IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Tiago de Araujo
10.03.2023

Outline

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

Executive Summary

- Summary of methodologies
  - Import data (API, Wikipedia)
  - Manipulate data (cleaning, wrangling)
  - Analyze data (SQL, Dash,Folium)
  - Predict data (ML)

- Summary of all results
  - Interactive Dashboard
  - Exploratory Analysis
  - Prediction via ML models

Introduction

- Project background and context
  - SpaceX wants to revolutionize space expeditions. They try to achieve that, by reusing rockets after the launch. In theory with this way it is possible to cut 100 million dollar from the total cost.
    We want to get a better look at this project, crunch some numbers and find out how viable this approach is and will be.
- Problems you want to find answers
  - Predict successful return of rockets
  - Understand variables that impact that result

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Split in two parts:
    - SpaceX API: SpaceX provides data about all their launches. Request data from API.
    - Wikipedia: Holds additional information about Falcon9. Web scrap website.
- Perform data wrangling
  - Standard cleaning of data (NaN, …) and one-hot encoding for prediction
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Sklearn package: We build KNN, SVM, DT and LR and compare accuracy

Data Collection

- SpaceX API
  - We send a get request to the SpaceX API.
  - In return we get a json file containing all the relevant data.
  - We transform data to useful fromat
- Web Scraping
  - Web scraping is done by getting a http response from the website (Wikipedia).
  - We then parse that response and transform it to a useful format.

https://github.com/linushen/coursera_final_cap/blob/main/Collection_API.ipynb

Data Collection – SpaceX API

- Request API: send request to API
- Normalize: transform data to json format
- Restrict data: choose parts of data that we want to analyze

```python
# request the SpaceX launch data
res = requests.get(static_json_url)
print(len(res.content))
```

Python

```
269589
```

```python
# Use json_normalize meethod to convert the json result into a dataframe
static_json_df = res.json()

data = pd.json_normalize(static_json_df)
```

Python

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

# We will remove rows with multiple cores because those are falcon rockets with 2 ex
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
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 th
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)]
```

Python

Data Collection - Scraping

- Request: Send request to website. Returns https response
- Parse: Parse object with beautiful soup
- Filter: Filter column names
- Iterate and create DataFrame

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

Python

```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text)
```

Python

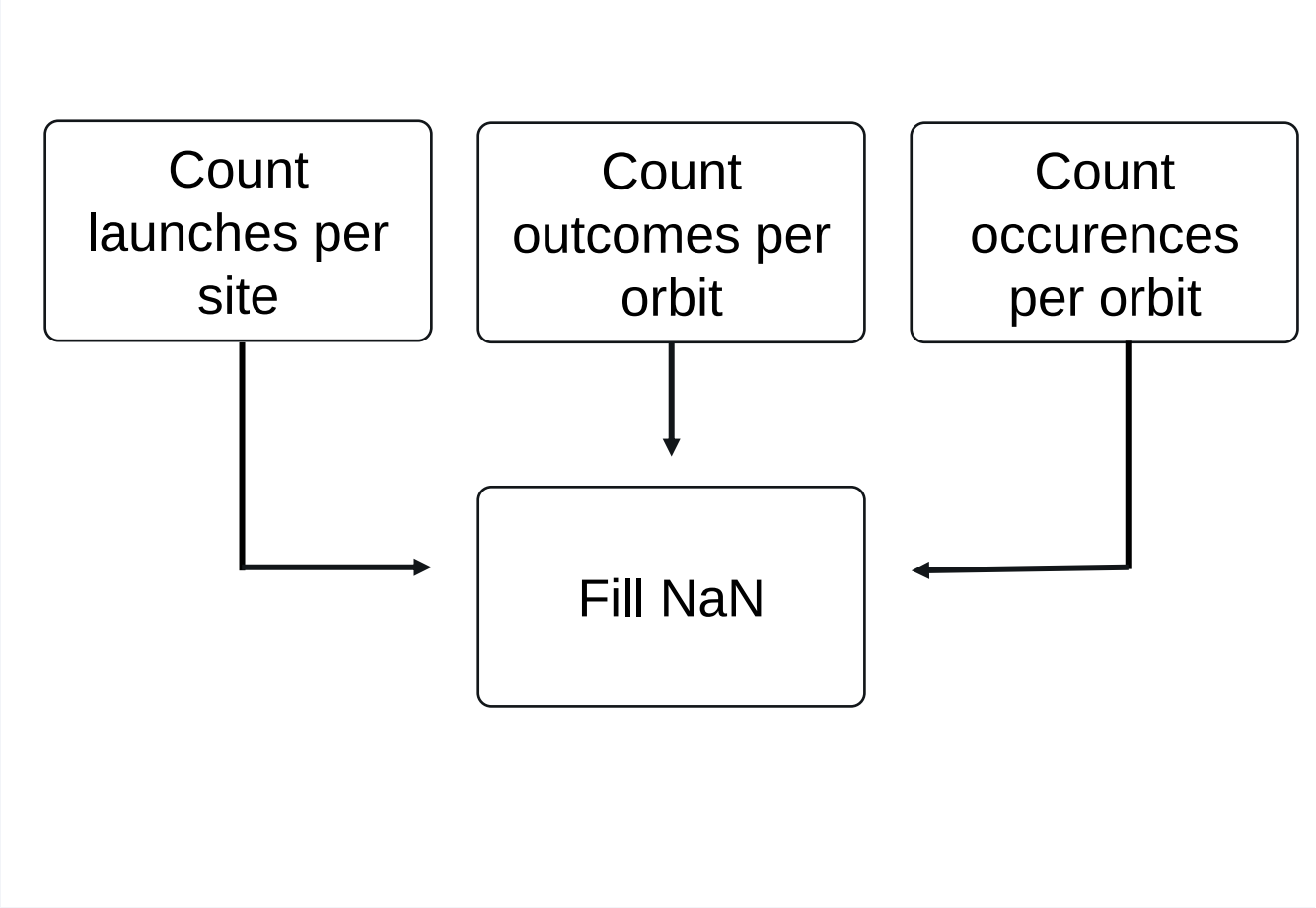
```python
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a

