IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

MARCELO MACHADO GOMES
17/07/2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection with SpaceX API and Web Scraping
  - Data Wrangling with Pandas
  - EDA with Data Visualization and SQL
  - Interactive Map Visualization with Folium
  - Dashboard Visualization with Plotly Dash
  - Predictive Analysis (Classification)
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

- Project background and context
- SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards 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. Instead of using rocket science to determine if the first stage will land successfully, you will train a machine learning model and use public information to predict if SpaceX will reuse the first stage.

- This project want to find answers to the following questions:
  - How to determine the price of each launch using a machine learning pipeline?
  - Will SpaceX will reuse the first stage?
  - Can we predict whether SpaceX will attempt to land a rocket or not?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data collected via Space X API and from Wikipedia via Web scrapping
- Perform data wrangling
  - Data was processed using Pandas python package
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Data was visualized with Matplotlib and Seaborn Packages
- Perform interactive visual analytics using Folium and Plotly Dash
  - Interactive Maps were built in Folium and an interactive panel was built using Plotly
- Perform predictive analysis using classification models
  - Data was processed with Pandas and tuned and evaluated with Scikit-learning packages

Data Collection

- The SpaceX data was collected using many different methods, such as:
  - Using the get request from the SpaceX API and decoding the response (returned as a Json data) from this request using .json() function and afterwards converting to a Pandas dataframe using .json_normalize()
  - Using pandas we cleaned and wrangled this data, formatting as a dataframe , removing and filling missing values
  - We got the Falcon 9 launch data using BeautifulSoup to web scrap the Wikipedia records, collecting the tables, parsing and converting to Pandas dataframes, as we did with the SpaceX data

Data Collection – SpaceX API

- After collecting data via SpaceX API we did some wrangling and formatting as can be seen aside

- GitHub URL for this notebook https://github.com/mgomes79/Applied-Data-Science-Capstone/blob/096e4f9417cc93d0c4111c5a7a76d9aac483a83e/Data%20Collection%20API%20Lab.ipynb

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

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

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

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

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

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 mult
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)]
```

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=data[data['BoosterVersion']!='Falcon 1']
data_falcon9.head()
```

Data Collection - Scraping

- To collect the Falcon 9 data we applied Web scrapping using BeautifulSoup package
- This data was parsed and converted to a Pandas dataframe
- Below is GitHub link to notebook
- https://github.com/mgomes79/Applied-Data-Science-Capstone/blob/773c5ecc2a324a39e657d1fa992d15181bd3af36/jupyter-labs-webscraping.ipynb

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.

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

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

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

```
#find a html table in the web page
html_tables = soup.find_all('table')
#for row in table.find_all('tr'): # in html table row is represented by the tag <tr>
#    print(row)
```

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

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

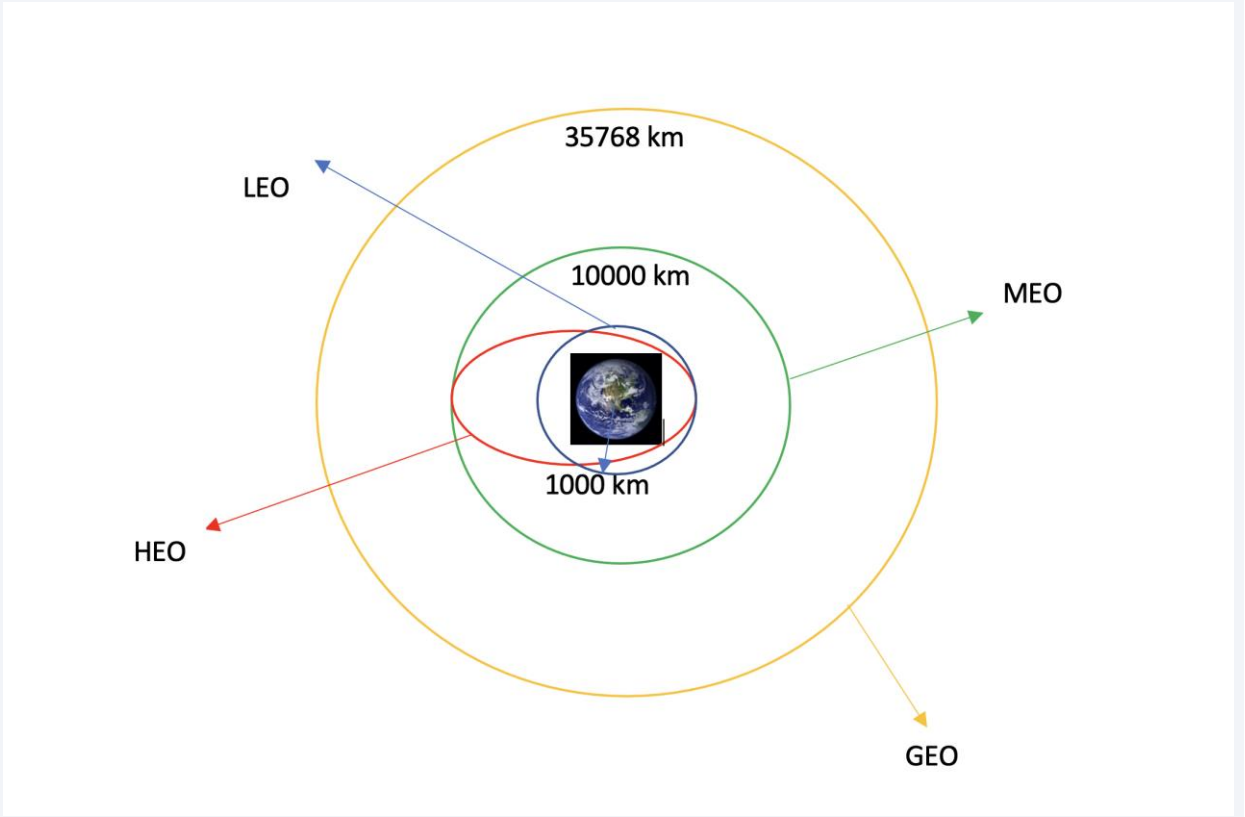
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table

