IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Saad Ahmed-uz Zaman
February 7, 2022

Outline

Executive Summary

Summary of methodologies

- Data Collection
- Data Wrangling
- EDA with data visualization
- EDA with SQL
- Building an interactive map with Folium
- Building dashboard with Plotly dash
- Predictive analysis (Classification)

Summary of all results

- EDA results
- Interactive Visual Analytics results
- Predictive analysis results

Introduction

Project background and context

It’s like we are living in a science fiction movie now – commercial space travel is finally a reality! Many companies are now competing to make commercial space travel affordable for all. The most successful among them is perhaps SpaceX - an American aerospace manufacturer co-founded by South African-born American billionaire entrepreneur Elon Musk. SpaceX advertises its Falcon 9 rocket launches with a cost of 62 million dollars on its website when its competitors are asking for a minimum of 165 million dollars for similar services. SpaceX has been able to reduce costs thanks to its groundbreaking method of reusing the Falcon 9’s first stage.

In this project, we will attempt to predict if the Falcon 9 first stage will land successfully. If we can determine that the first stage will make a successful landing, it will also allow us to determine the cost of a launch. If a rival company wants to compete against SpaceX in the commercial space travel industry, this information will be very valuable for them.

Problems you want to find answers

- ➢ Correlation between each rocket variable and percentage of successful landings
- ➢ Conditions which will ensure the best successful landing rates

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX API & Web Scraping Falcon 9 and Falcon Heavy Launches Records from Wikipedia
- Perform data wrangling
  - Convert outcomes into training Labels with the booster successfully/unsuccessful landed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Find best Hyperparameter for SVM, Classification Trees and Logistic Regression

Data Collection

- The data collection process includes a combination of API requests from the SpaceX API and web scraping data from a Wikipedia page of SpaceX, Falcon 9 and Falcon Heavy Launches Records.
- The columns obtained from SpaceX Data Column are: *FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude and Latitude*
- The columns obtained from Web Scraping the Wikipedia page are: *Flight No., Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Version Booster, Booster landing, Date and Time*

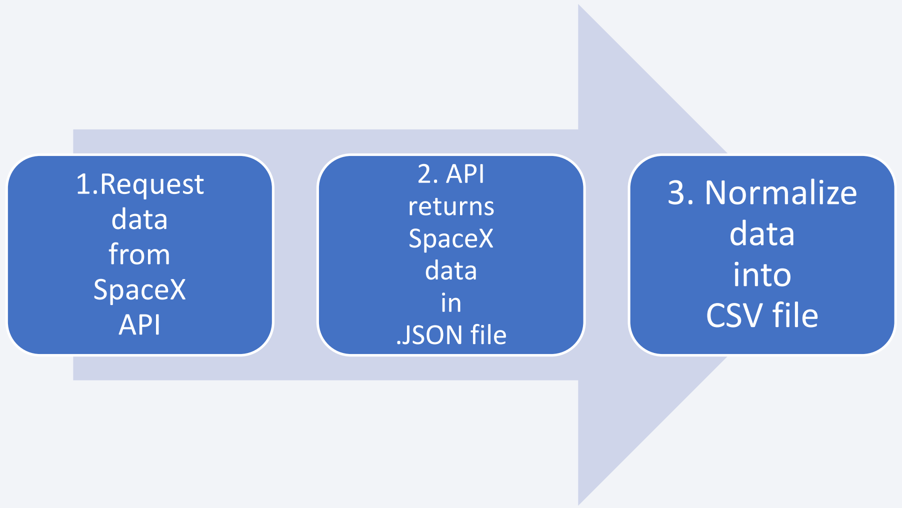

1.Request data from SpaceX API
2. API returns SpaceX data in .JSON file
3. Normalize data into CSV file

Fig. Space X API process steps

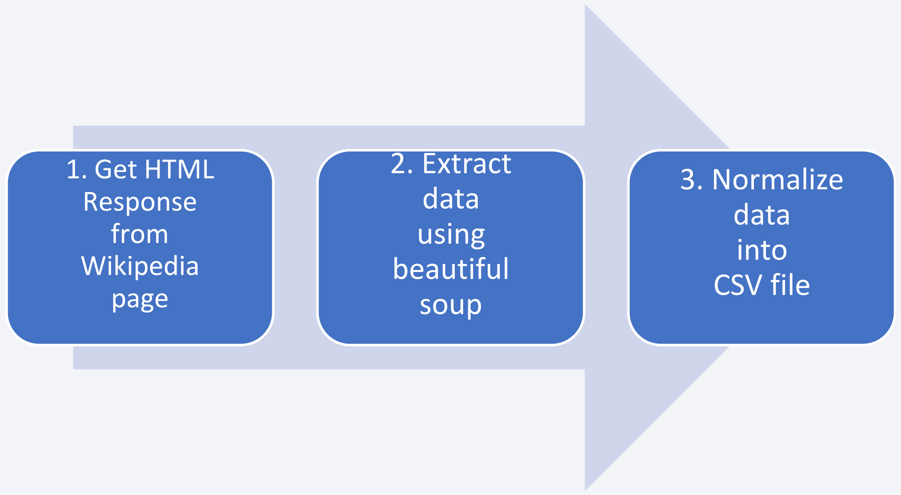
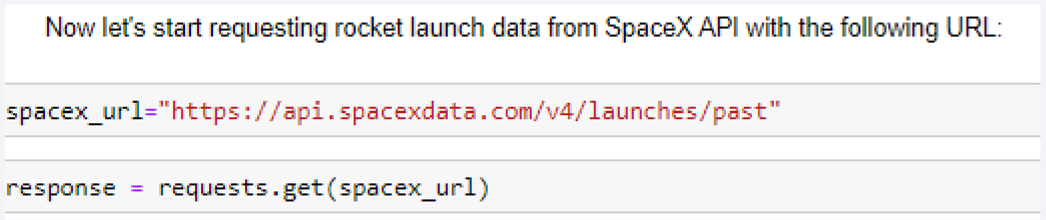

1. Get HTML Response from Wikipedia page
2. Extract data using beautiful soup
3. Normalize data into CSV file

Fig. Web scraping process steps

Data Collection - SpaceX API

1. Requesting rocket launch data from SpaceX API

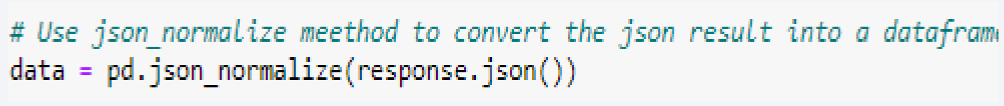


Now let's start requesting rocket launch data from SpaceX API with the following URL:

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

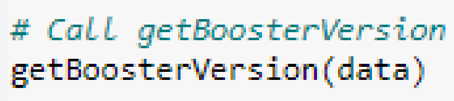
```
response = requests.get(spacex_url)
```

2. Convert Response to a JSON file.

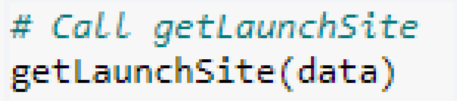


```
# Use json_normalize meethod to convert the json result into a datafram
data = pd.json_normalize(response.json())
```

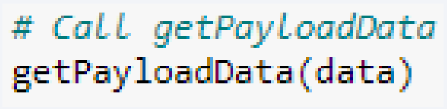
3. Data cleaning using functions.



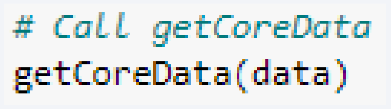
```
# Call getBoosterVersion
getBoosterVersion(data)
```



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



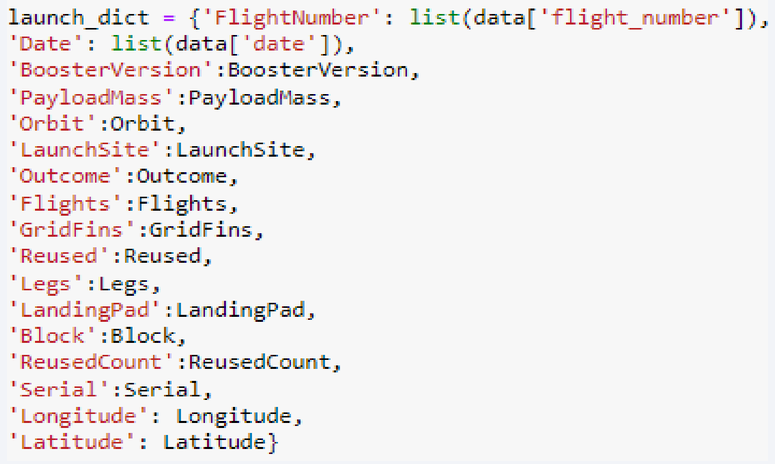
```
# Call getPayloadData
getPayloadData(data)
```



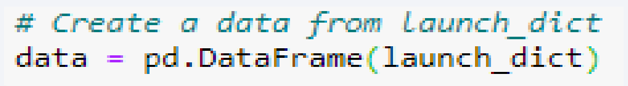
```
# Call getCoreData
getCoreData(data)
```

- Github URL link: https://github.com/saad20201/Coursera-Saad/blob/master/Data%20Collection%20API.ipynb

4. Combine the columns into a dictionary to create data frame.

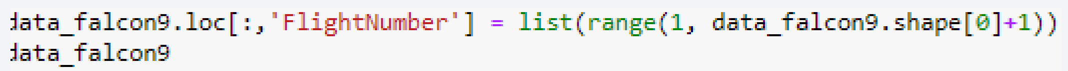


```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,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

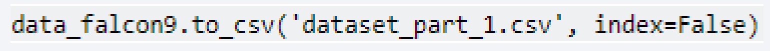


```
# Create a data from launch_dict
data = pd.DataFrame(launch_dict)
```

5. Exporting the dataframe to a csv file



```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```



```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

1. Getting HTML response using HTTP GET

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

2. Create a BeautifulSoup object from the response

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

3. Extract all column/variable names from the HTML table header

```
# 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')
```

4. Extract the names one by one.