element_list = soup.find_all('th')
for element in element_list:
    try: #extract_column_from_header cant handle empty content
        name = extract_column_from_header(element)
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

Python

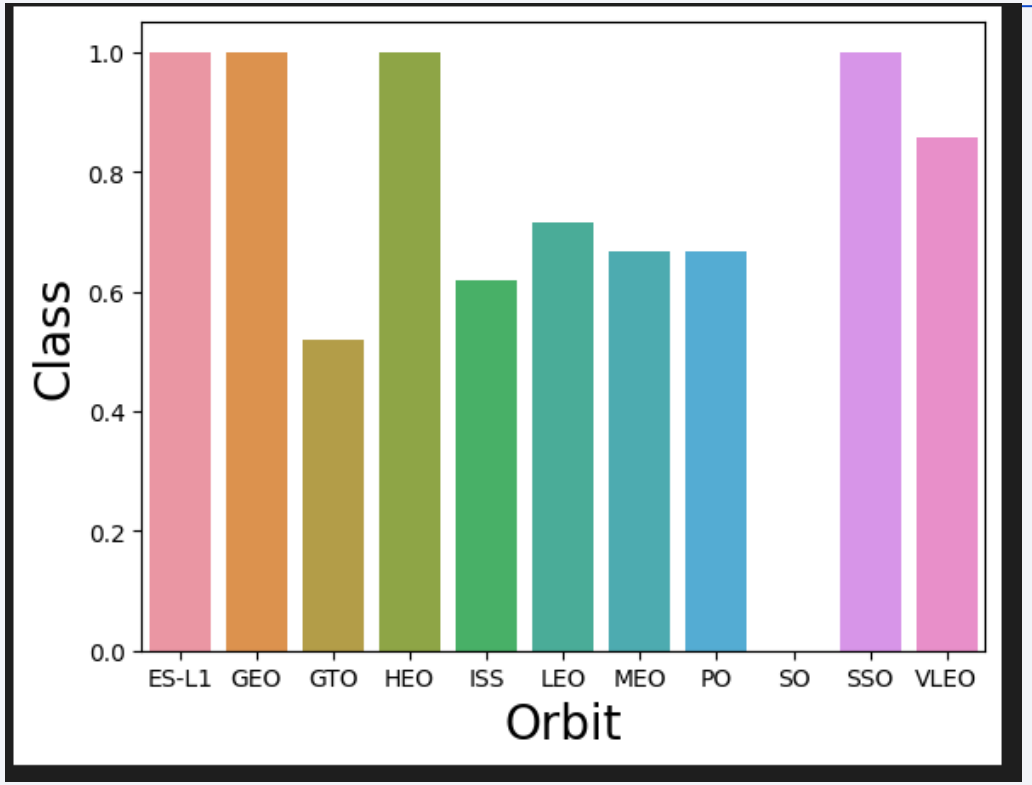
```python
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders
    # get table row
     for rows in table.find_all("tr"):
```