for c in first_launch_table.find_all('th'):
    # Iterate each th element and apply the provided extract_column_from_header() to get a column name
    name = extract_column_from_header(c)
    # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
    if name is not None and len(name) > 0:
        column_names.append(name)
    #print("{}---->{}".format(color_name,color_code))
```

Data Wrangling

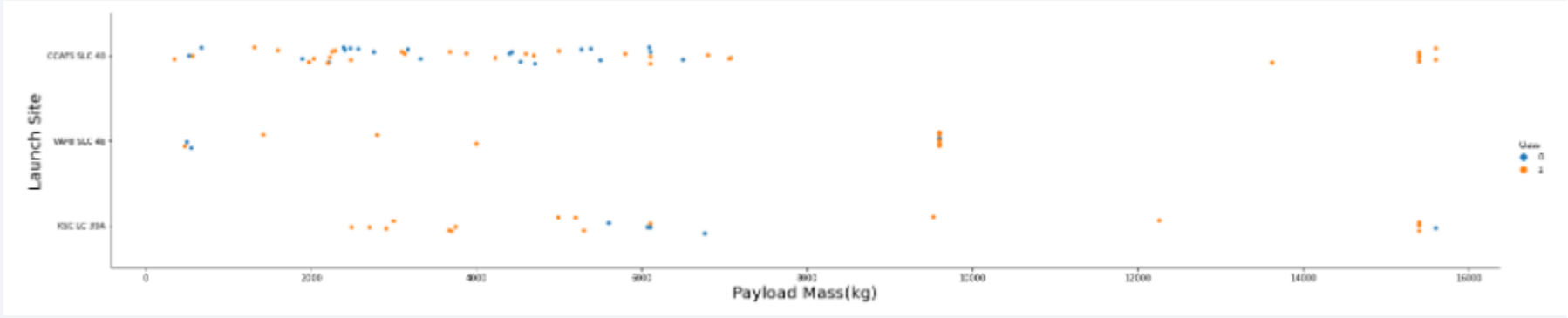
- We processed the data with pandas dataframe calculating:
  - The number of launches on each site
  - The number and occurrence of each orbit
  - The number and occurrence of mission outcome per orbit type, generating a bad_outcomes list
- After that, we created a landing outcome label from Outcome column

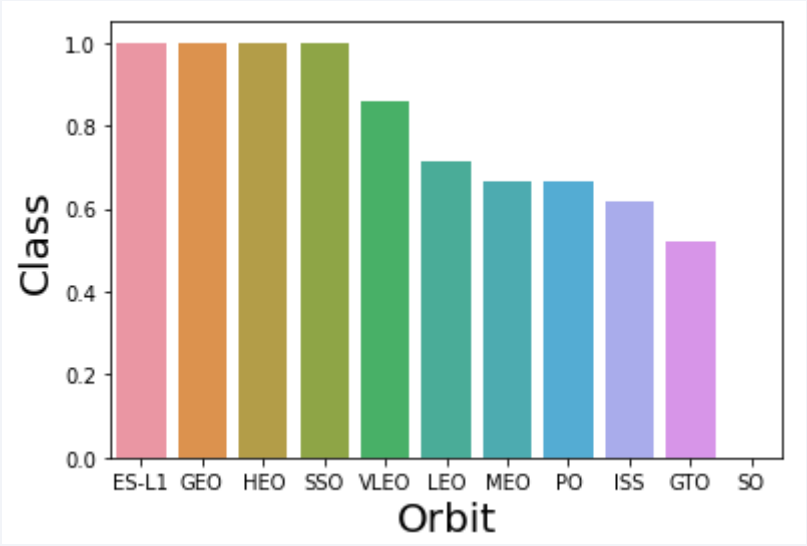


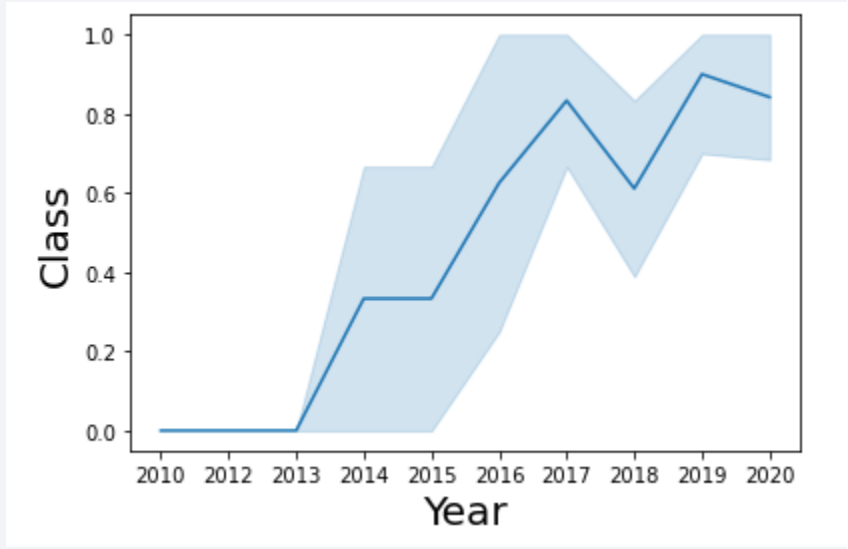
- Below is the GitHub link to the notebook
  - https://github.com/mgomes79/Applied-Data-Science-Capstone/blob/85dbded9bc202352901bbe46fffcca20b249f5fc/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- We explored the data using several plots available in the visualization Seaborn package such as:
  - scatter plot(to see the relationship between Flight Number x Launch Site, Payload x Launch Site, Flight Number x Orbit type, Payload x Orbit type)
  - bar chart (Success rate of each orbit type)
  - line plot (Visualize the launch success yearly trend)





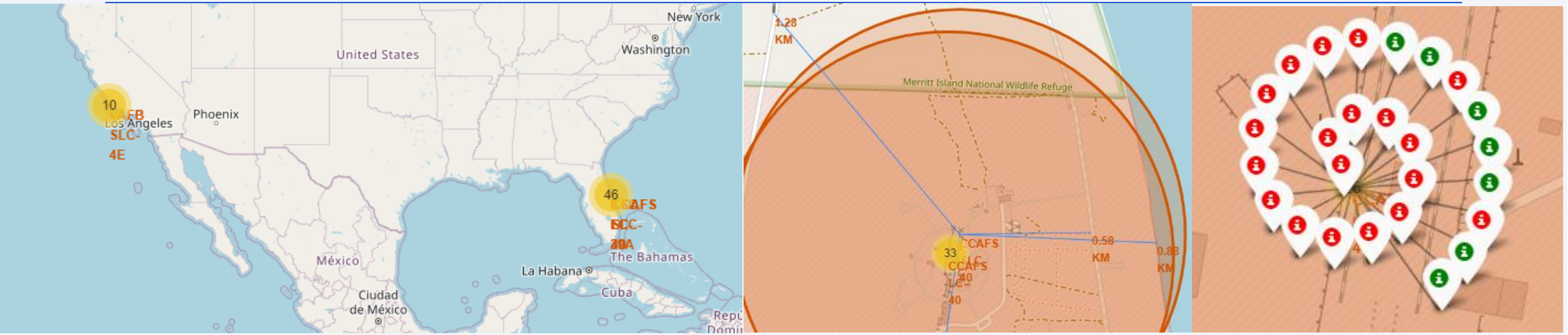


- The GitHub URL to the notebook :
  - https://github.com/mgomes79/Applied-Data-Science-Capstone/blob/319cf39496576cbf008c769a3b3bcedd51197ce6/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

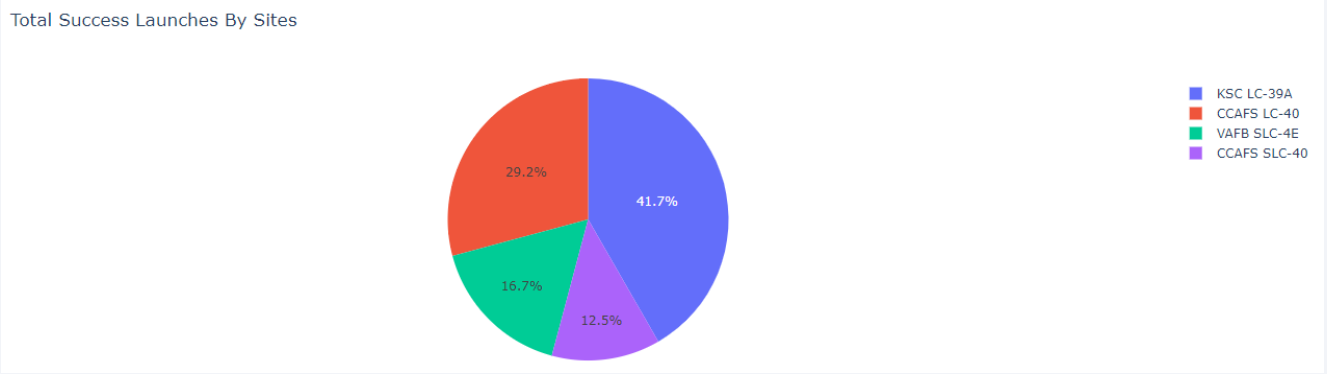
- Inside the Jupyter notebook we used built-in magic commands to make several queries in SQL to the available database
- We wrote queries to answer this questions:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The full code to the queries is available in the Appendix slides (47-50)
- Below is the GitHub link to the notebook:
  - https://github.com/mgomes79/Applied-Data-Science-Capstone/blob/16aea930e6648193918b239b6202f9d919626763/jupyter-labs-eda-sql-coursera_sqllite.ipynb

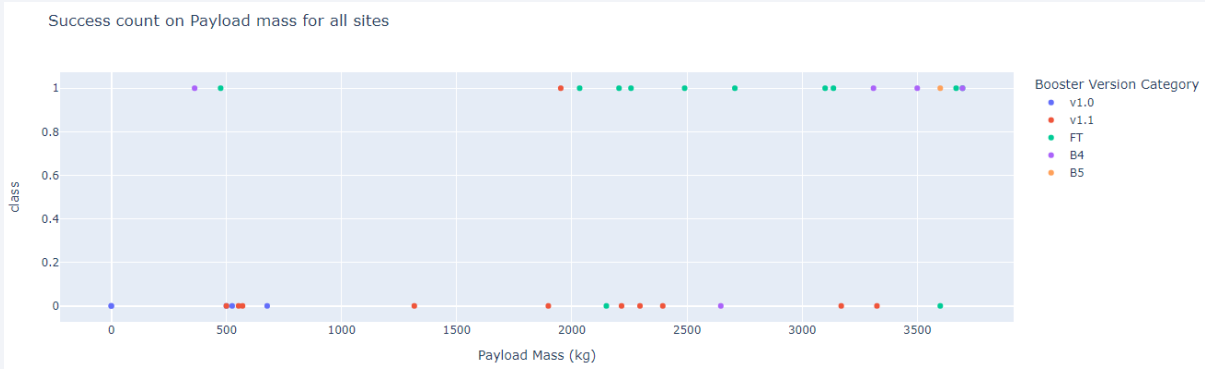
Build an Interactive Map with Folium

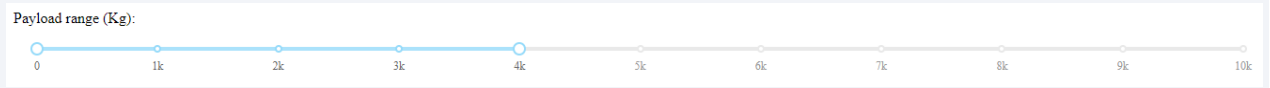


- We used Folium to create interactive maps with objects such as markers, circles, lines to visualize the launch sites and their proximities, as well to show the distances to points of interest
- GitHub URL of the notebook:
  - https://github.com/mgomes79/Applied-Data-Science-Capstone/blob/81687e0c78444389c1c56917a34396c63e7f2584/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash







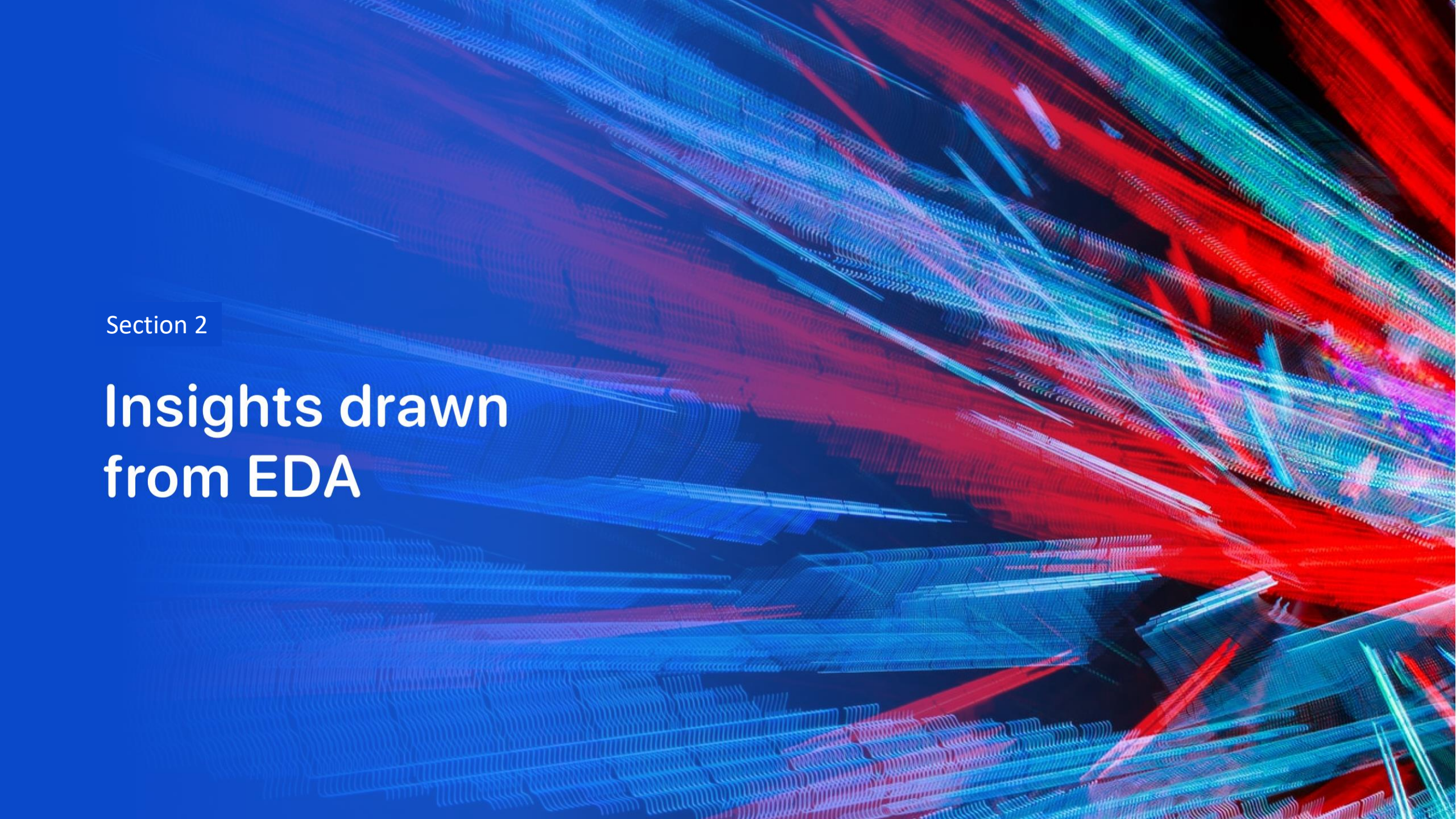
- Using Plotly Dash we constructed an interactive panel with selection of the Launch site done in dropdown box and the Payload range in a slider
- The GitHub URL to Python source file:
  - https://github.com/mgomes79/Applied-Data-Science-Capstone/blob/8ec320916a81472e3cc5a55f2bb4200cc5f62ae5/spacex_dash_app.py

Predictive Analysis (Classification)