```
column_names = []
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if name != None and len(name) > 0:
        column_names.append(name)
```

- GitHub URL Link: https://github.com/saad20201/Coursera-Saad/blob/master/Data%20Collection%20with%20Web%20Scraping.ipynb

5. 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 value to be an empty list
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']=[]
```

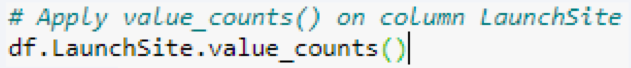
6. Create dataframe and export to a csv file.

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

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

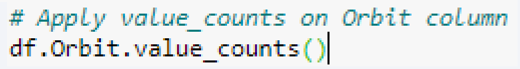
Data Wrangling

- Calculate the number of launches on each site.



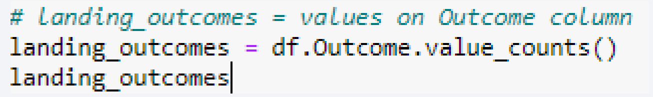
```
# Apply value_counts() on column LaunchSite
df.LaunchSite.value_counts()
```

- Calculate the number and occurrence of each orbit.



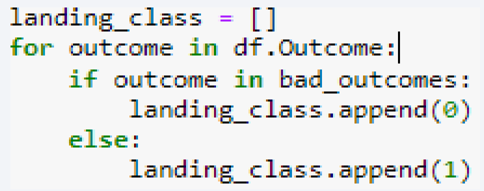
```
# Apply value_counts on Orbit column
df.Orbit.value_counts()
```

- Calculate the number and occurrence of mission outcome per orbit.

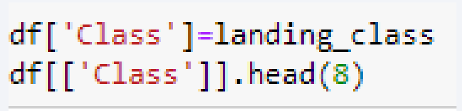


```
# landing_outcomes = values on Outcome column
landing_outcomes = df.Outcome.value_counts()
landing_outcomes
```

- Create a landing outcome label from Outcome column.

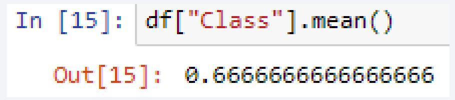


```
landing_class = []
for outcome in df.Outcome:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```



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

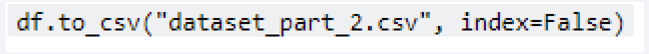
- We can use the following line of code to determine the success rate.



```
In [15]: df["Class"].mean()

Out[15]: 0.6666666666666666
```

- Finally export data to the csv file.



