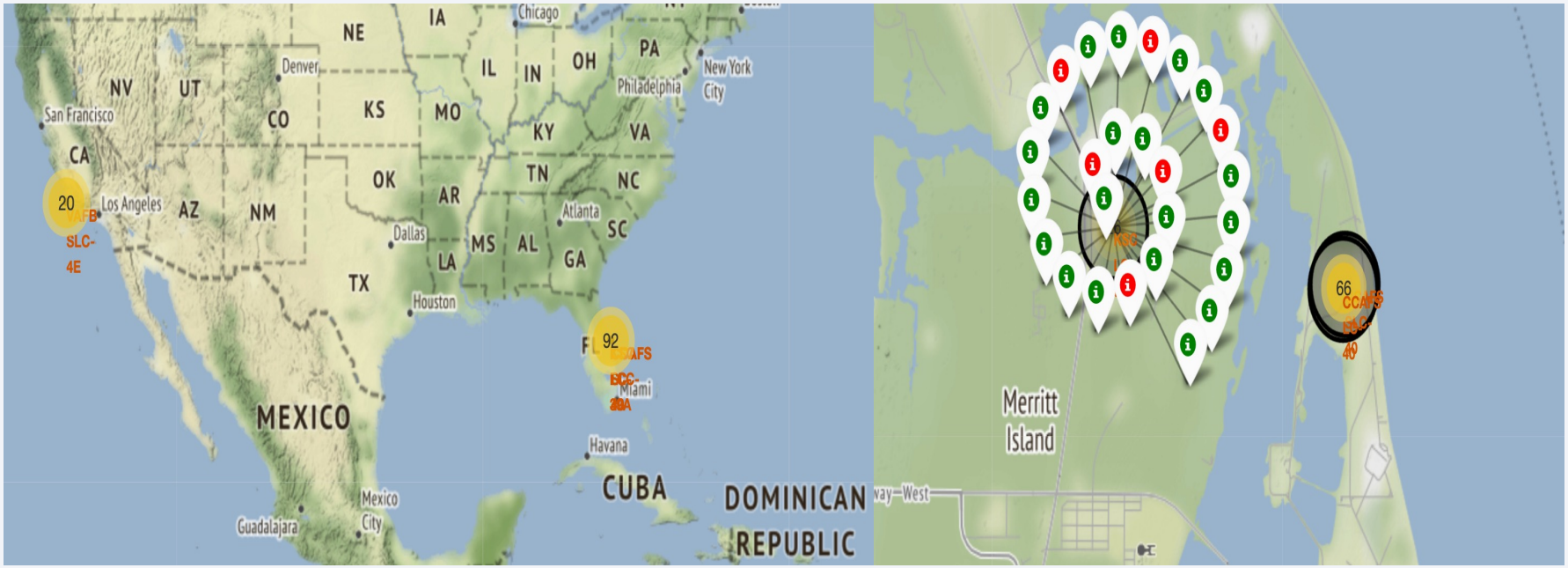

Section 3
Launch Sites
Proximities Analysis

SpaceX global launch sites

- Two launch sites are located in US territory, east and west cosat each.