- To make the Predictive Analysis, we used many Python resources, described as follows:
- We used Pandas and NumPy to the ETL(Extract, Load, Transform) process
- After we had a formatted Dataframe, we used scikit-learn to:
  - normalize the data (preprocessing.StandardScaler() function),
  - split the data as training and testing datasets (train_test_split() function)
  - For each model in a set of different machine learning models (logistic regression, support vector machine, decision tree , k nearest neighbors) we prepared a set of hyperparameters and used GridSearchCV in training data to search for the best parameters
  - With the best parameters in hands, we ran the models using the test dataset and compared the results
- To better understanding of results, we visualized them using Matplotlib and Seaborn packages
- The GitHub URL to the notebook
  - https://github.com/mgomes79/Applied-Data-Science-Capstone/blob/7044611022cc4374e30c671d45b462ab15cd0ed1/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

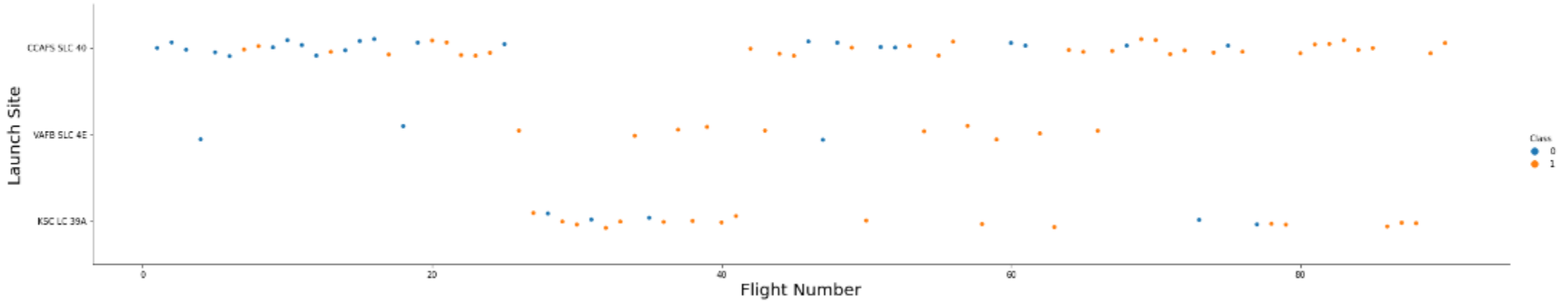
Results

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


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



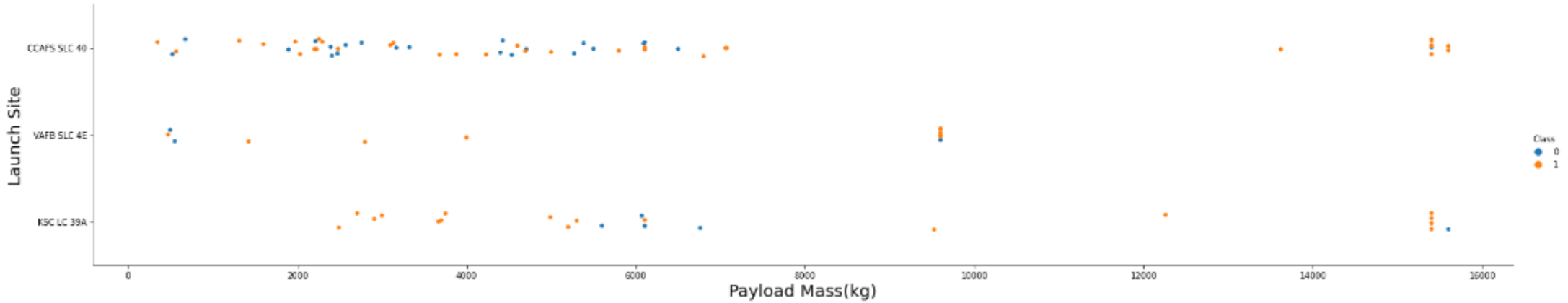
Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

We can see in the 3 launch sites the same pattern, that larger the Flight Number, the greater the success rate

- As is written in the screenshot, We can see in the 3 launch sites the same pattern, that larger the Flight Number, the greater the success rate

Payload vs. Launch Site

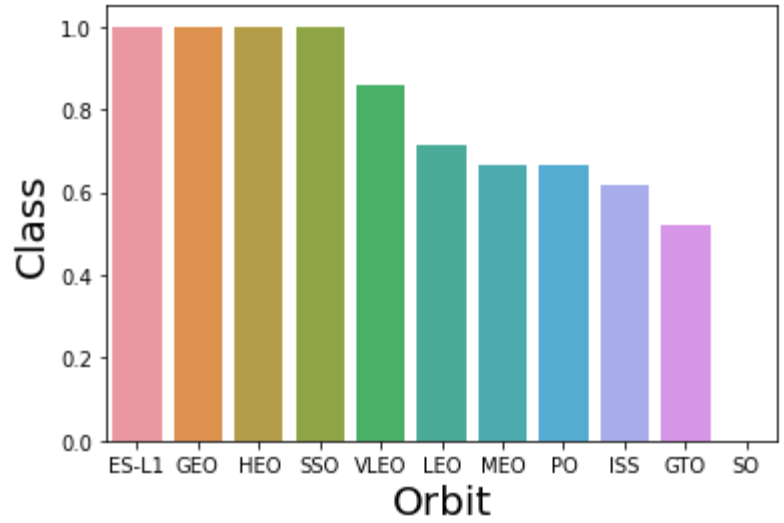
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload Mass(kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



- Show a scatter plot of Payload vs. Launch Site
- We can see the same pattern in the 3 sites, that the success rate is greater when the payload is near its respective maximum payload.

Success Rate vs. Orbit Type

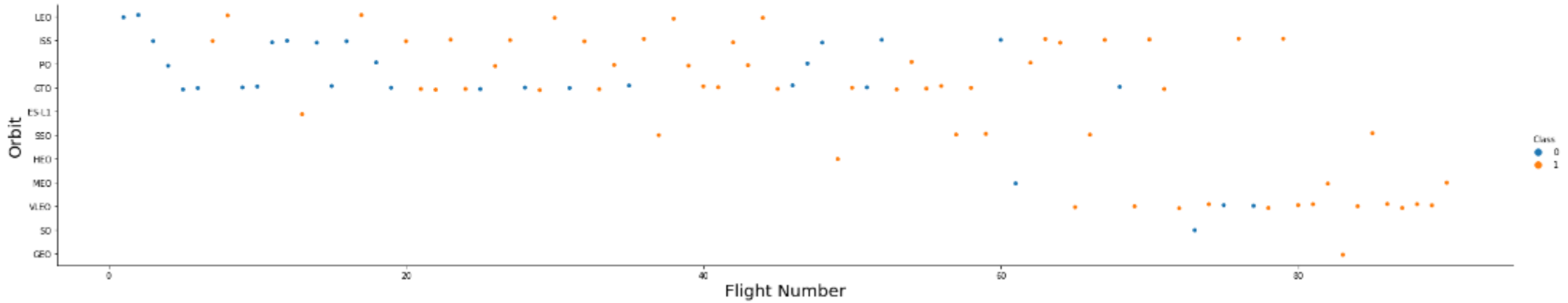
```
# HINT use groupby method on Orbit column and get the mean of Class column
sns.barplot(y="Class", x="Orbit", data=df.groupby("Orbit").mean().reset_index().sort_values(by='Class', ascending=False))
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("Class",fontsize=20)
plt.show()
```



- Below is top 5 orbit types ordered by the success rate
  - ES-L1 100%
  - GEO 100%
  - HEO 100%
  - SSO 100%
  - VLEO 86%

Flight Number vs. Orbit Type

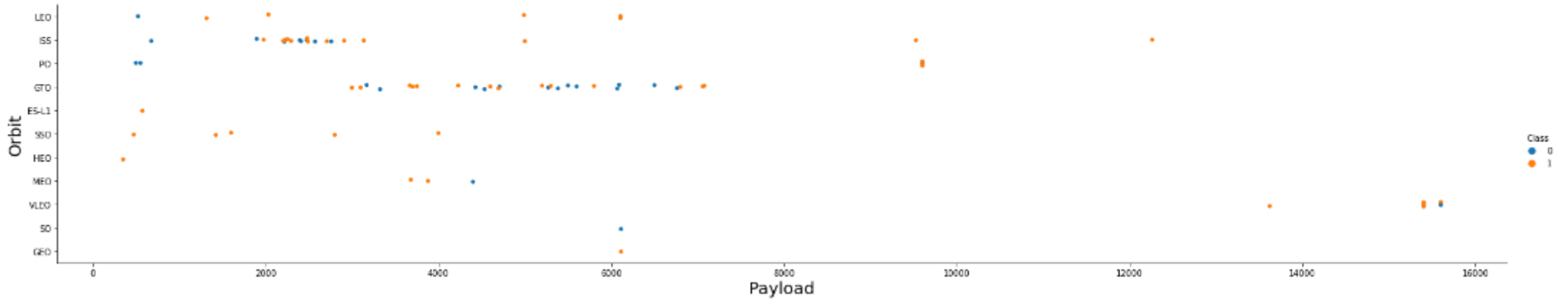
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



- We can see that 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

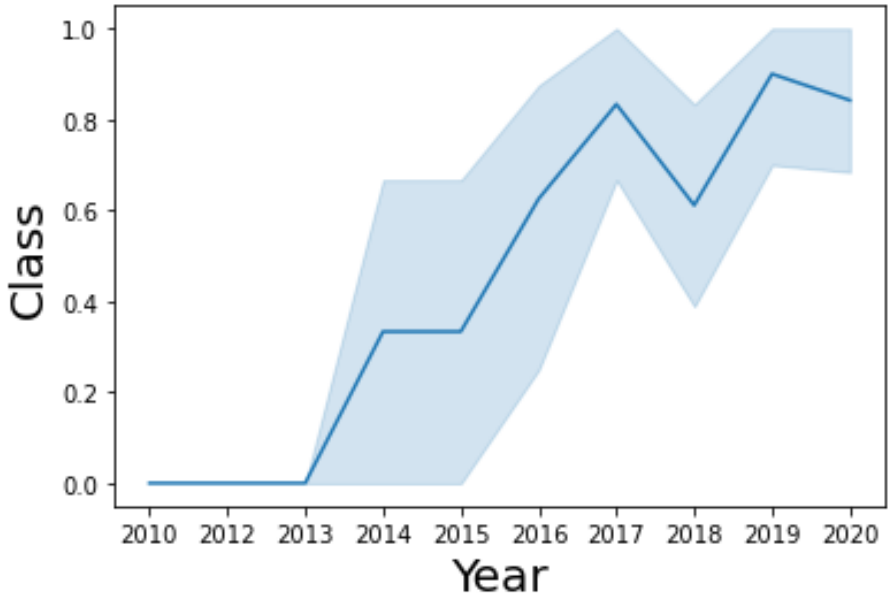
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



- We can see in the scatter plot above that with heavy payloads the percentage of successful landing is greater for Polar, LEO and ISS orbits.
- However, for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
sns.lineplot(y="Class", x="Year", data=df_t6) #.groupby('Year').mean().reset_index() )
plt.xlabel("Year",fontsize=20)
plt.ylabel("Class",fontsize=20)
plt.show()
```