```
df.to_csv("dataset_part_2.csv", index=False)
```

- GitHub URL link: https://github.com/saad20201/Coursera-Saad/blob/master/EDA.ipynb

EDA with Data Visualization

- The first type of chart that is plotted is the scatter plot which shows how a variable is affected by another and this relationship is defined as "Correlation". The following scatter plots are drawn in the project:

➢ **Flight Number vs. Launch Site**

➢ **Payload vs. Launch Site**

➢ **Flight Number vs. Orbit Type**

➢ **Payload vs. Orbit Type**

- The second type of chart used in the project is the bar chart which makes it easy for viewers to compare datasets just from a single glance. The only bar chart used here is the **"Orbit type vs success rate"** chart.
- The third and final type of chart used is line chart which is used to show trends between variables clearly. The one we used here is titled **"Year vs success rate".**
- Github URL link: https://github.com/saad20201/Coursera-Saad/blob/master/EDA%20for%20data%20visualization.ipynb

EDA with SQL

- The following SQL queries were performed during this lab:

➢ 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 acheived.

➢ List the names of boosters which have success in drone ship and have payload mass between 4000 and 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

- Github URL Link: https://github.com/saad20201/Coursera-Saad/blob/master/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- **Objects created and added to a folium map** –
- ➢ Markers showing all the launch sites on a map
- ➢ Markers showing the successful and failed launches for each site on the map
- ➢ Lines that show the distances between a launch site to nearby locations like highway, railway and so on.
- By adding the above objects, it was determined that the launch sites were in close proximity to **railways, highways** and **coastlines** but further away from **cities**.
- GitHub URL Link: https://github.com/saad20201/Coursera-Saad/blob/master/Interactive%20Visual%20Analytics%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

- The dashboard application contains a pie chart and a scatter point chart. The purpose of the pie chart is to display the following:

➢ Total number of successful launches by each site.

➢ The success rate of landing across all the sites as well as each individual site.

- The purpose of the scatter chart is to display the following:

➢ Relationship between Outcomes and Payload mass of different boosters using a slider.

➢ Determine how success depends on the launch point, payload mass and booster types.

Github URL Link (only code included here, screenshots coming in later slides):
https://github.com/saad20201/Coursera-Saad/blob/master/Plotly_dash_code

Predictive Analysis (Classification)

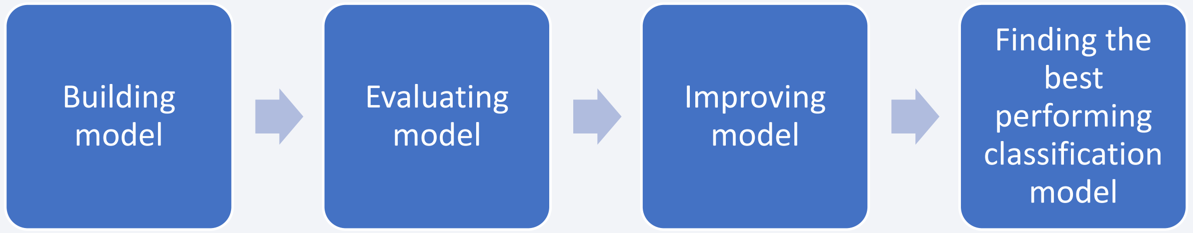
- We performed exploratory data analysis and determined training labels. These 3 steps were involved:

➢ Create a column for the class

➢ Standardize the data

➢ Split into training data and test data

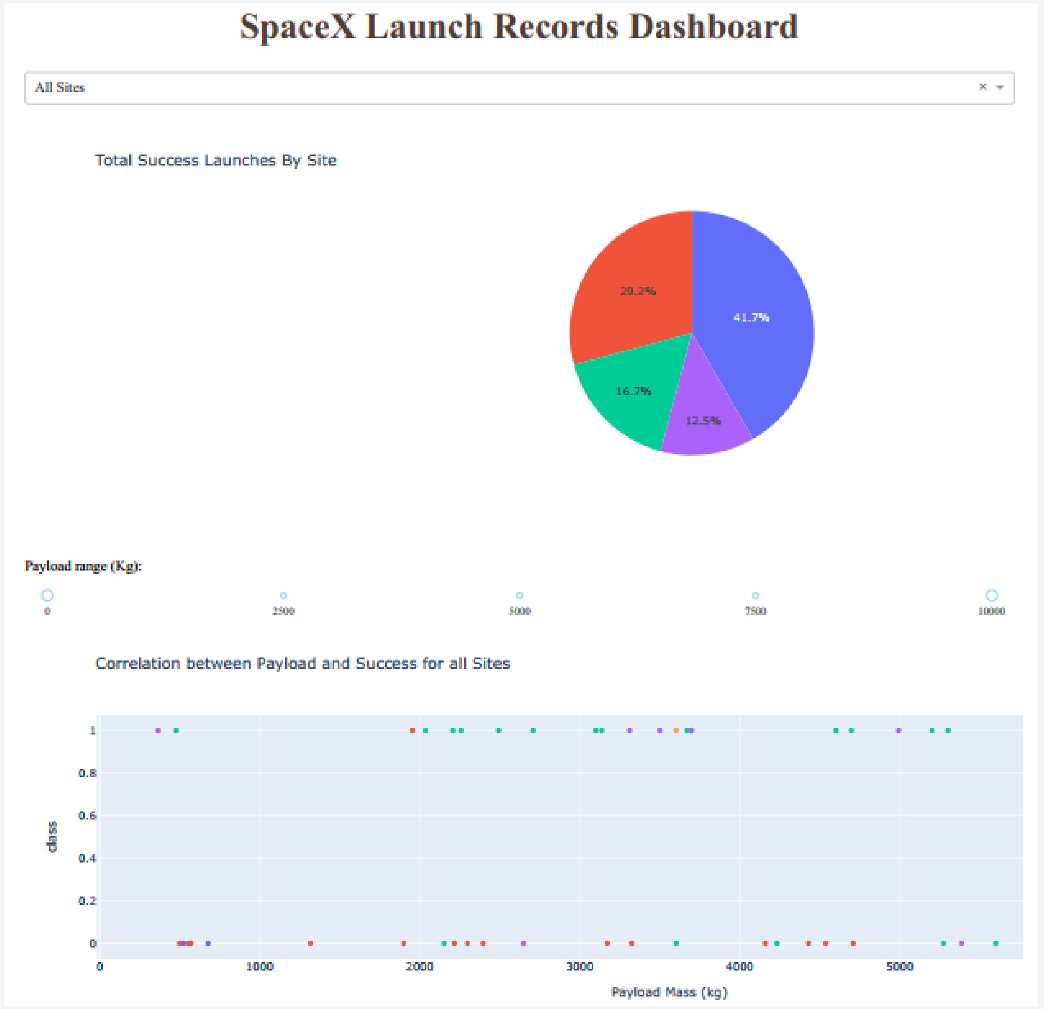
- Then we calculated best Hyperparameter for SVM, Classification Trees and Logistic Regression and found the method that performs best using test data.

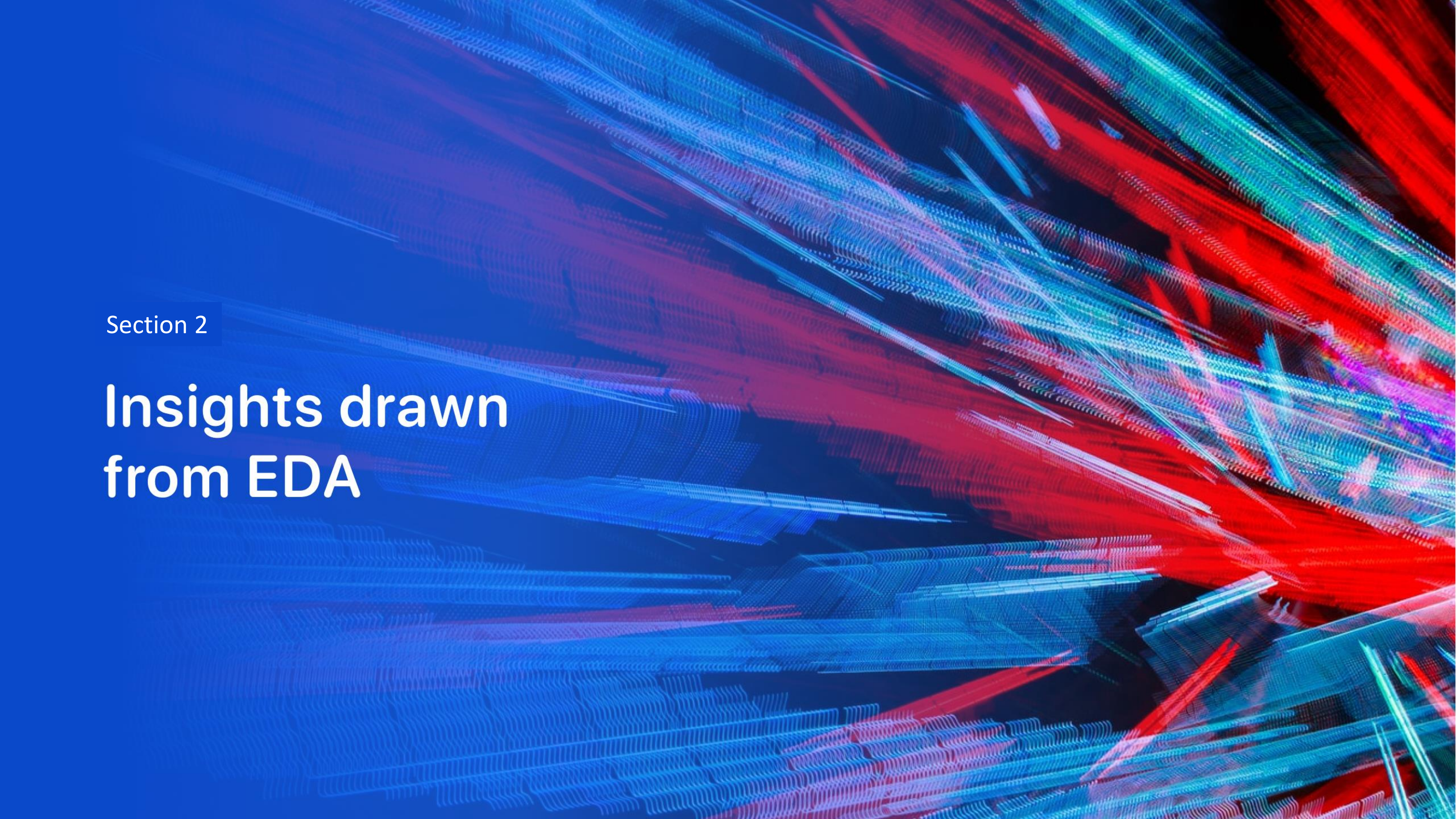


GitHub URL link: https://github.com/saad20201/Coursera-Saad/blob/master/Predictive%20analysis.ipynb