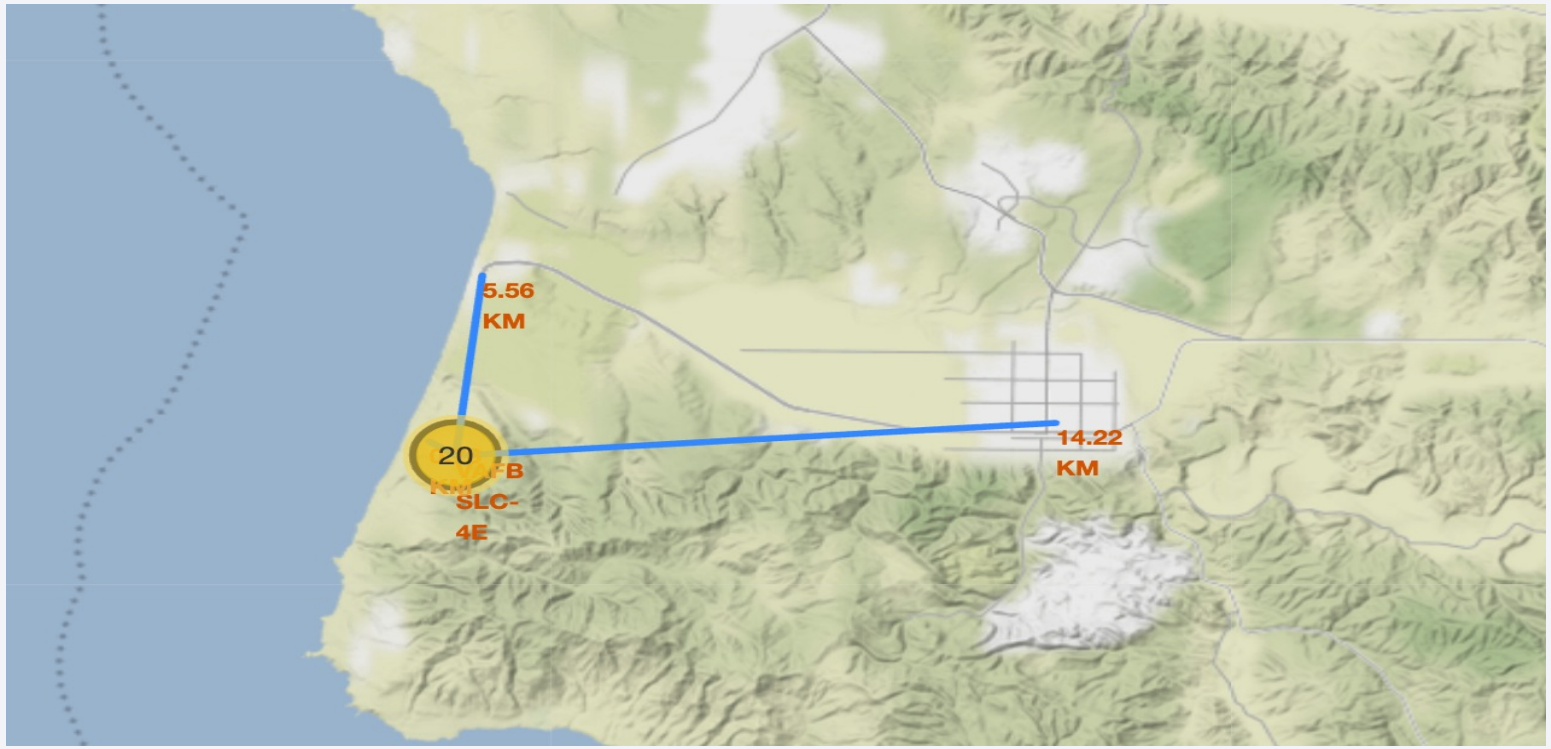
Geo launch mission outcomes

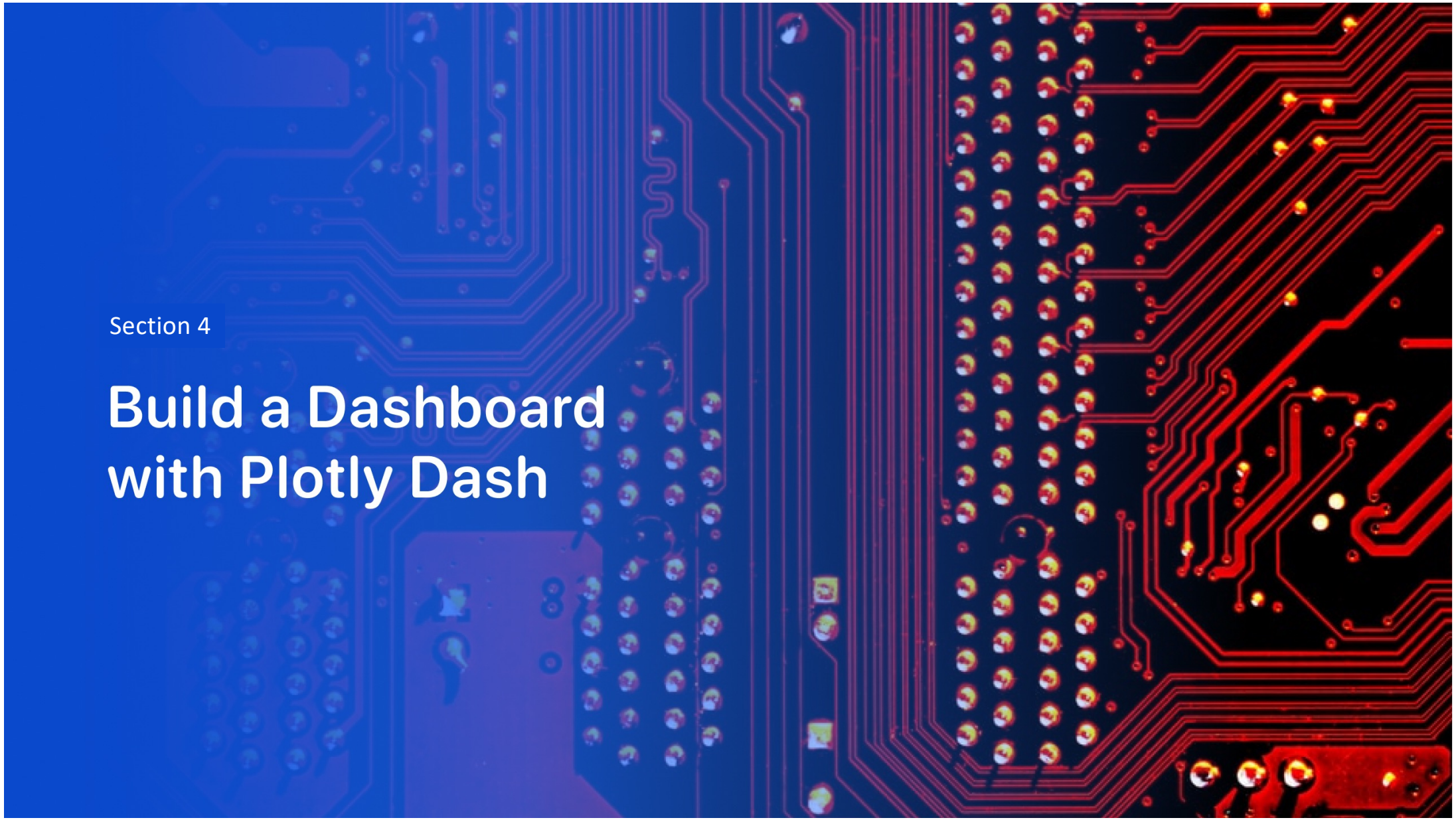
- West cost launch site undertake most of launch missions