- We see that the success rate since 2013 kept increasing till 2020, with exception of a fall in 2018.

All Launch Site Names

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

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

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

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

- The names of the unique launch sites
  - CCAFS LC-40
  - VAFB SLC-4E
  - KSC LC-39A
  - CCAFS SLC-40
- The query selects the distinct values in LAUNCH_SITE column

Launch Site Names Begin with 'CCA'

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

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

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

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

- SQL query from SPACEXTBL for rows with column LAUCH_SITE starting with ‘CCA

Total Payload Mass

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

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

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

SUM(PAYLOAD_MASS__KG_)
45596

- The total payload carried by boosters from NASA is 45596 KG
- The query sums the values in column PAYLOAD_MASS__KG_ for all rows that column CUSTOMER has value 'NASA (CRS)'

Average Payload Mass by F9 v1.1

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

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

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

AVG(PAYLOAD_MASS__KG_)
2928.4

- The average payload mass carried by booster version F9 v1.1 is 2928.4 KG
- The query calculate the average in PAYLOAD_MASS__KG_ column) in rows where BOOSTER_VERSION = 'F9 v1.1'

First Successful Ground Landing Date

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

Hint:Use min function

```
%%sql
SELECT min(substr(DATE,7,4)||'-'||substr(DATE,4,2)||'-'||substr(DATE,1,2))  as Date
FROM SPACEXTBL
 WHERE "Landing _Outcome" = 'Success (ground pad)';
```