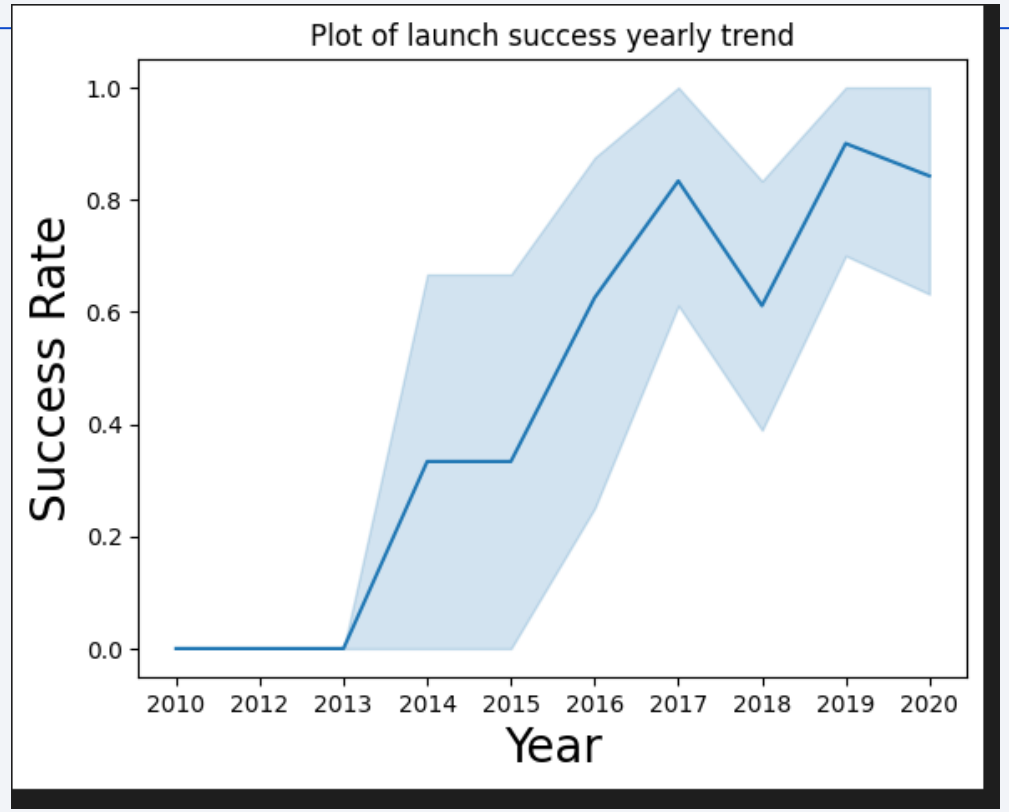
Data Wrangling

- We created extra features (count launches, …)
- We filled all null values
- We also created an extra label for outcome



EDA with Data Visualization





- We created plots that help us understand the relationship between e.g. success rate and orbit type or to see trends in the data
- https://github.com/linushen/coursera_final_cap/blob/main/EDA_Visual_SNS.ipynb

EDA with SQL

- SQL queries included:
  - List of unique launch sites
  - Total payload mass carried by booster
  - Find booster with success in drone ship
  - Total number of successful mission outcomes
- https://github.com/linushen/coursera_final_cap/blob/main/EDA_SQL.ipynb

Build an Interactive Map with Folium

- We created a folium map and added markers. This include:
  - Circles (To highlight location of launch site)
  - Marker (To show name of launch site)
  - Marker cluster (To show amount of successful missions for each launch site)
  - Lines (To show the closest coastline)
- https://github.com/linushen/coursera_final_cap/blob/main/Folium.ipynb

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

- Load data: load data into dataframe
- Scale data: scale data for better performance
- Split data: split into train and test set
- GridSearch: perform gridsearch for each model
- Score: calculate the score for the best model from the GridSearch
- Compare scores of models to find best overall model
- https://github.com/linushen/coursera_final_cap/blob/main/Prediction.ipynb

```python
data = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.clo
```

```python
# students get this
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
```