West cost launch site nearest highway and town center

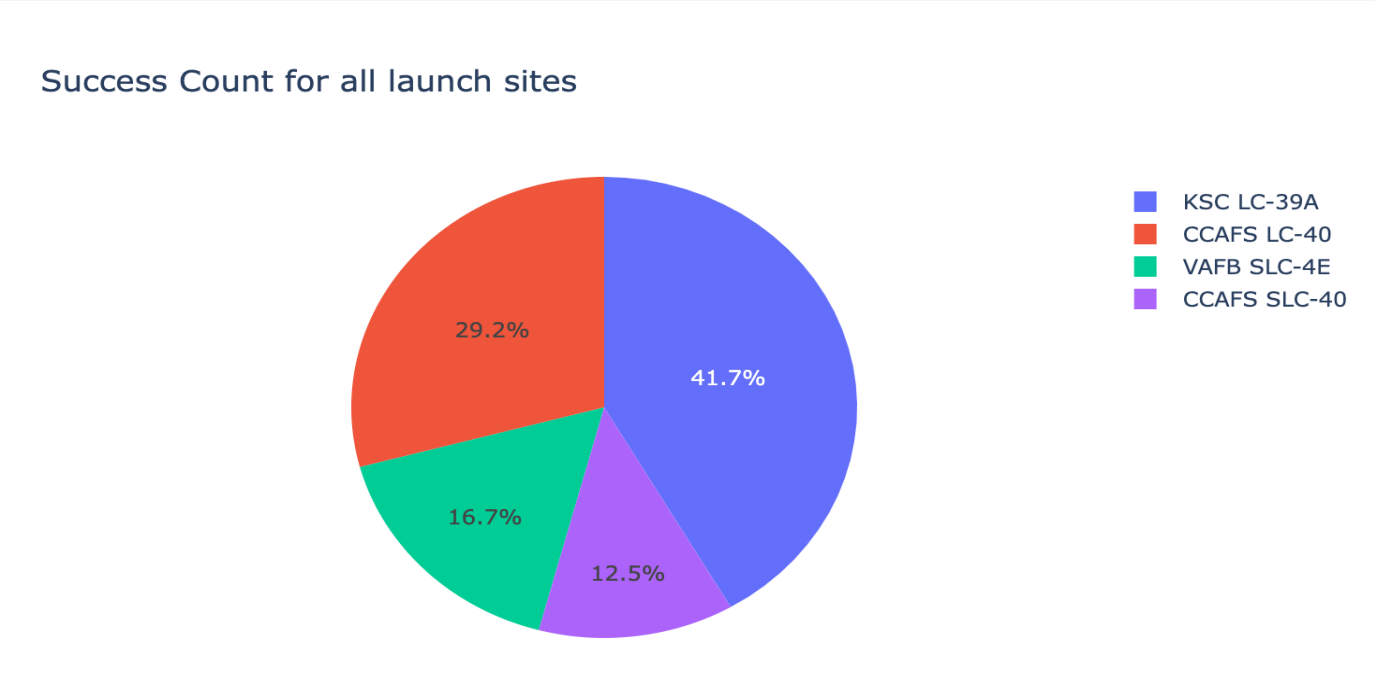
- The launch site have certian distance from the residential area, and with convenient land transportation