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

Date
2015-12-22

- The dates of the first successful landing outcome on ground pad is 12/22/2015
- The query formats the Date string field so that min function works as expected, and select the minimum value in Date column in rows where the "Landing _Outcome" is 'Success (ground pad)';

Successful Drone Ship Landing with Payload between 4000 and 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_ < 6000
and PAYLOAD_MASS__KG_ > 4000 ;
```

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

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

- The names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
  - F9 FT B1022
  - F9 FT B1026
  - F9 FT B1021.2
  - F9 FT B1031.2
- The query selects the Booster_Version in rows where "Landing _Outcome" = 'Success (drone ship)' and PAYLOAD_MASS__KG_ is between 4000 and 6000

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%%sql
  SELECT trim(MISSION_OUTCOME) as MISSION_OUTCOME, count(*) as count
 FROM SPACEXTBL
  group by trim(MISSION_OUTCOME) ;
```

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

MISSION_OUTCOME	count
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

- Total number of successful and failure mission outcomes
  - Failure (in flight) 1
  - Success 99
  - Success (payload status unclear) 1
- The query selects the aggregation of values of MISSION_OUTCOME column and count how many row with this value. We use the trim function to remove space so that the group by works as expected

PAYLOAD_MASS__KG_

Boosters Carried Maximum Payload

- The names of the booster which have carried the maximum payload mass
  - 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
- The subquery selects the max value in PAYLOAD_MASS__KG_. The main query select the Booster_Version in rows where PAYLOAD_MASS__KG is equal with the subquery result

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)
;
```

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

2015 Launch Records

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 , "Landing _Outcome", Booster_Version, Launch_Site
     FROM SPACEXTBL
 where   substr(Date,7,4)='2015'
  and "Landing _Outcome" = 'Failure (drone ship)' ;
```

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

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

- The failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  - Failure (drone ship) F9 v1.1 B1012 CCAFS LC-40
  - Failure (drone ship) F9 v1.1 B1015 CCAFS LC-40
- The selects the substring related to Month in Date column and "Landing _Outcome", Booster_Version, Launch_Site columns in rows with the substring related to Year in Date column equal to '2015'

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

- 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
  - No attempt 10
  - Success (drone ship) 5
  - Failure (drone ship) 5
  - Success (ground pad) 3
  - Controlled (ocean) 3
  - Uncontrolled (ocean) 2
  - Precluded (drone ship) 1
  - Failure (parachute) 1
- The query group by values in "Landing _Outcome“ column and count how many rows filtered by the 2010-06-04 and 2017-03-20 date range

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

```
%%sql

select "Landing _Outcome",count(*) as landing_outcomes
    FROM SPACEXTBL
 where   substr(DATE,7,4)||'-'||substr(DATE,4,2)||'-'||substr(DATE,1,2) > '2010-06-04'
 and
substr(DATE,7,4)||'-'||substr(DATE,4,2)||'-'||substr(DATE,1,2)  < '2017-03-20'
  group by "Landing _Outcome"
  order by count(*) DESC ;
```

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

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

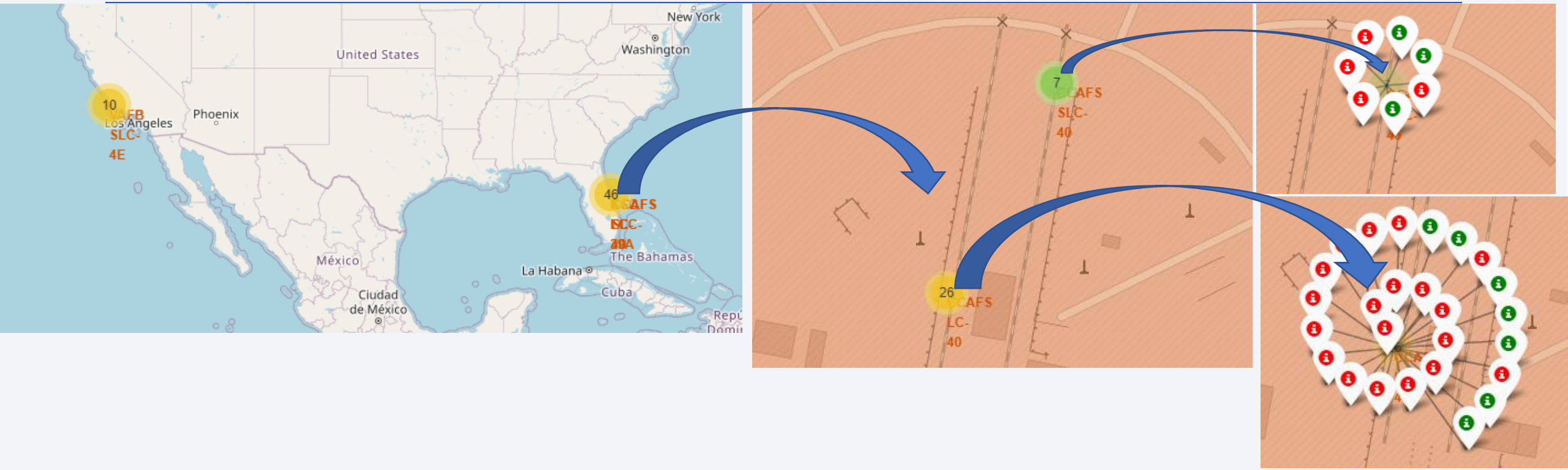
Section 3

Launch Sites Proximities Analysis

All SpaceX launch sites on the Global map

- Are all launch sites in proximity to the Equator line?
- Are all launch sites in very close proximity to the coast?
- Yes to both questions. It makes sense to be near the Equator line since it will consume less fuel to get into the space. And the coast is a natural choice, for safety reasons.

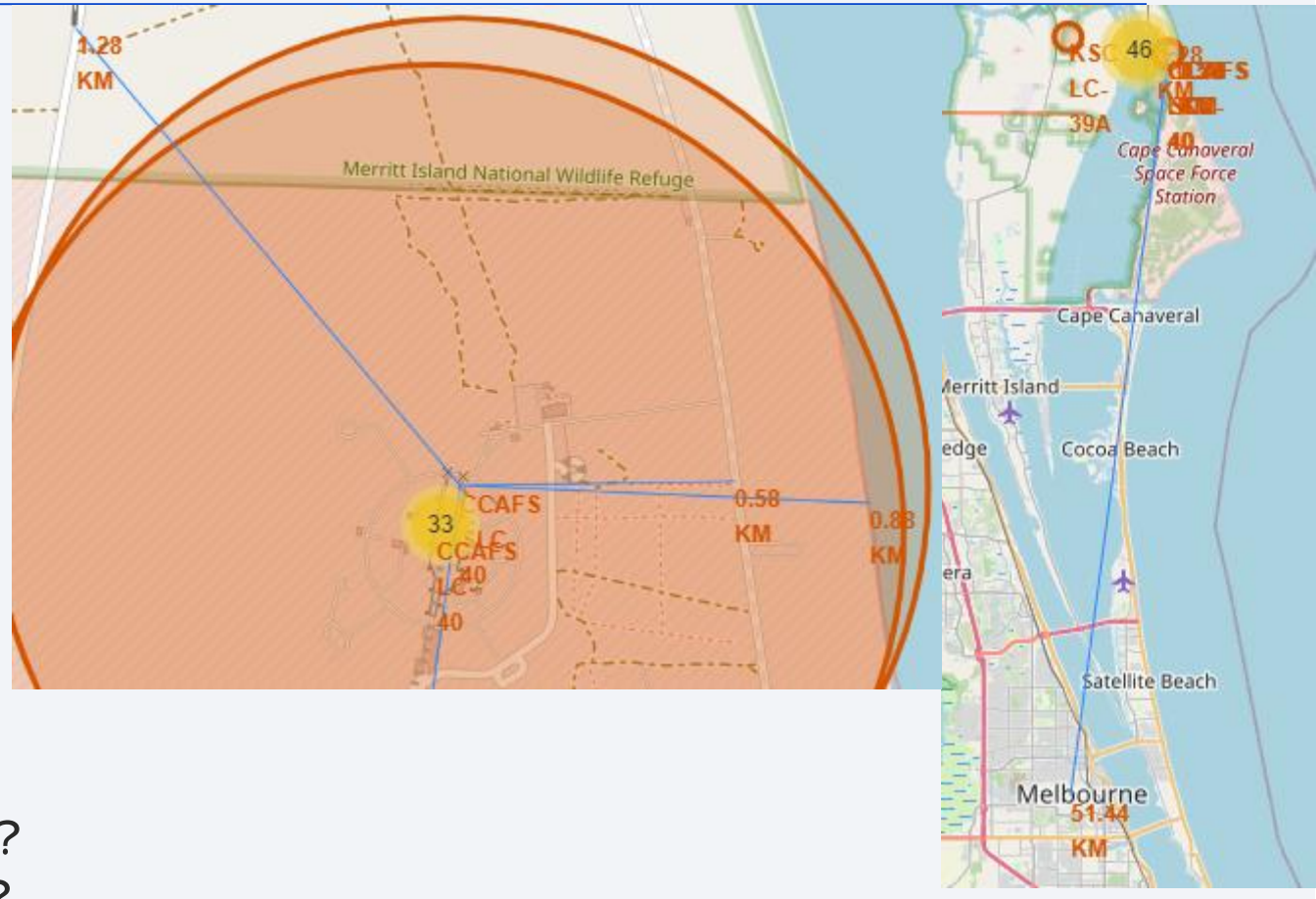
Mark the success/failed launches for each site on the map