Python

```python
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_stat
```

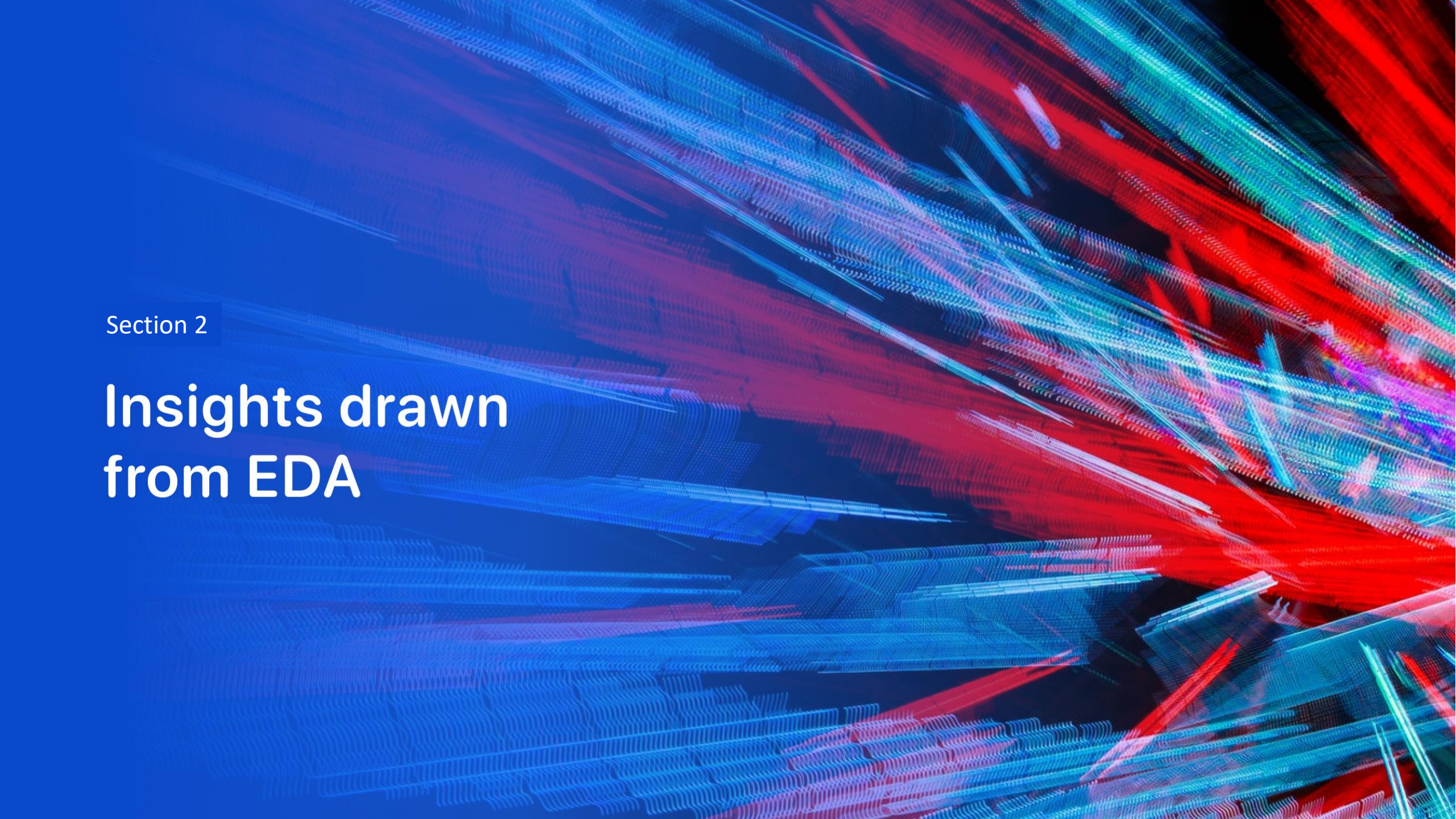
```python
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}

lr=LogisticRegression()

my_gs = GridSearchCV(
    estimator = lr,
    param_grid = parameters,
    scoring = 'accuracy',
    cv = 10
)

logreg_cv = my_gs.fit(X_train,Y_train)
```

```python
logreg_cv.score(X_test, Y_test)
```