Results

- The screenshot on the right is taken from my Dashboard with Plotly dash.
- The upper part shows total success launches by site
- The lower part shows us the correlation between payload mass (in kg) and class.



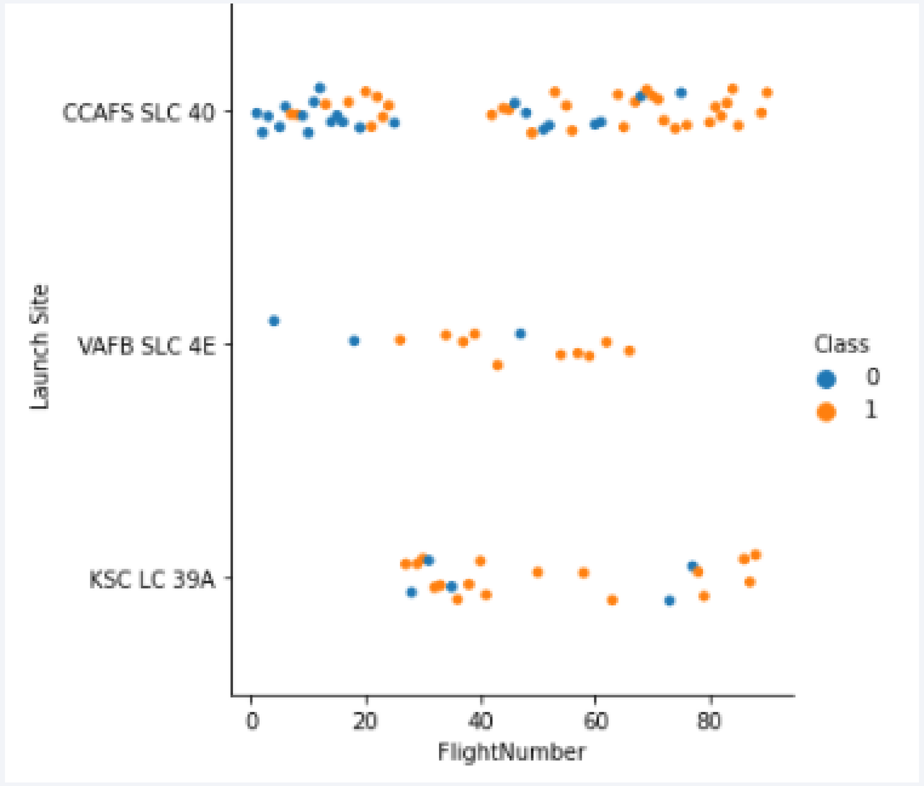


Section 2

Insights drawn from EDA

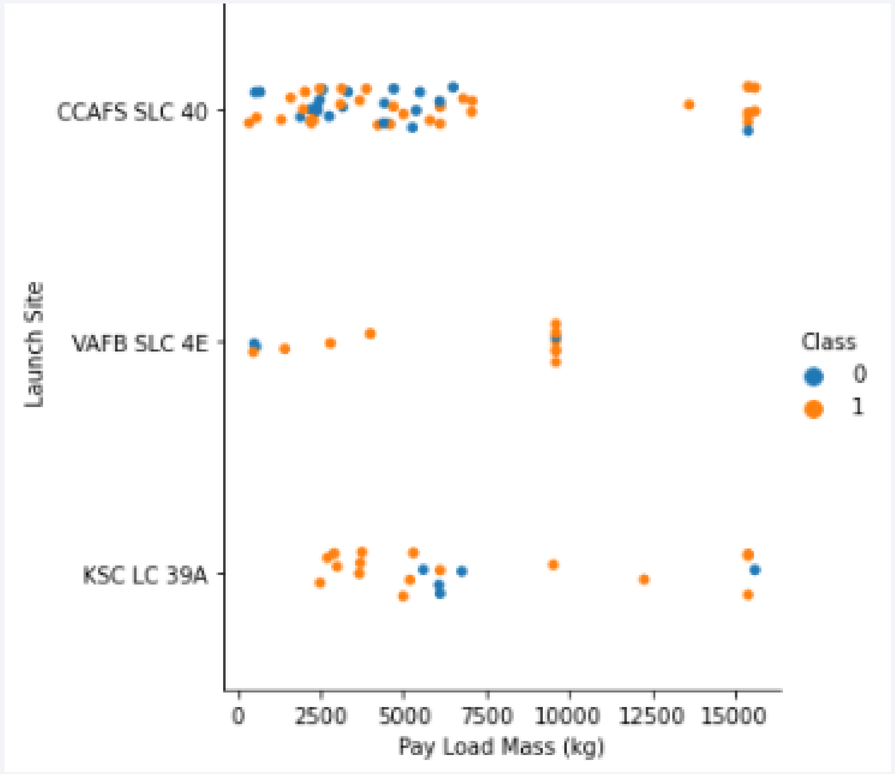
Flight Number vs. Launch Site

- Here, in the scatter plot, blue dots represent launch failures while the orange ones represent successful launches.
- The main pattern I am able to observe is that the number of successful launches rise as the number of flights increase.



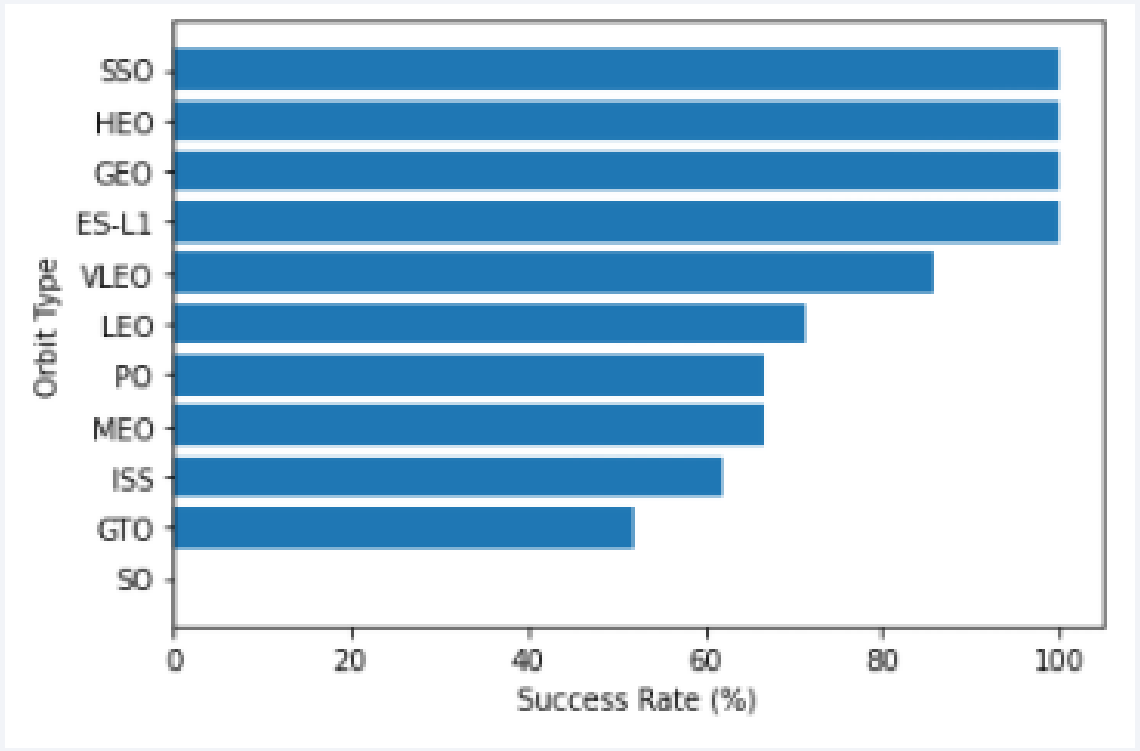
Payload vs. Launch Site

- Like the previous graph, the orange dots represent successful while the blue ones unsuccessful launches as well
- If we observe this scatter point chart we will notice that for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).



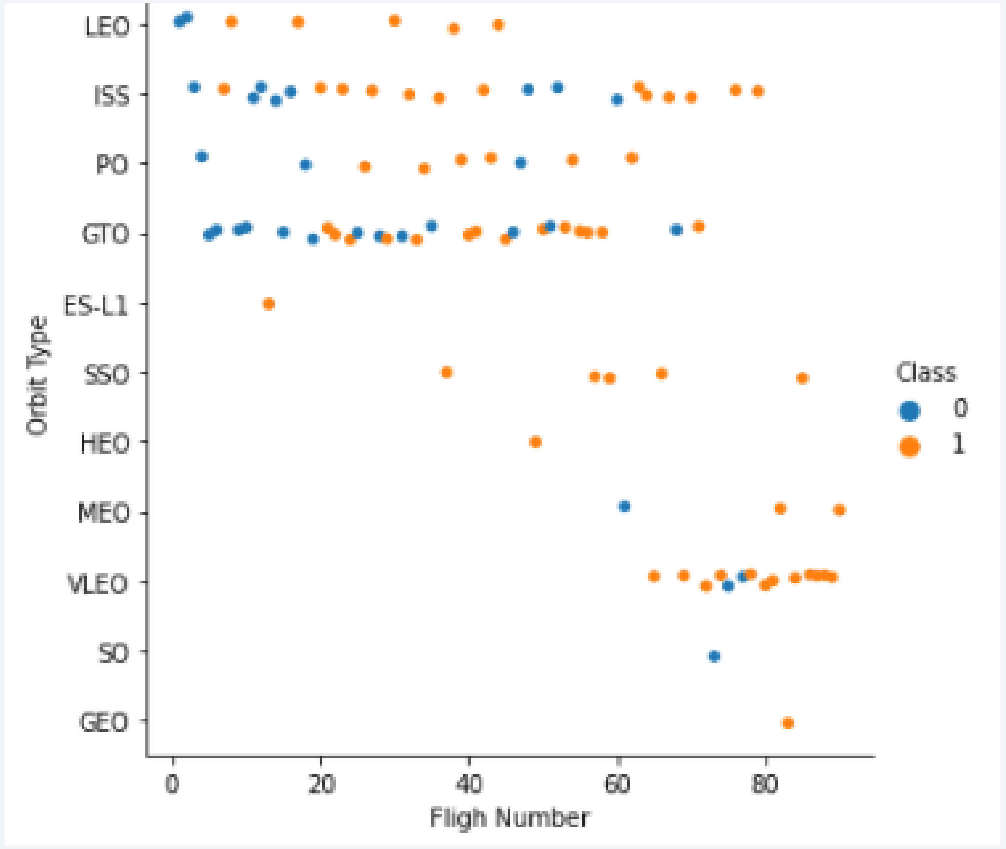
Success Rate vs. Orbit Type

- The bar chart shows that the first four orbit types SEO, HEO, GEO and ES-L1 have 100% success rates.
- The lowest recorded success rate belongs to GTO at 50% (if we ignore SO which had just 1 launch and that ended in a failure).



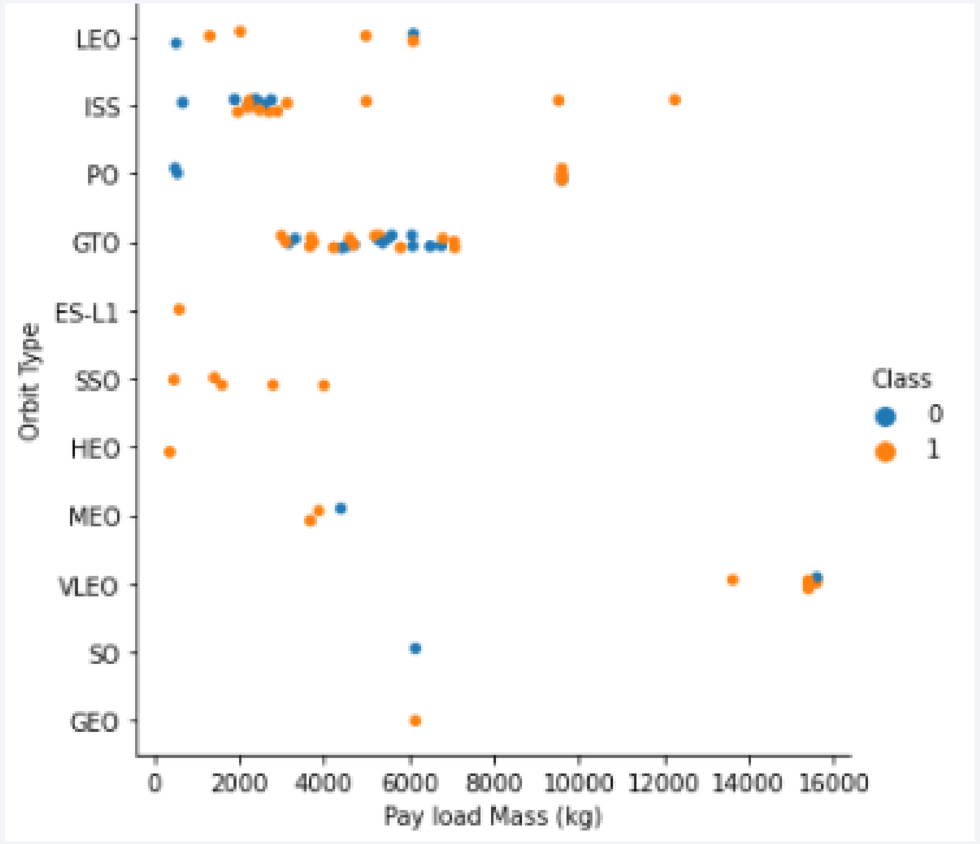
Flight Number vs. Orbit Type

- Like earlier plots, the orange dots are successful launches and the blue one are failures.
- For most orbit types, the number of successful launches go up with increase in flight numbers
- However, there seems to be no relationship between flight number when in GTO orbit.



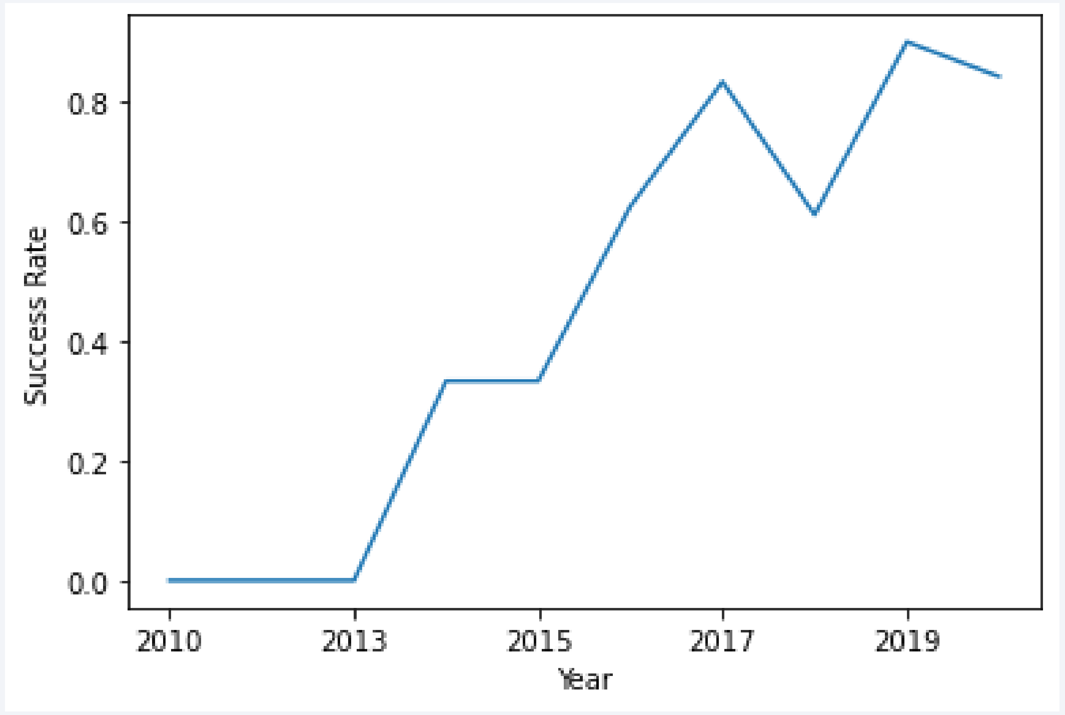
Payload vs. Orbit Type

- Like earlier plots, the orange dots are successful launches and the blue one are failures.
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- However, for GT, like before there seems to be both positive rate and negative landing throughout the payload masses.



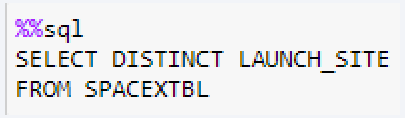
Launch Success Yearly Trend

- We can see that launch success rate has been going up since 2013.
- The peak seems to be in 2019 when the success rate reached close to 90%.
- 2020 has seen a drop from the previous year at around 80%.



All Launch Site Names

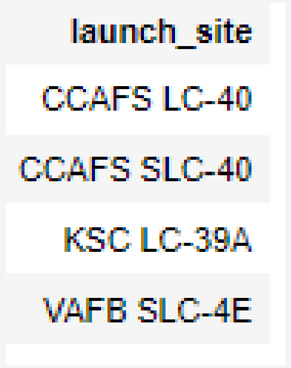
- We use the query:



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

- The list of unique launch site names are CCAFS LC-40, CCAFS SLC-40,

KSC LC-39A and VAFB SLC-4E

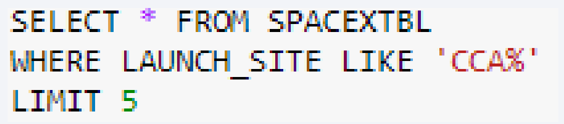


launch_site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

- We used the DISTINCT clause in our query to display only the unique values from the Launch_site column.

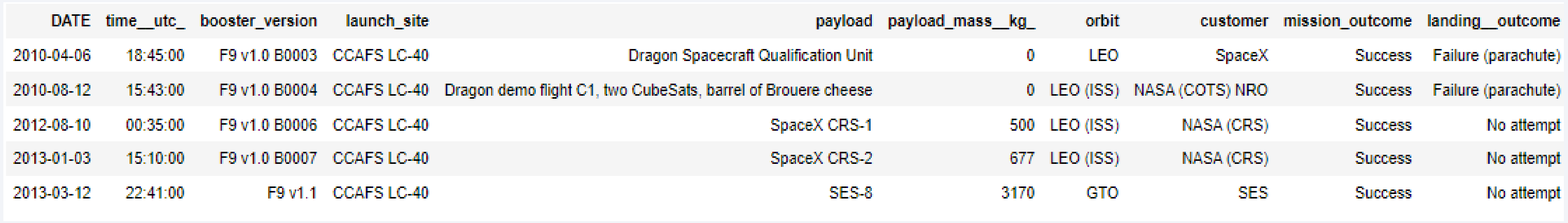
Launch Site Names Begin with 'CCA'

- We use the following query here:



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

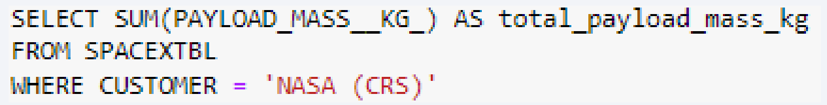
- We used the LIKE operator and ‘%’ sign along with the **limit 5** clause in our query and the following 5 records were obtained.



DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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)
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-12	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170	GTO	SES	Success	No attempt

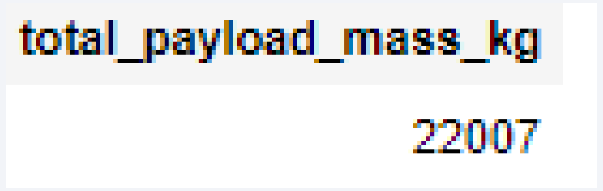
Total Payload Mass

- We used the following query here:



```
SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload_mass_kg
FROM SPACEXTBL
WHERE CUSTOMER = 'NASA (CRS)'
```

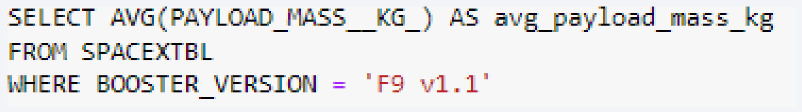
- The SUM() function calculates the total payload mass. We use the WHERE clause to carry out that function only if the NASA (CRS) is the customer.
- The total payload mass obtained is shown in the screenshot below.



total_payload_mass_kg
22007

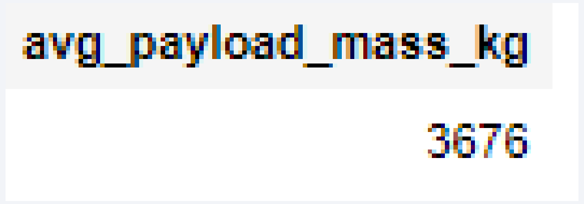
Average Payload Mass by F9 v1.1

- The query used here was:



```
SELECT AVG(PAYLOAD_MASS__KG_) AS avg_payload_mass_kg
FROM SPACEXTBL
WHERE BOOSTER_VERSION = 'F9 v1.1'
```

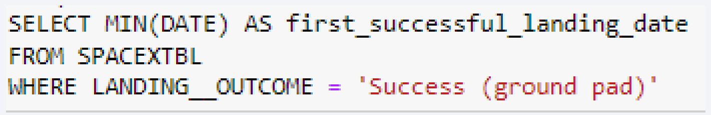
- We used the AVG() function to calculate the average payload mass value.
- The WHERE clause is used to separate only the values when the booster version is F9 v1.1.
- The calculated value of the average payload mass by F9 v1.1 is shown below:



avg_payload_mass_kg
3676

First Successful Ground Landing Date

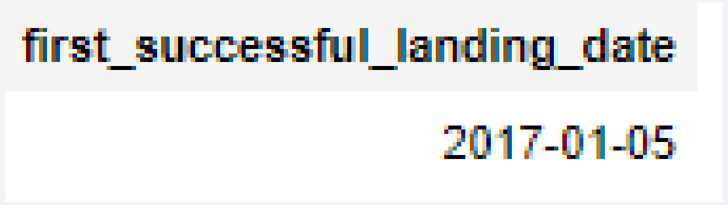
- The query used here was:



```
SELECT MIN(DATE) AS first_successful_landing_date
FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Success (ground pad)'
```

- The MIN() function is used to locate the earliest date while the WHERE clause is used filter only those values where the landing outcome is successful.

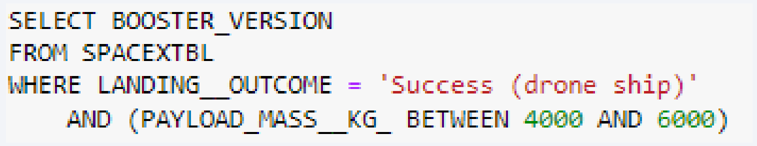
- The first successful landing date obtained is shown below.



first_successful_landing_date
2017-01-05

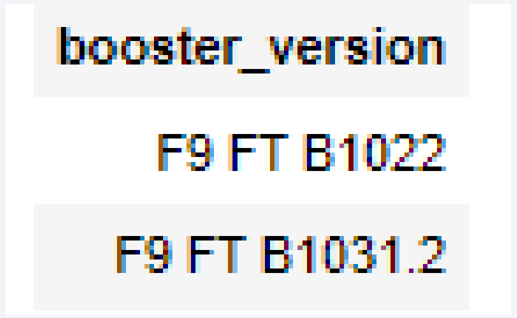
Successful Drone Ship Landing with Payload between 4000 and 6000

- The query we used here is:



```
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Success (drone ship)'
    AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)
```

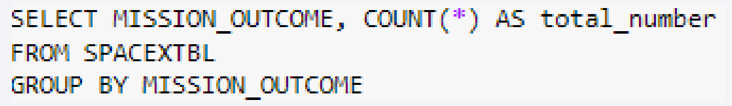
- The WHERE clause is used to filter only those values when landing outcome equals ‘Success (drone ship)’ and we use the AND operator to add the BETWEEN 4000 and 6000 condition.
- The following results were obtained:



booster_version
F9 FT B1022
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

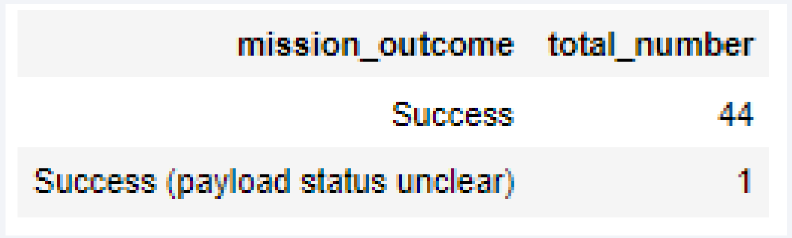
- The query used here is:



```
SELECT MISSION_OUTCOME, COUNT(*) AS total_number
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME
```

- The COUNT() function is used to calculate total number of columns and the GROUP BY statement is used to gather the same values and the calculate the total value in each mission outcome.

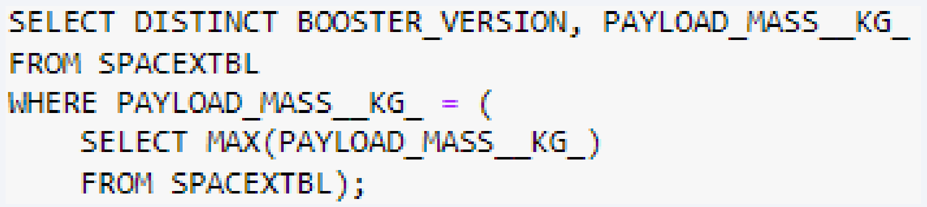
- The results obtained are shown in the screenshot below:



mission_outcome	total_number
Success	44
Success (payload status unclear)	1

Boosters Carried Maximum Payload

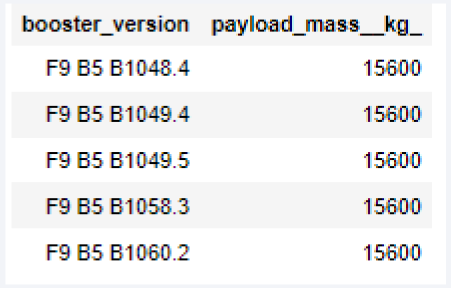
- The query used here is:



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

- The MAX() function is used to locate the maximum payload value while WHERE is used to perform search only when PAYLOAD_MASS_KG_ is the maximum payload value.

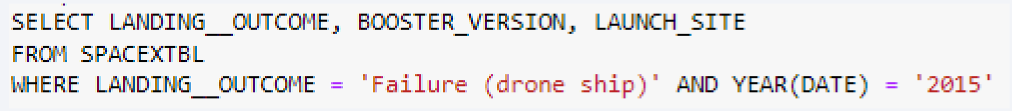
- The results obtained are shown in the screenshot below:



booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600

2015 Launch Records

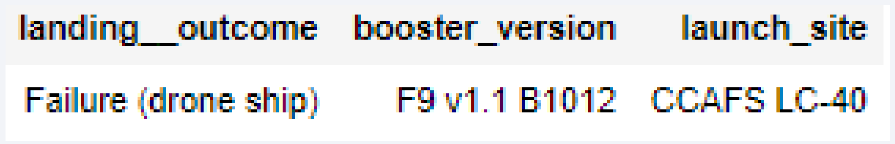
- We used the following query here:



```
SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND YEAR(DATE) = '2015'
```

- We used the WHERE clause to perform search only when the value of landing outcome equals Failure (drone ship). AND operator is used to add the year 2015 condition.

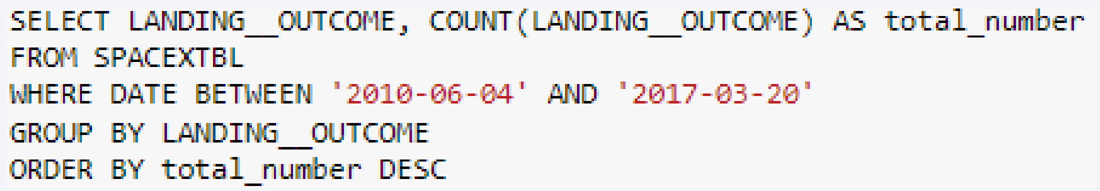
- The following results were obtained:



landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40

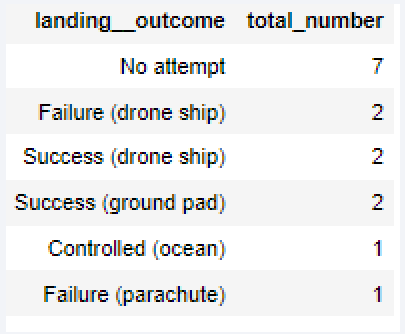
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- We used the following query here:



```
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS total_number
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY total_number DESC
```

- The WHERE clause is used here to perform search between the two specified dates.
- ORDER BY is used to sort the records by total number of landings and DESC is used to sort the records in descending order.
- The following results were obtained:



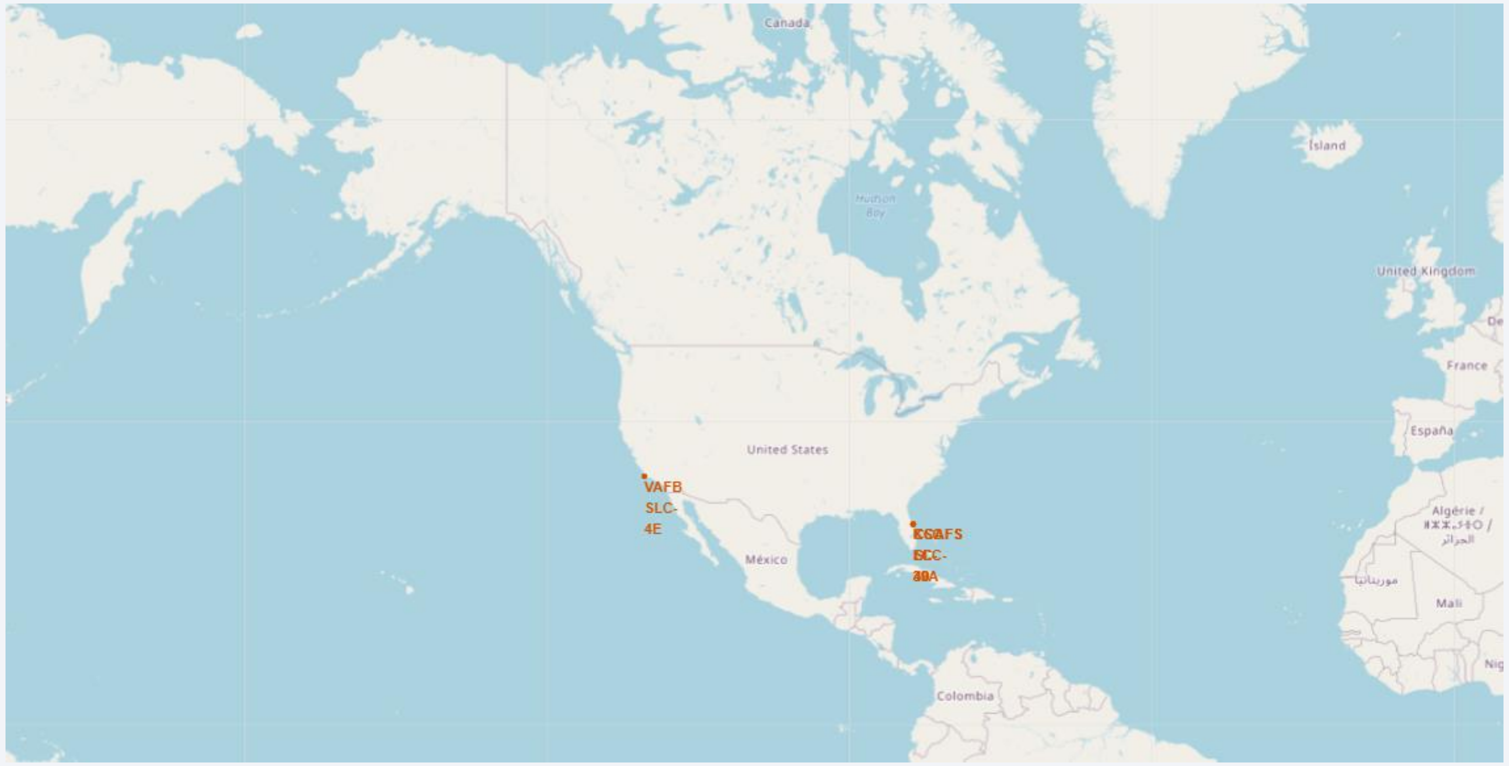
landing__outcome	total_number
No attempt	7
Failure (drone ship)	2
Success (drone ship)	2
Success (ground pad)	2
Controlled (ocean)	1
Failure (parachute)	1

Section 3

Launch Sites Proximities Analysis

Launch site locations

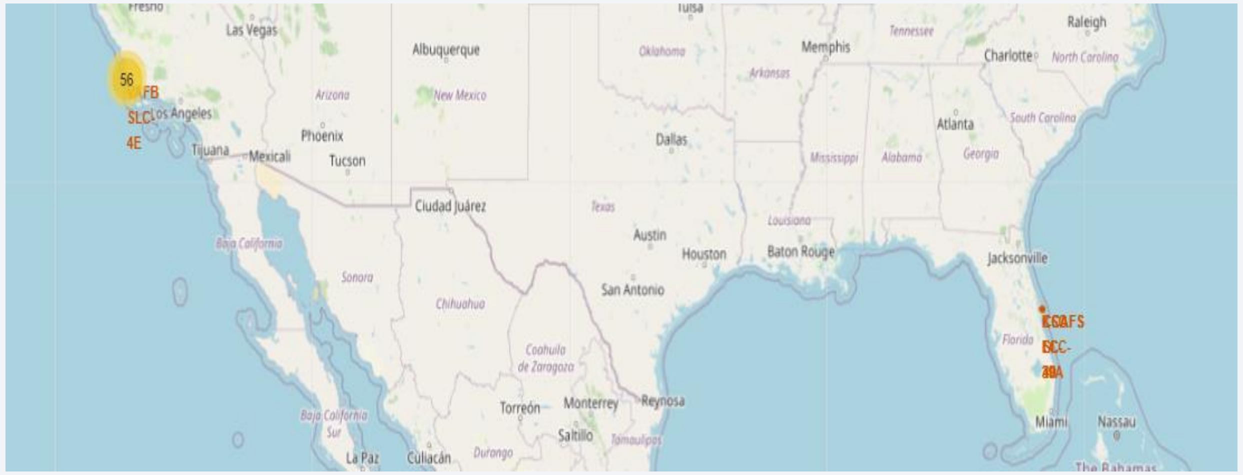
- The map below shows all the SpaceX launch site locations.

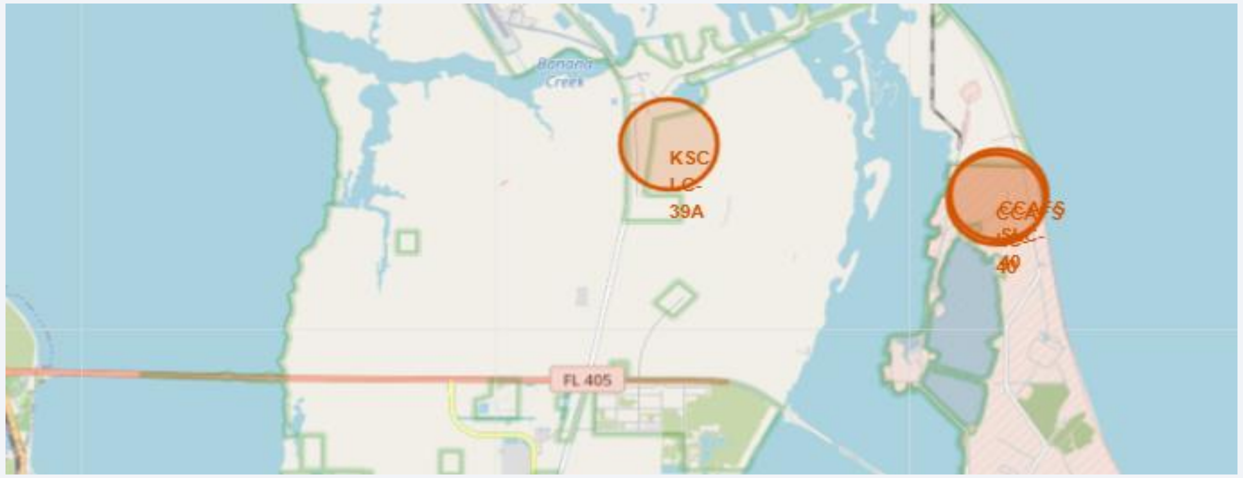


- It’s quite apparent from the map that all launch site locations are close to the coast.

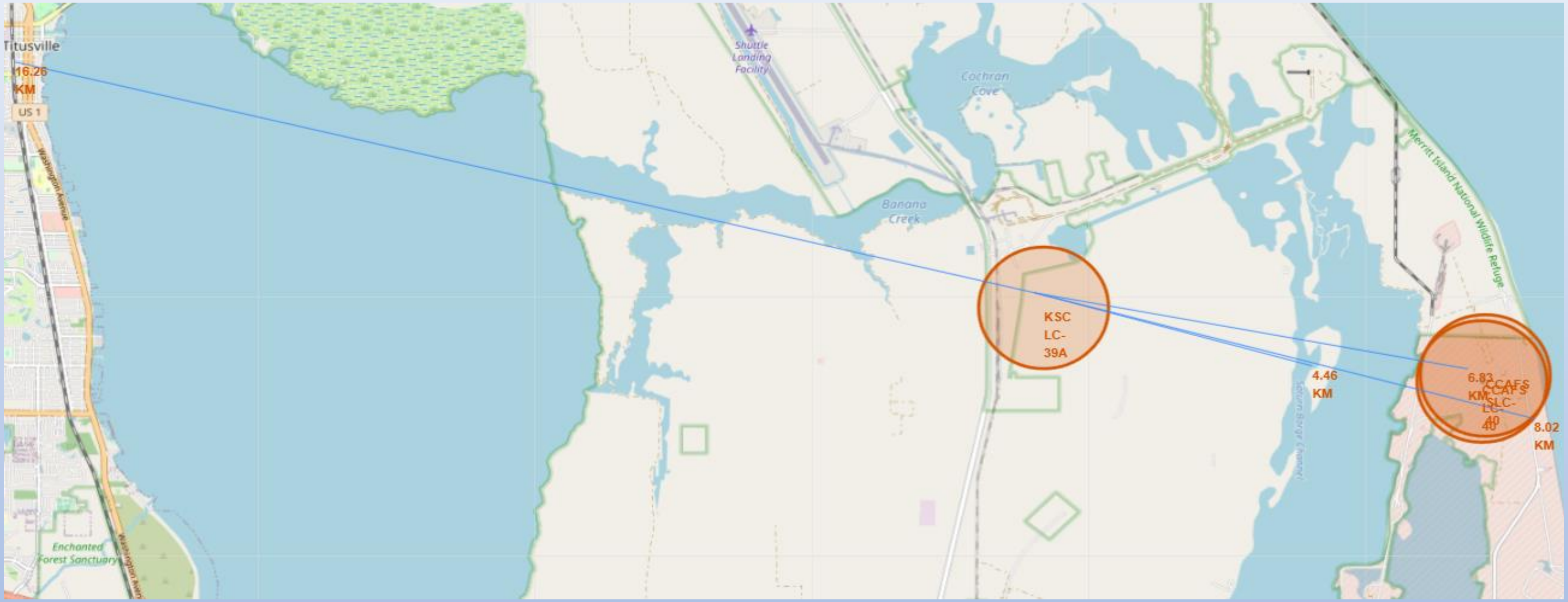
Color marked launch outcomes

➢The maps on the right show color marked launch sites (the lower one is zoomed in while the upper one is zoomed out).





Launch site proximities

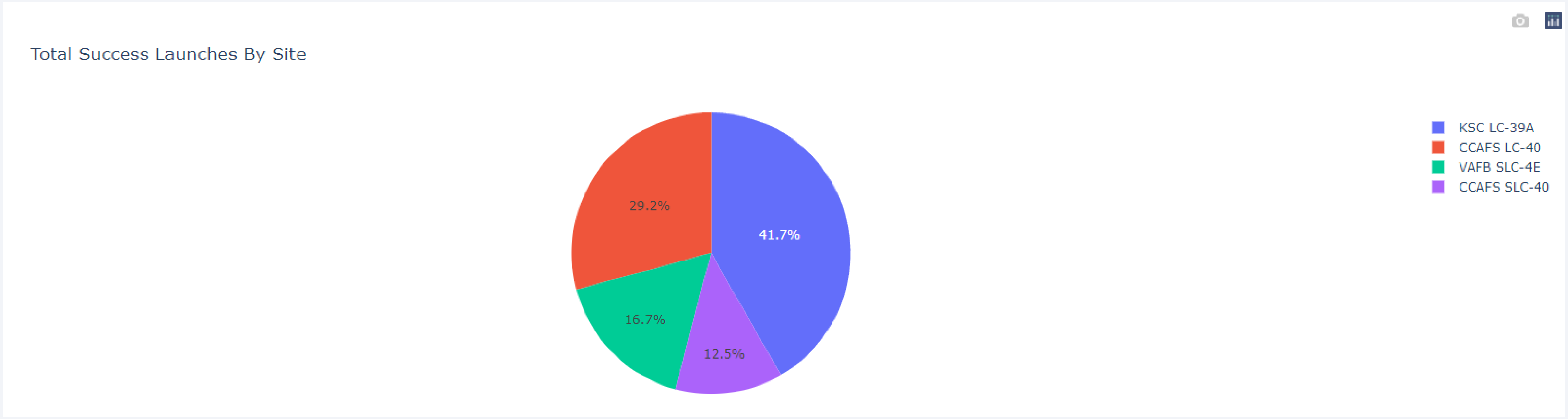


- We can see from the above screenshot of the zoomed in map that the launch site is quite close to highways and railway stations (to allow easier transportation of equipment and personnel) but farther away from cities to keep civilians out of harm's way in case of launch failures.

Section 4

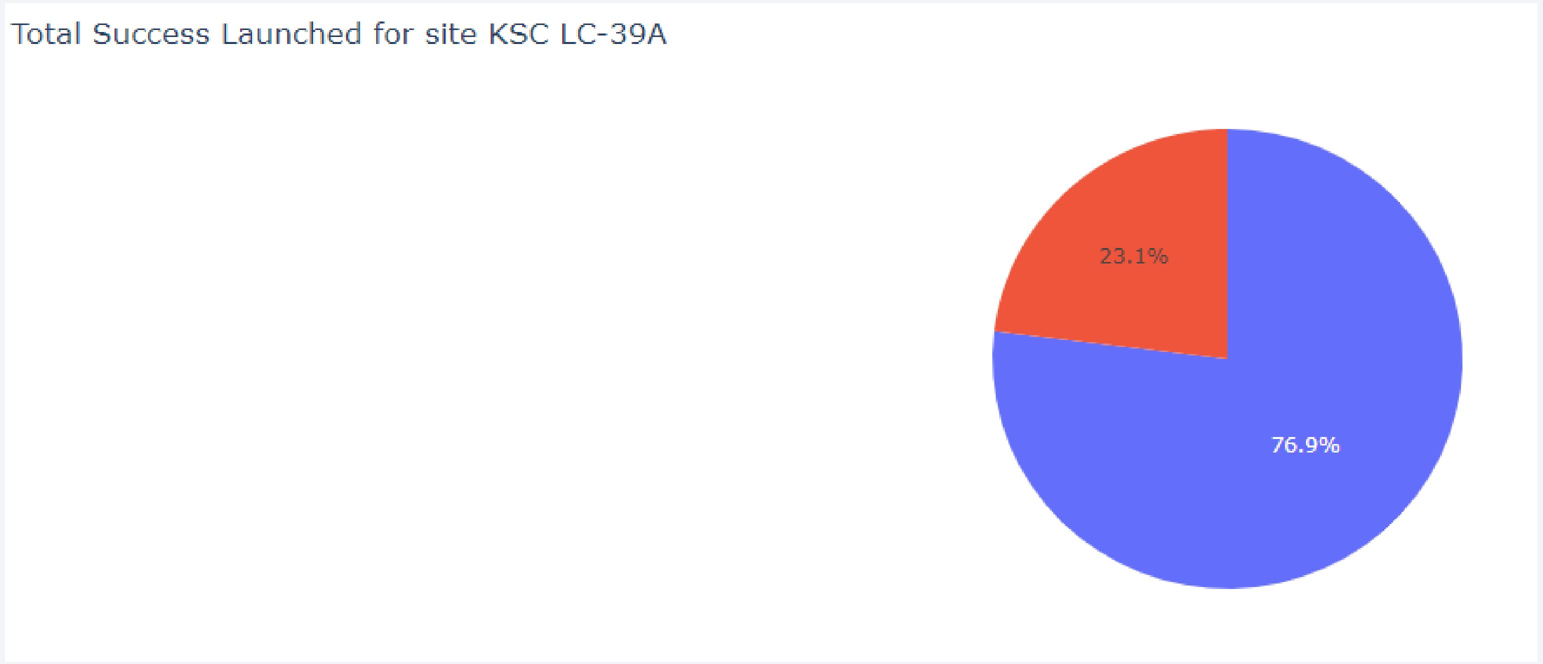
Build a Dashboard with Plotly Dash

Total successful launches by site



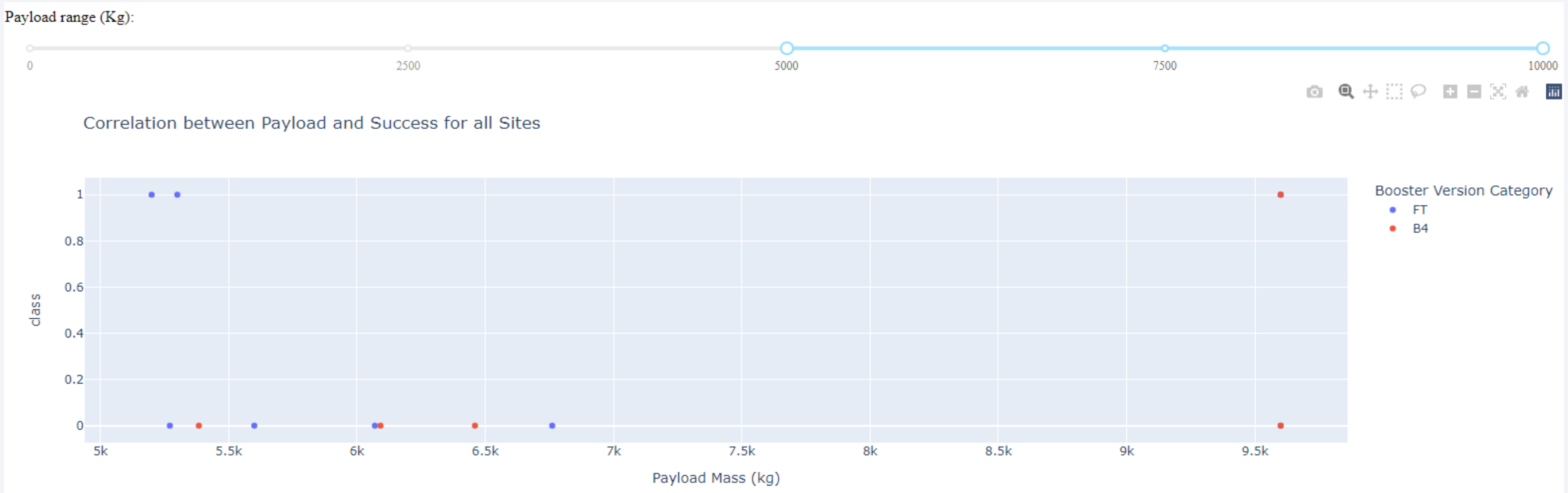
- We can observe from the above pie chart that KSC LC-39A has the highest number of successful launches.

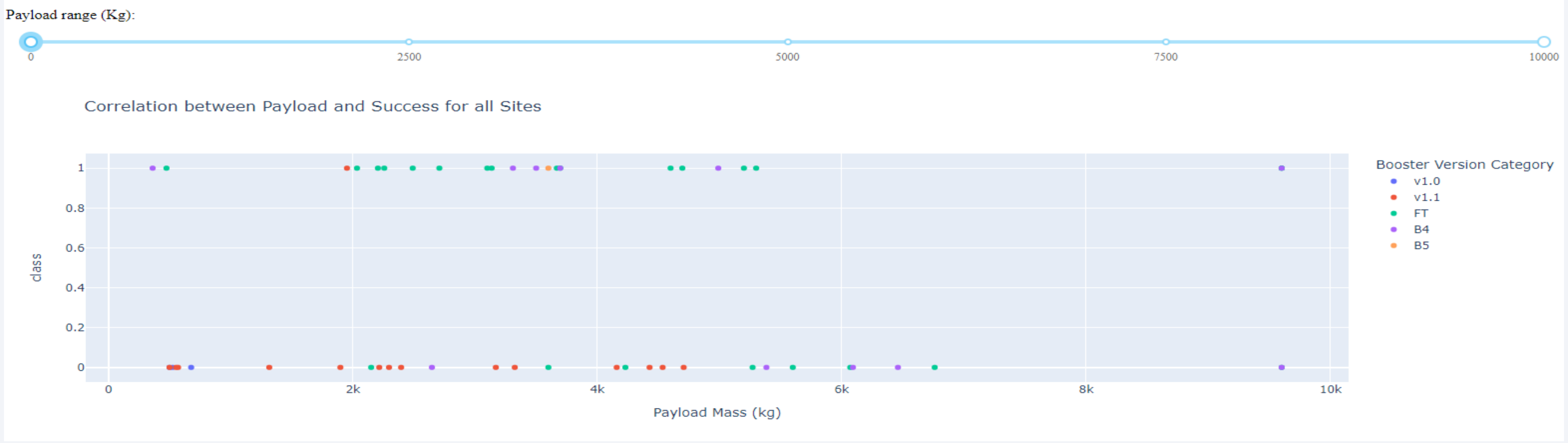
Launch Site with Highest Launch Success Percentage



- It can be seen from the above pie chart that the site KSC LC-39A has the best launch success rates with 76.9%.

Correlation between payload and success for all sites