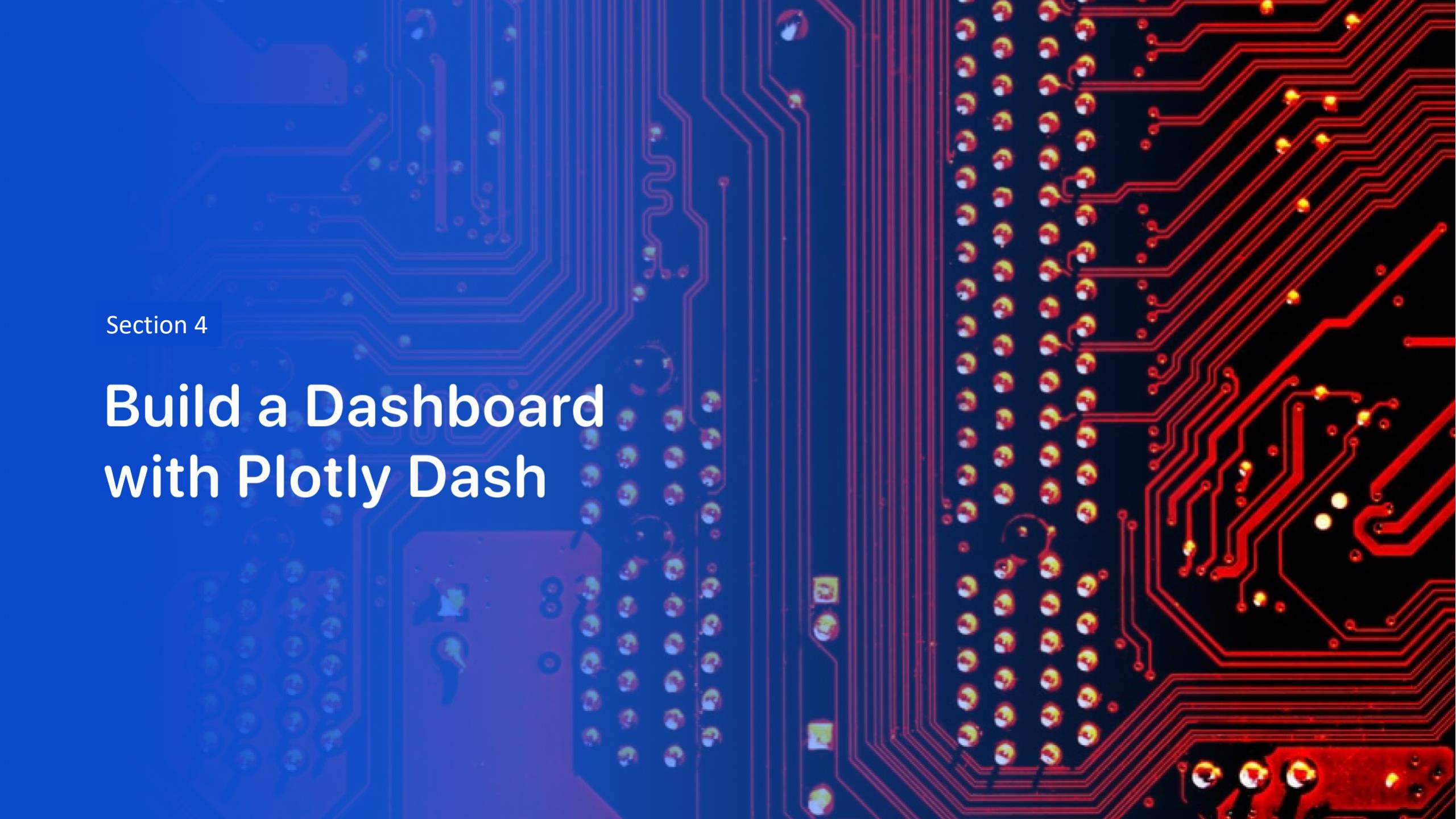
- We can see in the screenshot above the zooming in the launch areas
- The Green Markers are Successful Launches and Red Markers are failures

Calculate the distances between a launch site to its proximities¶

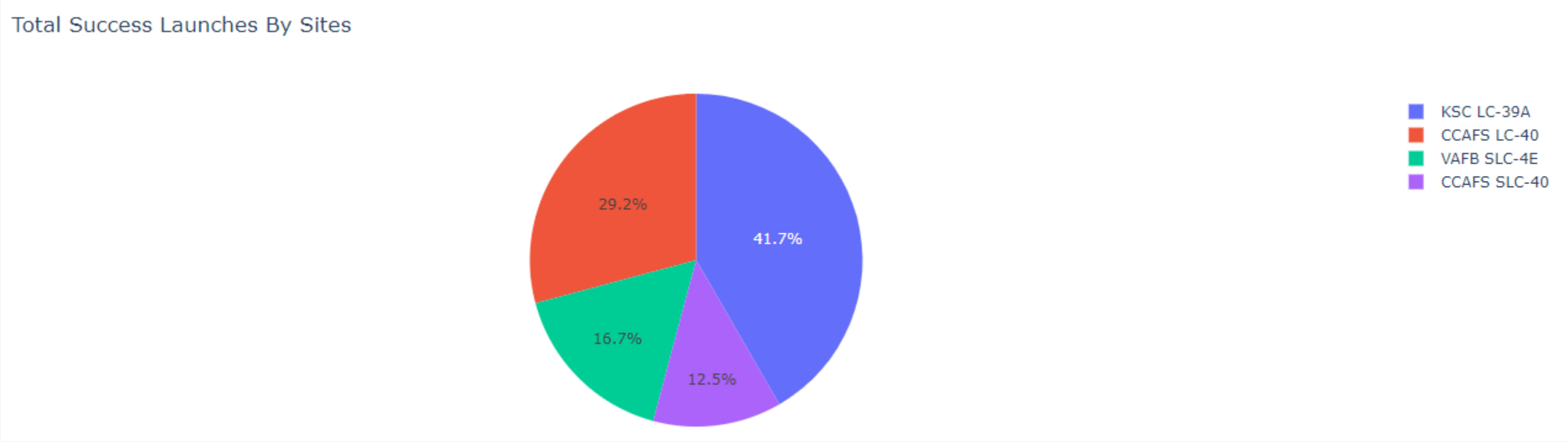
- We used this generated folium map to Explore the CCAFS SLC-40 launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed
  - distance_highway = 0.5843041171932833 km
  - distance_railroad = 1.278535752724802 km
  - distance_coastline = 0.8781395250290789 km
  - distance_city = 51.43819216416339 km



- We have the following questions:
  - Are launch sites in close proximity to railways?
  - Are launch sites in close proximity to highways?
  - Are launch sites in close proximity to coastline?
  - Do launch sites keep certain distance away from cities?
- Yes to all questions. It's close to railways, highways and coastline for logistic reasons. It's near the coastline and far from cities for security reasons.