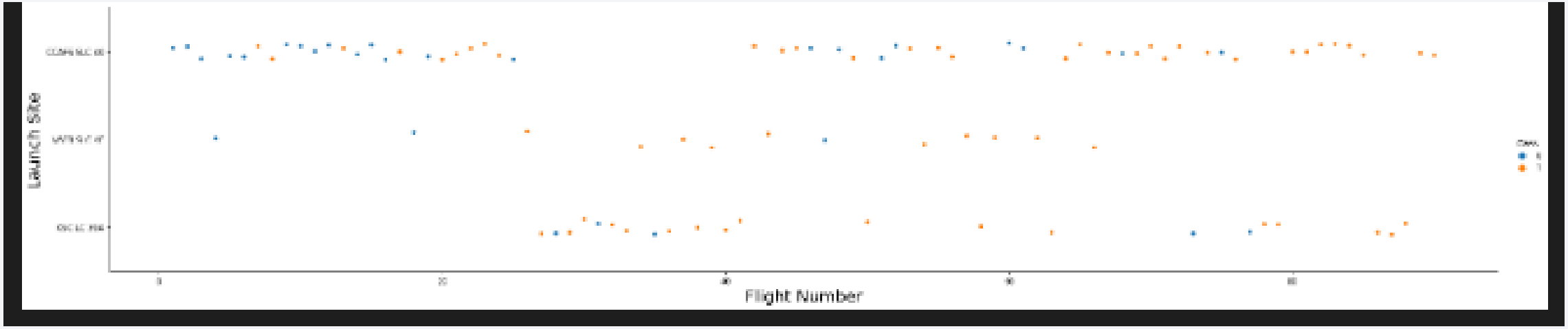
Python

Results

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

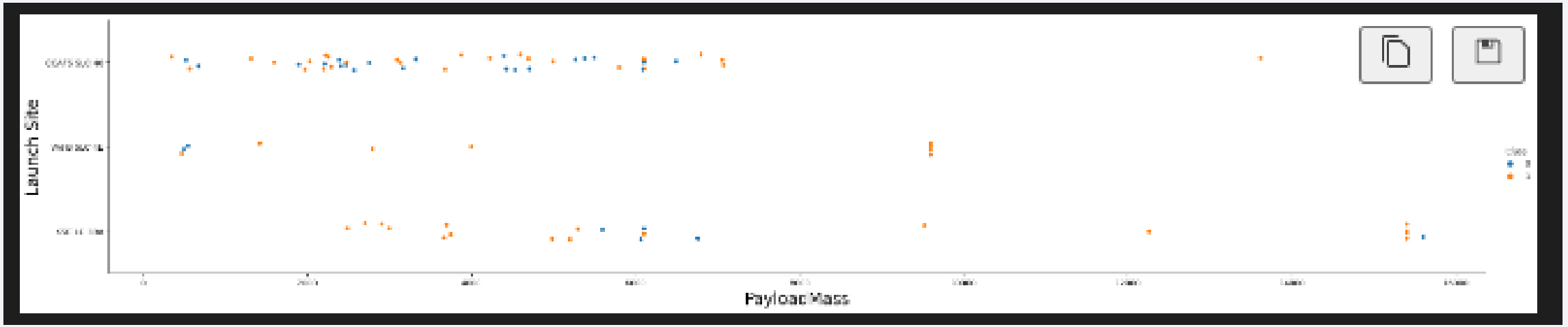

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



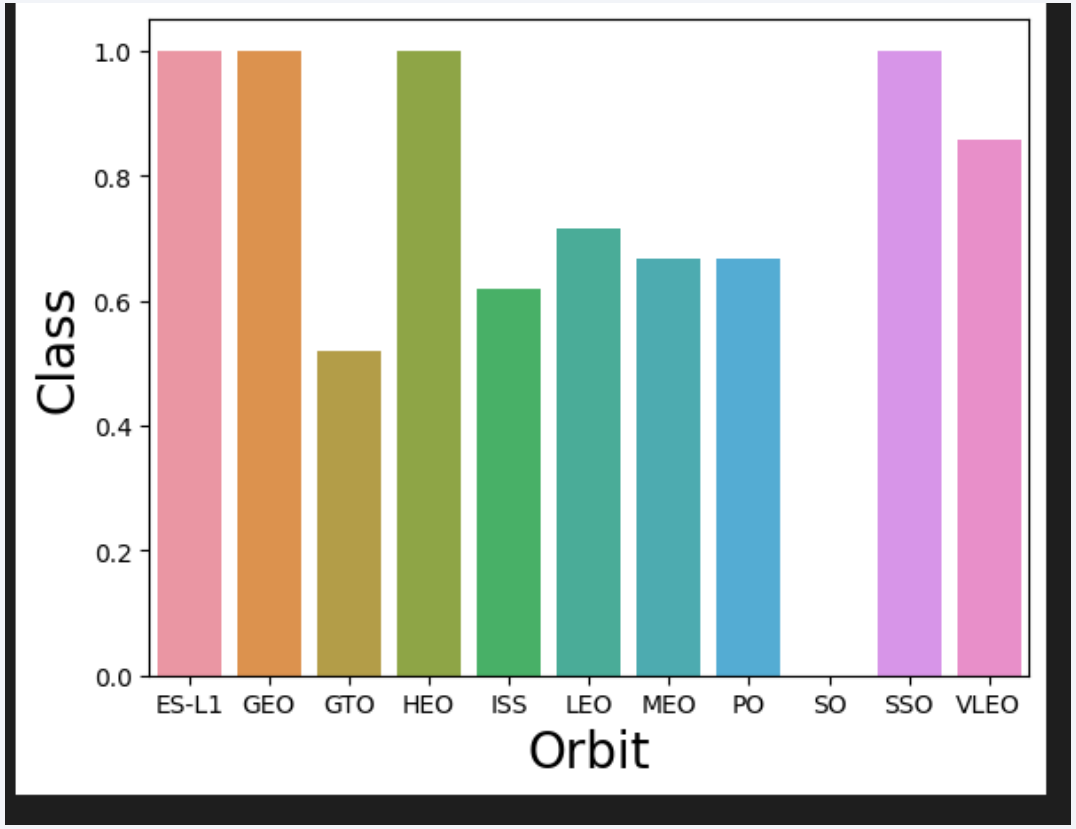
- On the x-axis are the flight numbers and on the y-axis are the names of each launch site. The color indicates if the mission was successful or not (orange/blue).