Section 4
Build a Dashboard with Plotly Dash

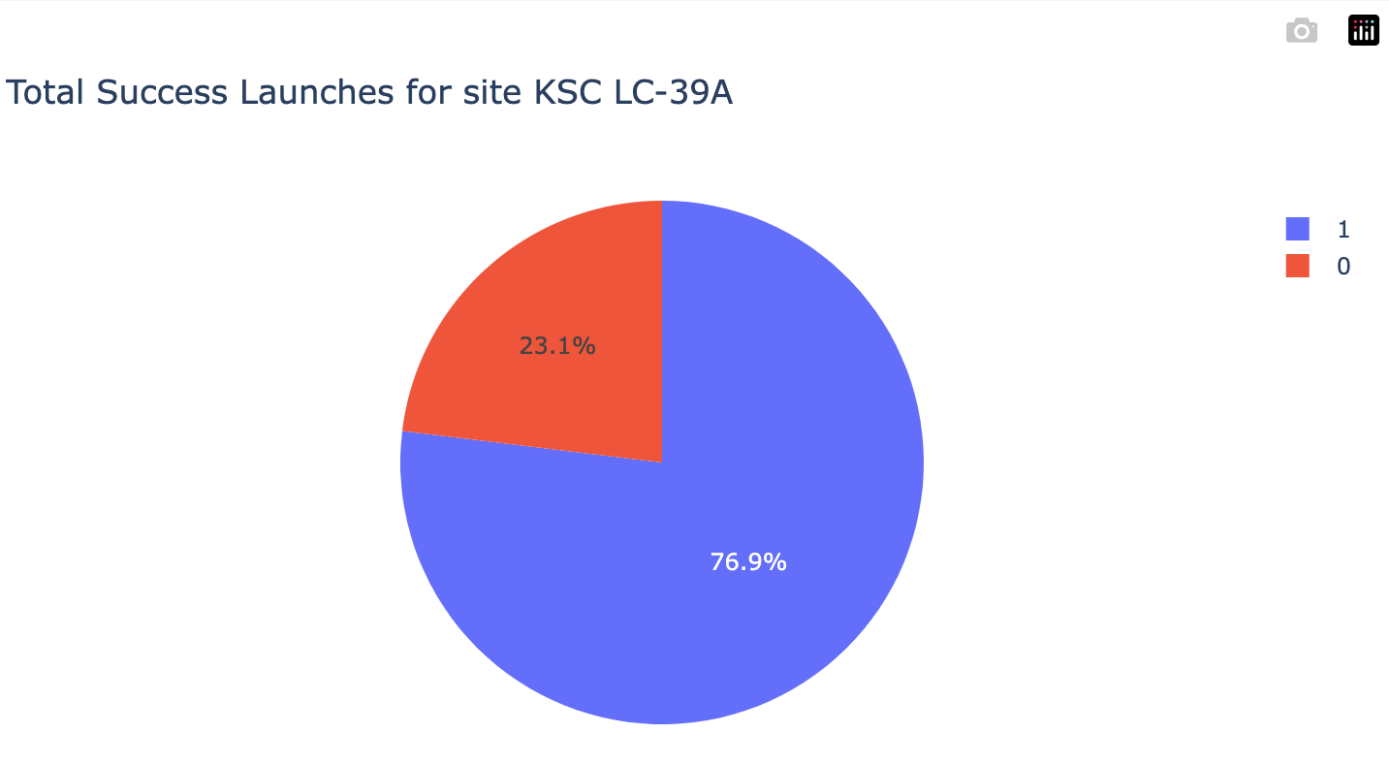
Launch site mission success rate

KSC LC-39A have half success launch mission



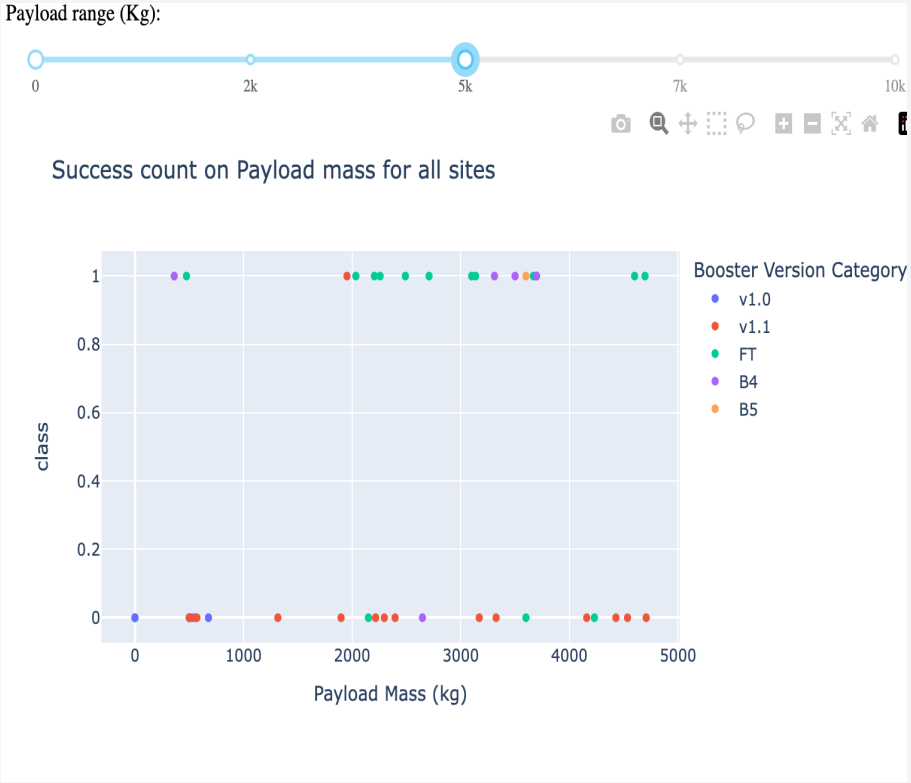
KSC LC-39A success rate

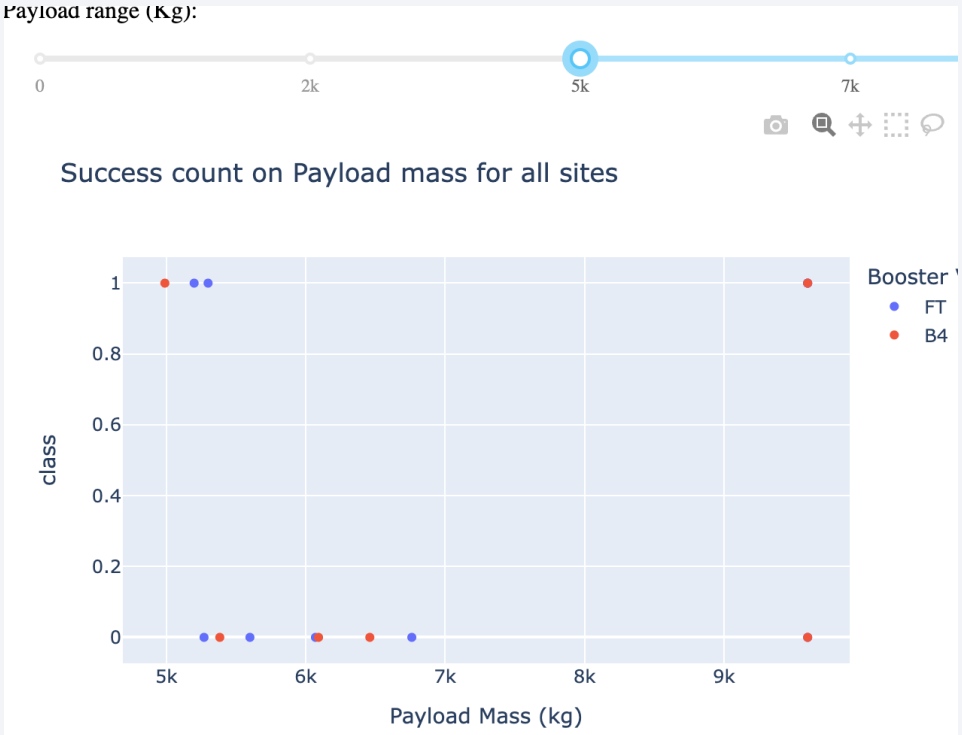
- 76.9% success rate as well 23.1% failure rate



Payload impact on launch success rate

- Low weight payload launch are with more success chance from the pass launch result