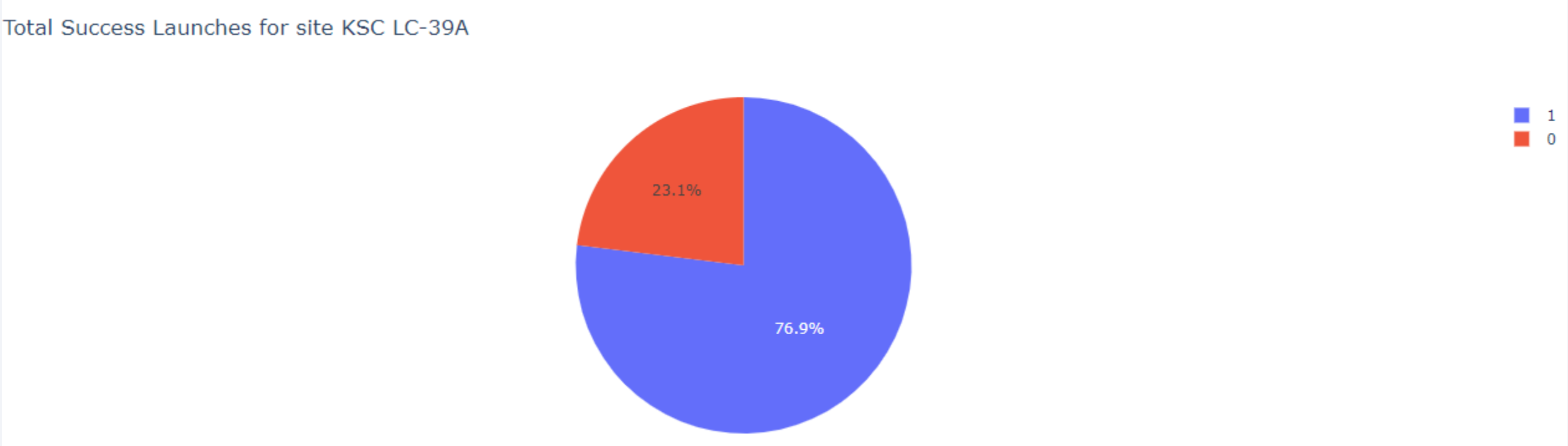

Section 4
Build a Dashboard with Plotly Dash

Pie chart with launch success for all sites



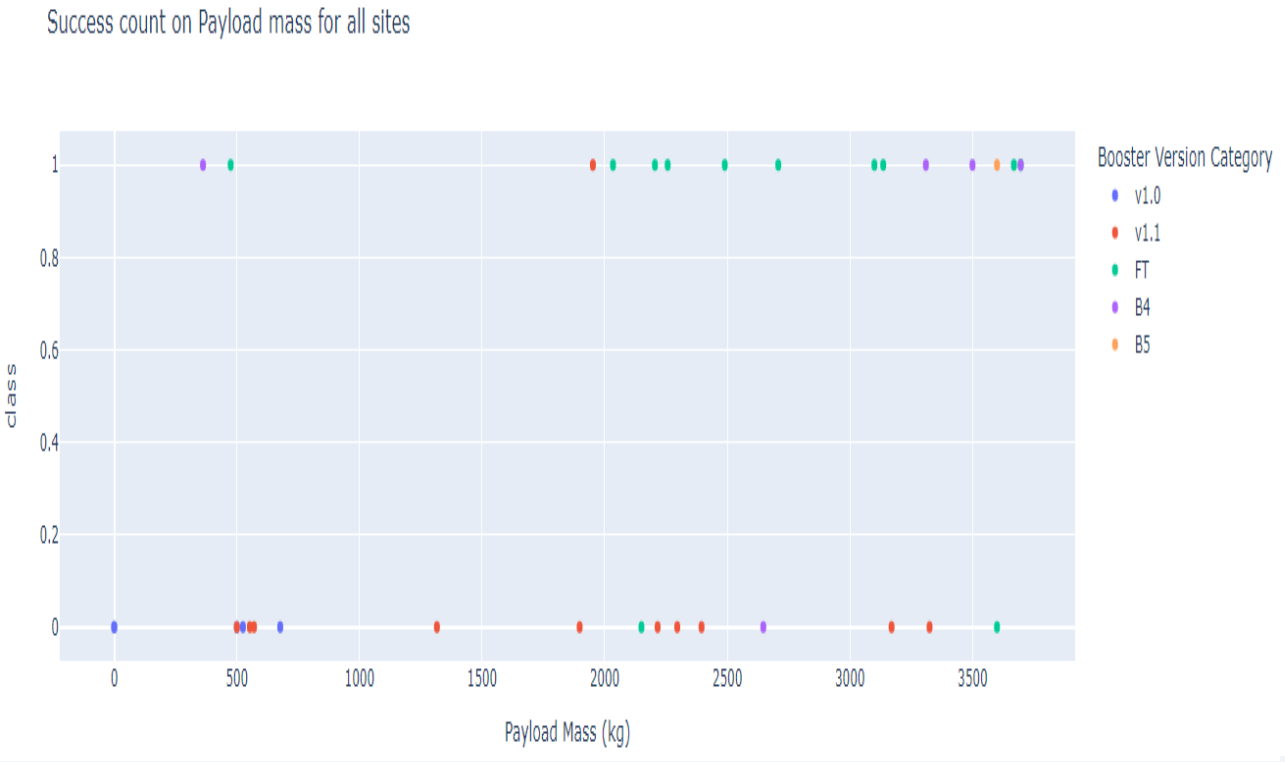
- We can see that KSC LC-39A is the most successful site, with 41.7% of total success launches

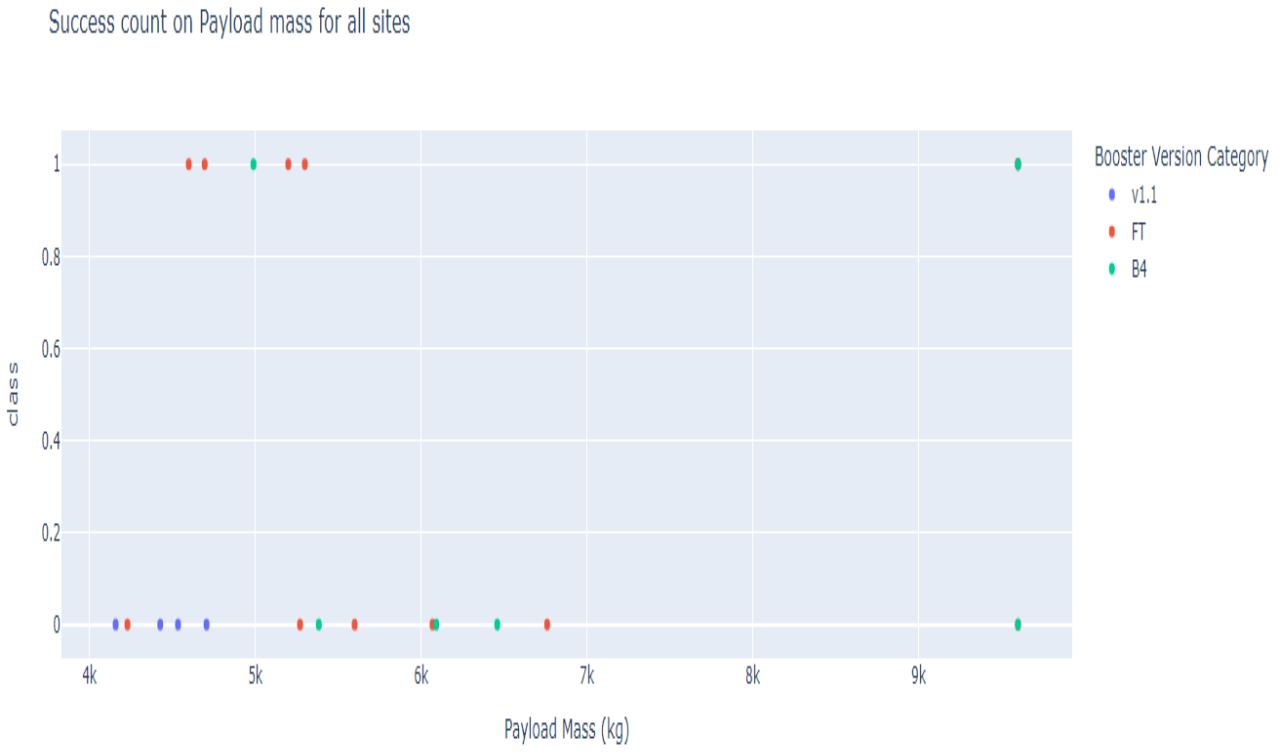
Comparing results in launches the most successful site (KSC LC-39A)



- We can see that 76.9% of KSC LC-39A launches were successful

Scatter plot Success count x Payload Mass (by Booster Version)