Payload vs. Launch Site



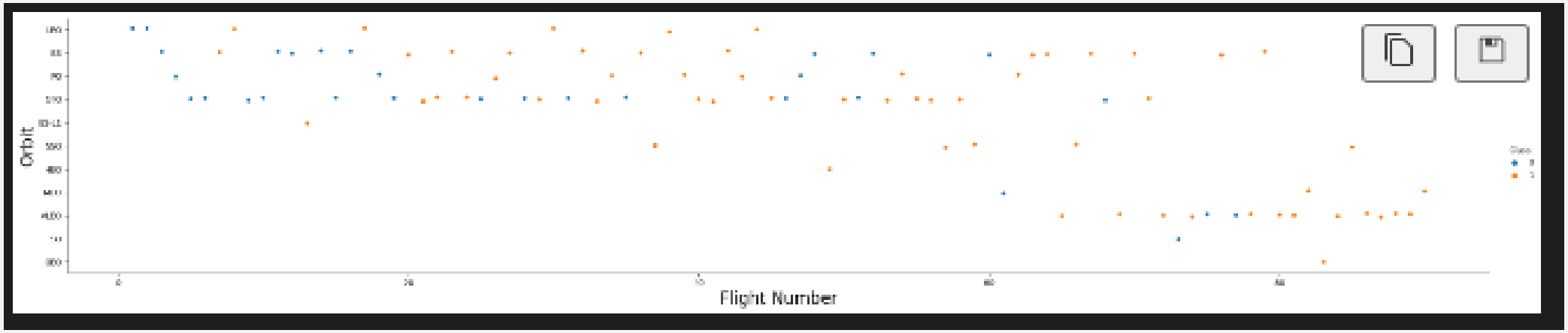
- On the x-axis are the payload mass and on the y-axis are the names of each launch site. The color indicates if the mission was successful or not (orange/blue).

Success Rate vs. Orbit Type



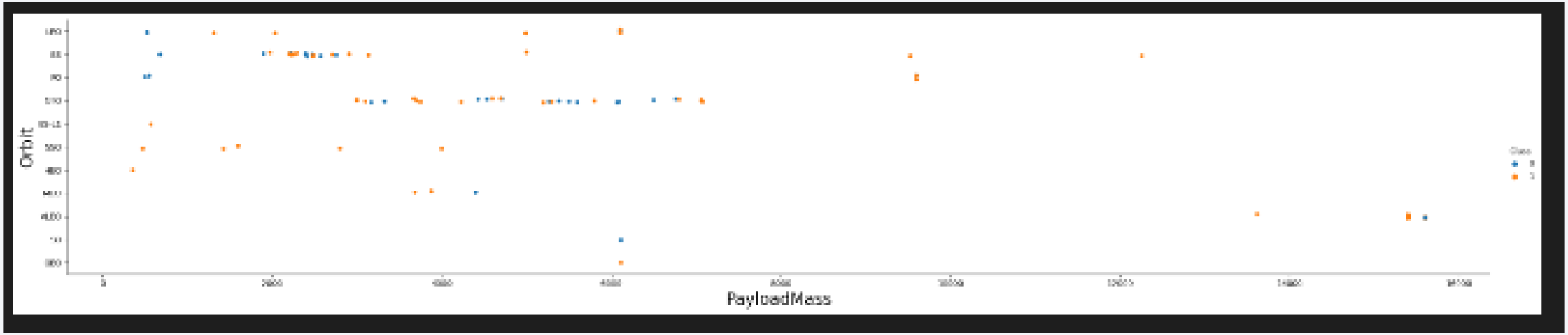
- On the x-axis are the Orbits and on the y-axis you can see the average successrate.