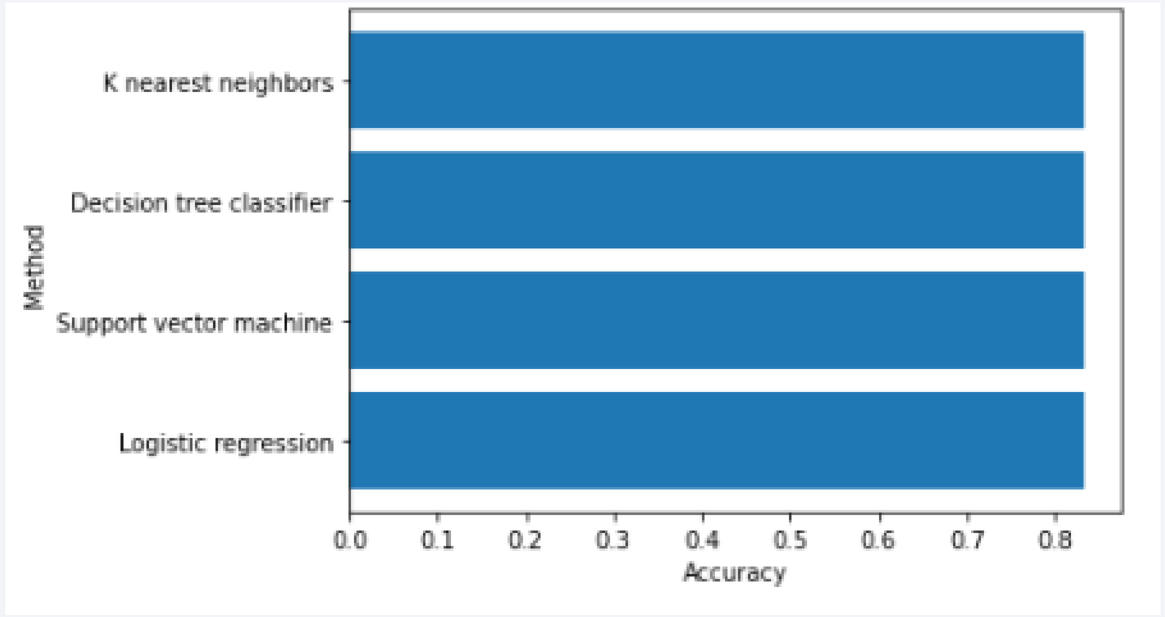
Looking at the two screenshots on the left we can observe that the launch success rates for Payloads lower than 5000 kg are higher than those of payloads higher than 5000 kg.

Section 5

Predictive Analysis (Classification)

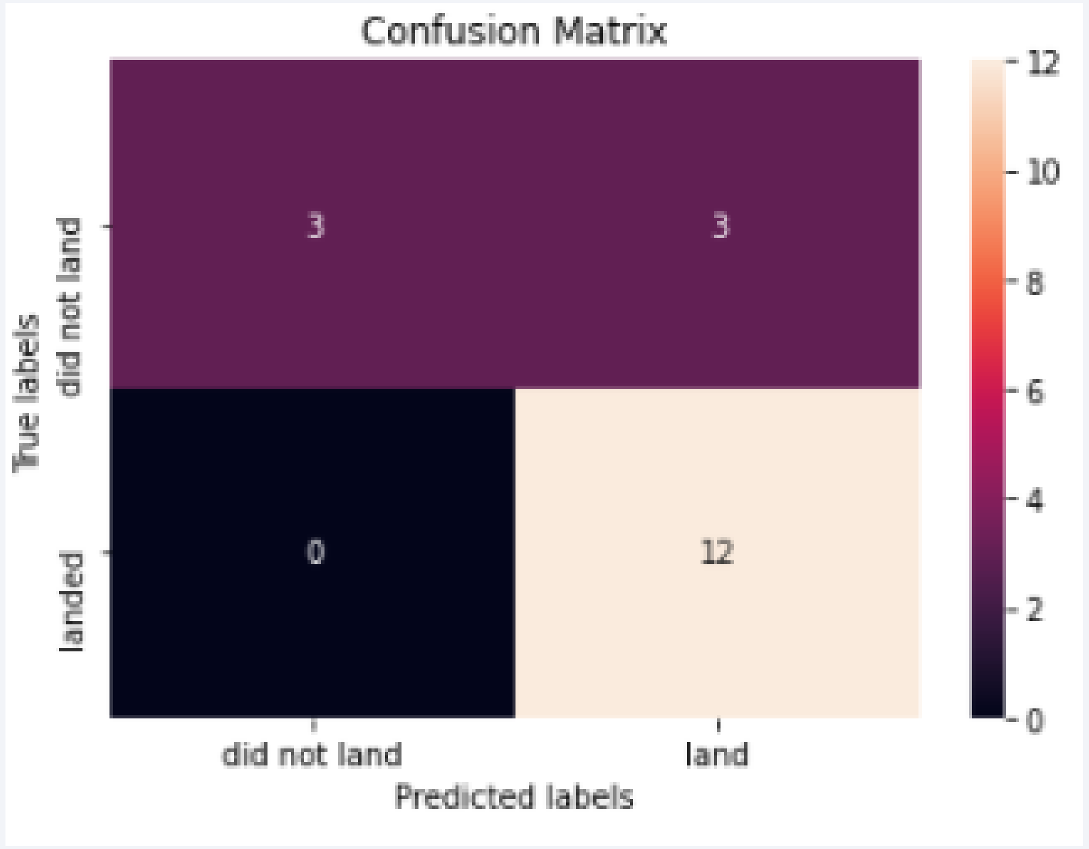
Classification Accuracy

- It's apparent from the bar chart obtained that the accuracy of all four models turned out to be equal at approximately 83 percent.



Confusion Matrix

- The confusion matrix turned out to be the same for all the models.
- The models predicted rightly predicted 12 successful landings and 3 failures.
- In addition there were 3 false positives.
- In conclusion, we can say that these models can predict successful landings.



Conclusions

- The success rate increases proportionally with the number of flights.
- The orbital types with the highest success rates are SSO, HEO, GEO, and ES-L1
- The launch sites are located in close proximity to railways, highways, and the coastlines, but are far from cities.
- The Kennedy Space Center Launch Complex 39a (KSC LC-39a) has the greatest number of successful launches and also the highest launch success rates.
- The launch success rates for Payloads lower than 5000 kg are higher than those of payloads higher than 5000 kg.
- In our project, all the models were found to have the same accuracy of around 83%. Perhaps more data is needed to determine the best model.

Appendix

1. “List of Falcon 9 and Falcon Heavy launches.” *Wikipedia*, 06 Feb. 2022, https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches
2. Sheetz, Michael. “Elon Musk touts low cost to insure SpaceX rockets as edge over competitors.” *CNBC*, 15 April 2020, https://www.cnbc.com/2020/04/16/elon-musk-spacex-falcon-9-rocket-over-a-million-dollars-less-to-insure.html
3. Github URL link for all files including the presentation: https://github.com/saad20201/Coursera-Saad/tree/master

Thank you!