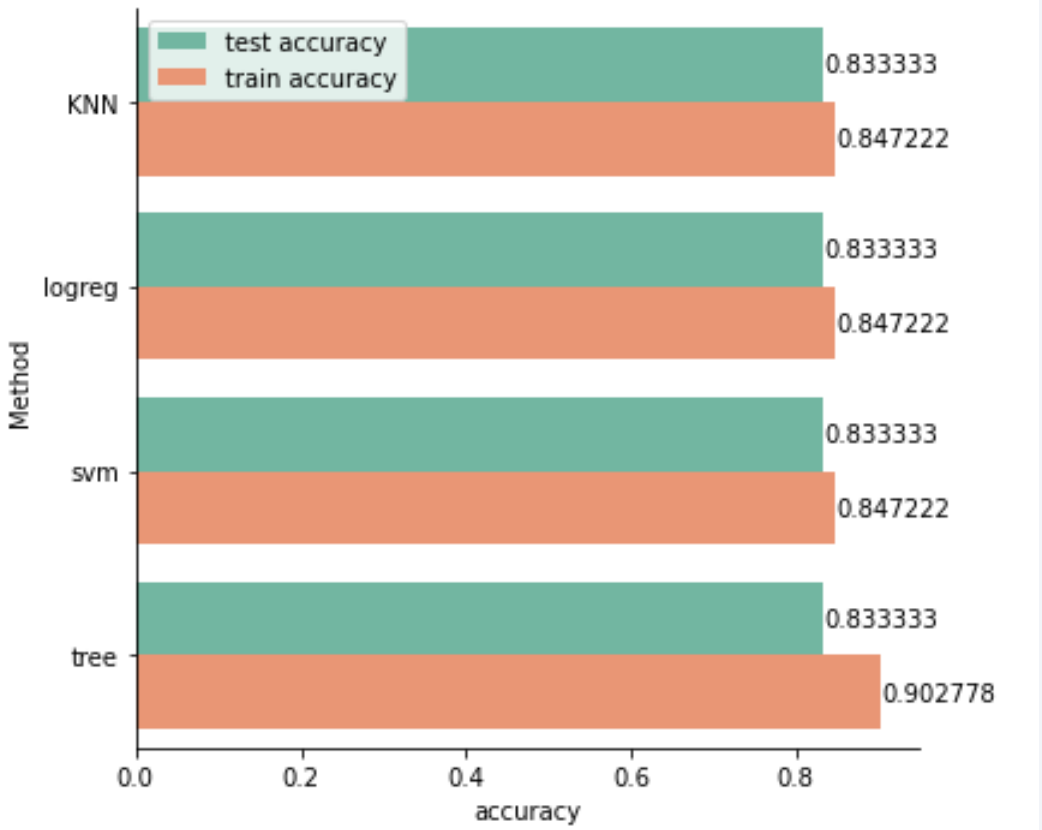


- We divided the Payload in 2 ranges (0 to 4,000kg and 4,000kg to 10,000kg) using the range slider to see the success launches for each Booster version (differenced by color)
- We can see that in the first range we have a better success rate

Section 5

Predictive Analysis (Classification)

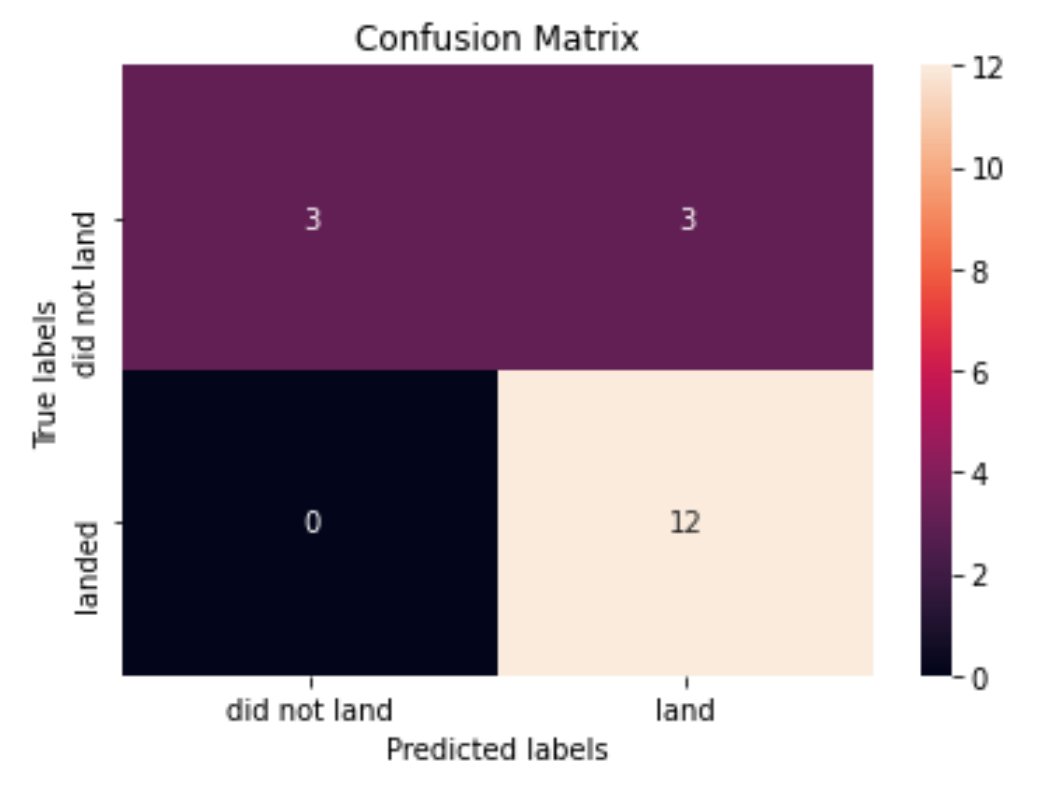
Classification Accuracy



	Method	best_params	train accuracy	test accuracy
0	logreg	{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}	0.847222	0.833333
1	svm	{'C': 1.0, 'gamma': 0.03162277660168379, 'kern...	0.847222	0.833333
2	tree	{'criterion': 'entropy', 'max_depth': 4, 'max_...	0.902778	0.833333
3	KNN	{'algorithm': 'auto', 'n_neighbors': 9, 'p': 1}	0.847222	0.833333

- We can see that logreg, svm tree and KNN have similar results in training dataset and the same result in test dataset, so their use is equivalent.

Confusion Matrix



- Here is the Logistic Regression confusion matrix. It performs well but there is 3 false positives (upper right box).

Conclusions

- From all this presented exploration we can conclude some findings:
- 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 (although a declining in 2018).
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A is responsible for most of the successful launches among all sites and the one with higher successful launch proportion.
- For the predictive analysis we found that Logistic Regression, SVM, Decision Tree and KNN have similar results, both in training and test datasets, so their use is equivalent.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Appendix - EDA with SQL

- Below and in the next 3 slides we present the queries for the respective tasks
- TASK 1
  SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL
- TASK 2
  SELECT * FROM SPACEXTBL
  WHERE LAUNCH_SITE LIKE 'CCA%'
  LIMIT 5;
- TASK 3
  SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL
  WHERE CUSTOMER = 'NASA (CRS)';
- TASK 4
  SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL
  WHERE BOOSTER_VERSION = 'F9 v1.1';

Appendix - EDA with SQL

- TASK 5
  SELECT min(substr(DATE,7,4)||'-'||substr(DATE,4,2)||'-'||substr(DATE,1,2))
  as Date
  FROM SPACEXTBL
  WHERE "Landing _Outcome" = 'Success (ground pad)';
- TASK 6
  SELECT Booster_Version FROM SPACEXTBL
  WHERE "Landing _Outcome" = 'Success (drone ship)'
  and PAYLOAD_MASS__KG_ < 6000
  and PAYLOAD_MASS__KG_ > 4000 ;
- TASK 7
  SELECT trim(MISSION_OUTCOME), count(*)
  FROM SPACEXTBL
  group by trim(MISSION_OUTCOME) ;

Appendix - EDA with SQL

- TASK 8

```
select Booster_Version FROM SPACEXTBL
where    PAYLOAD_MASS__KG_ =
 (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

- TASK 9

```
select
substr(Date, 4, 2) as Month , "Landing _Outcome", Booster_Version,
Launch_Site       FROM SPACEXTBL
 where    substr(Date,7,4)='2015'
  and "Landing _Outcome" = 'Failure (drone ship)' ;
```

Appendix - EDA with SQL

- TASK 10 – Two versions, the first one filtering the successful_landing_outcomes (as required in the notebook description) and the second one without the filter (as required in the Results section)

```
1) select "Landing _Outcome",count(*) as successful_landing_outcomes  FROM SPACEXTBL
where
substr(DATE,7,4)||'-'||substr(DATE,4,2)||'-'||substr(DATE,1,2) > '2010-06-04'
 and substr(DATE,7,4)||'-'||substr(DATE,4,2)||'-'||substr(DATE,1,2)  < '2017-03-20'
   and   "Landing _Outcome" LIKE 'Success%'
group by "Landing _Outcome"    order by count(*) DESC ;
2) select "Landing _Outcome",count(*) as successful_landing_outcomes  FROM SPACEXTBL
where
substr(DATE,7,4)||'-'||substr(DATE,4,2)||'-'||substr(DATE,1,2) > '2010-06-04'
 and substr(DATE,7,4)||'-'||substr(DATE,4,2)||'-'||substr(DATE,1,2)  < '2017-03-20'
   and   "Landing _Outcome" LIKE 'Success%'
```

- Below is the GitHub link to the notebook:
  - https://github.com/mgomes79/Applied-Data-Science-Capstone/blob/16aea930e6648193918b239b6202f9d919626763/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Thank you!