Section 5

Predictive Analysis (Classification)

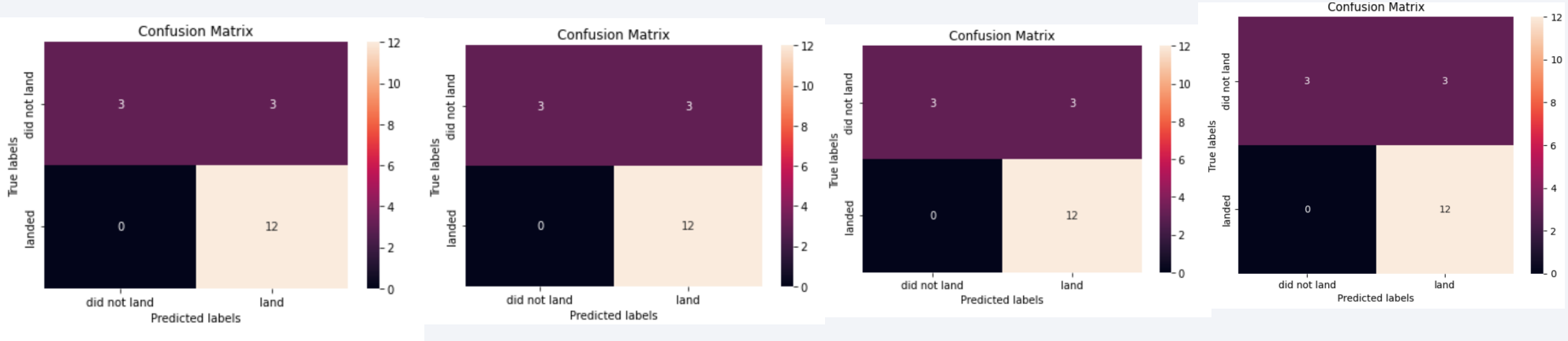
Classification Accuracy

```
score_dic = {"logreg_cv": logreg_cv.score(X_test, Y_test),
             "svm_cv": svm_cv.score(X_test, Y_test),
             "tree_cv": tree_cv.score(X_test, Y_test),
              "knn_cv": knn_cv.score(X_test, Y_test)}
max_value = max(score_dic, key=score_dic.get)
print(max_value)
```

```
logreg_cv
```

Confusion Matrix

- logistic regression. support vector. decision tree. k nearest neighbors



Conclusions

- The larger the flight number at a launch site, the greater the success rate at a launch site, large payload has negative impact on success launch
- 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.
- Logistic regression is the best machine learning algorithm for this task.

Appendix

- https://www.python.org/
- https://pandas.pydata.org/
- https://matplotlib.org/

Thank you!