Flight Number vs. Orbit Type



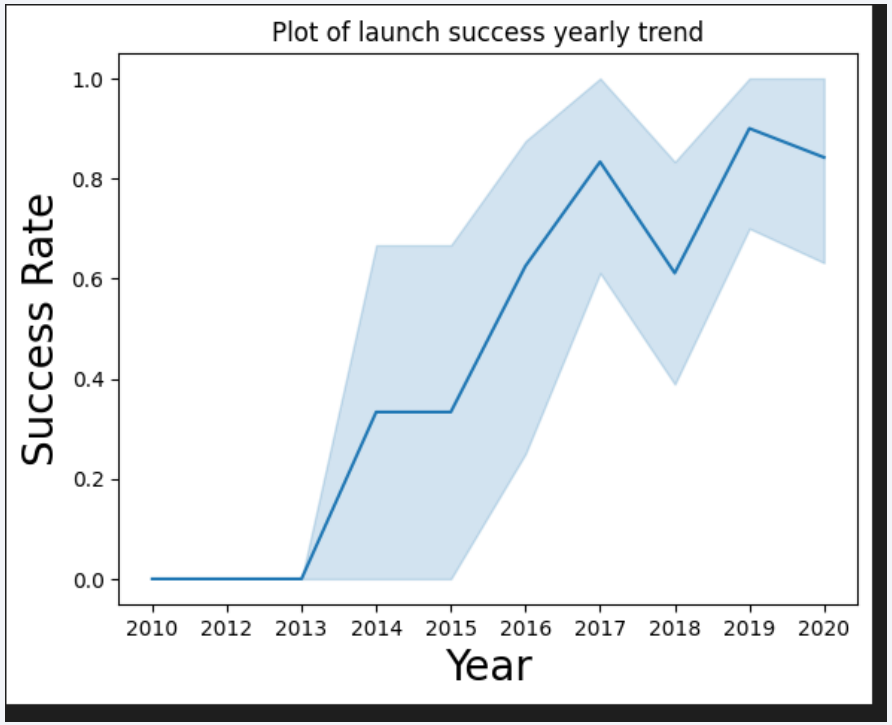
- On the x-axis are the flight numbers and on the y-axis are the different orbits. The color indicates if the mission was successful or not (orange/blue).

Payload vs. Orbit Type



- On the x-axis is the payload mass and on the y-axis are the different orbits. The color indicates if the mission was successful or not (orange/blue).

Launch Success Yearly Trend



- On the x-axis are the years and on the y-axis are the successrate. The blue line shows the average success for each year and the buffer shows the range of max and min.

All Launch Site Names

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

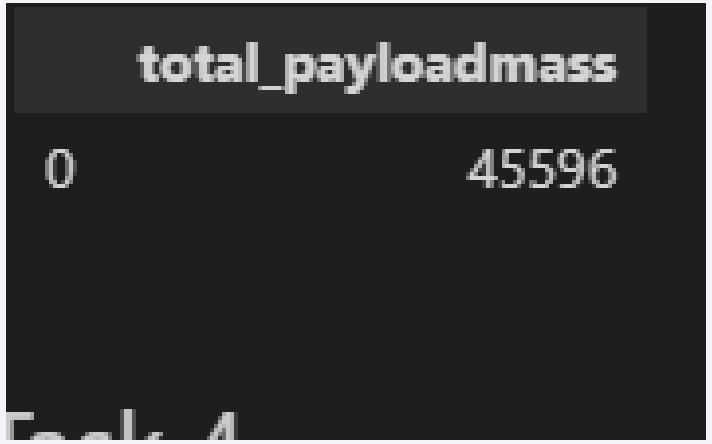
- Find the names of the unique launch sites
- Use “DISTINCT” to get all unique ls

Launch Site Names Begin with 'CCA'

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcom
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Succe
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Succe
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Succe
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Succe
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Succe

- Find 5 records where launch sites begin with `CCA`
- Use “WHERE” statement to specify that Is have to start with “CCA”

Total Payload Mass



- Calculate the total payload carried by boosters from NASA
- Use simple “SUM” to calculate total payload

Average Payload Mass by F9 v1.1

	avg_payloadmass
0	2928.4

- Calculate the average payload mass carried by booster version F9 v1.1
- Similar to task before, but use “AVG” instead of “SUM” and specify the boosterversion = F9 v1.1

First Successful Ground Landing Date

	firstsuccessfull_landing_date
0	2015-12-22

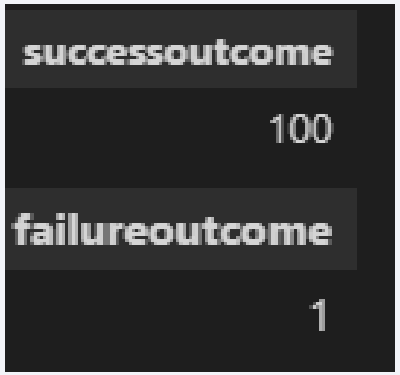
- Find the dates of the first successful landing outcome on ground pad
- As stated in hint, we can use “MIN” on dates to get the earliest date of successful landing

Successful Drone Ship Landing with Payload between 4000 and 6000

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
- Just use “AND” in the “WHERE” statement to specify the correct range (4000<x<6000)

Total Number of Successful and Failure Mission Outcomes



- Calculate the total number of successful and failure mission outcomes
- We “COUNT” all successful/unsuccessful outcomes

Boosters Carried Maximum Payload

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

- List the names of the booster which have carried the maximum payload mass
- We can use subquery to filter for the “MAX” payload mass

2015 Launch Records

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Use “BETWEEN” to specify the range of dates we want to allow

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

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

- 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
- Similar to task before but we also “GROUP BY” landing outcomes

Section 3

Launch Sites Proximities Analysis

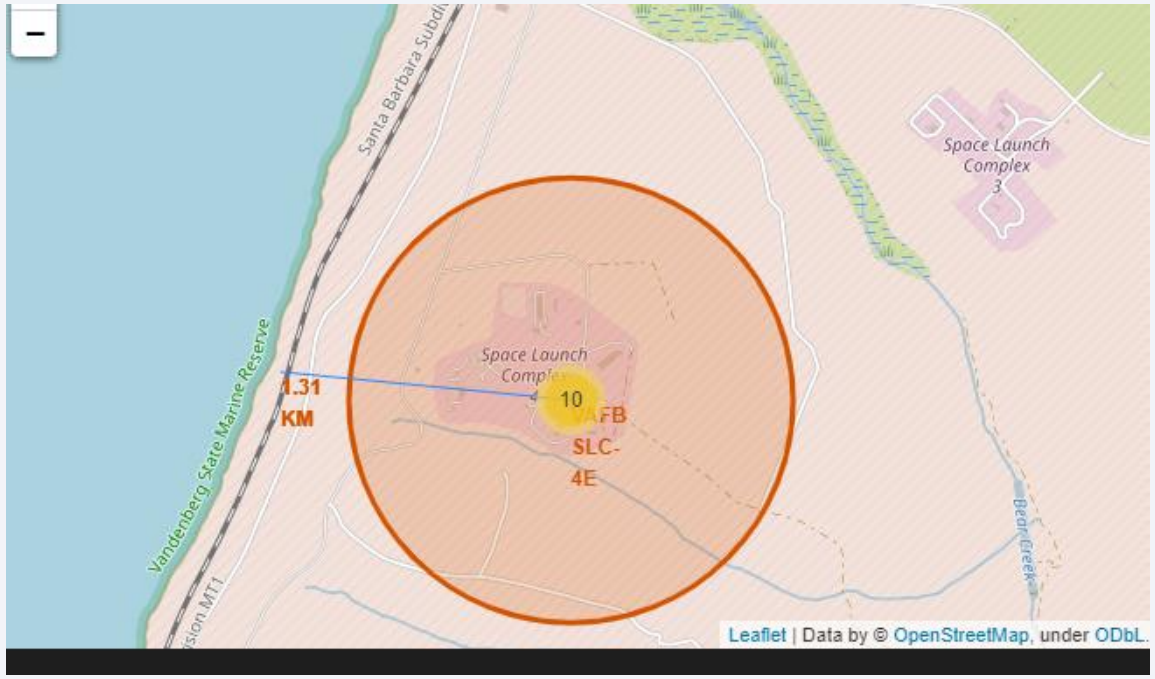
Initial Map

We can see that there are only two areas were the launch sites are located. One on the west- and one on the east coast. Launch sites are highlighted with marker

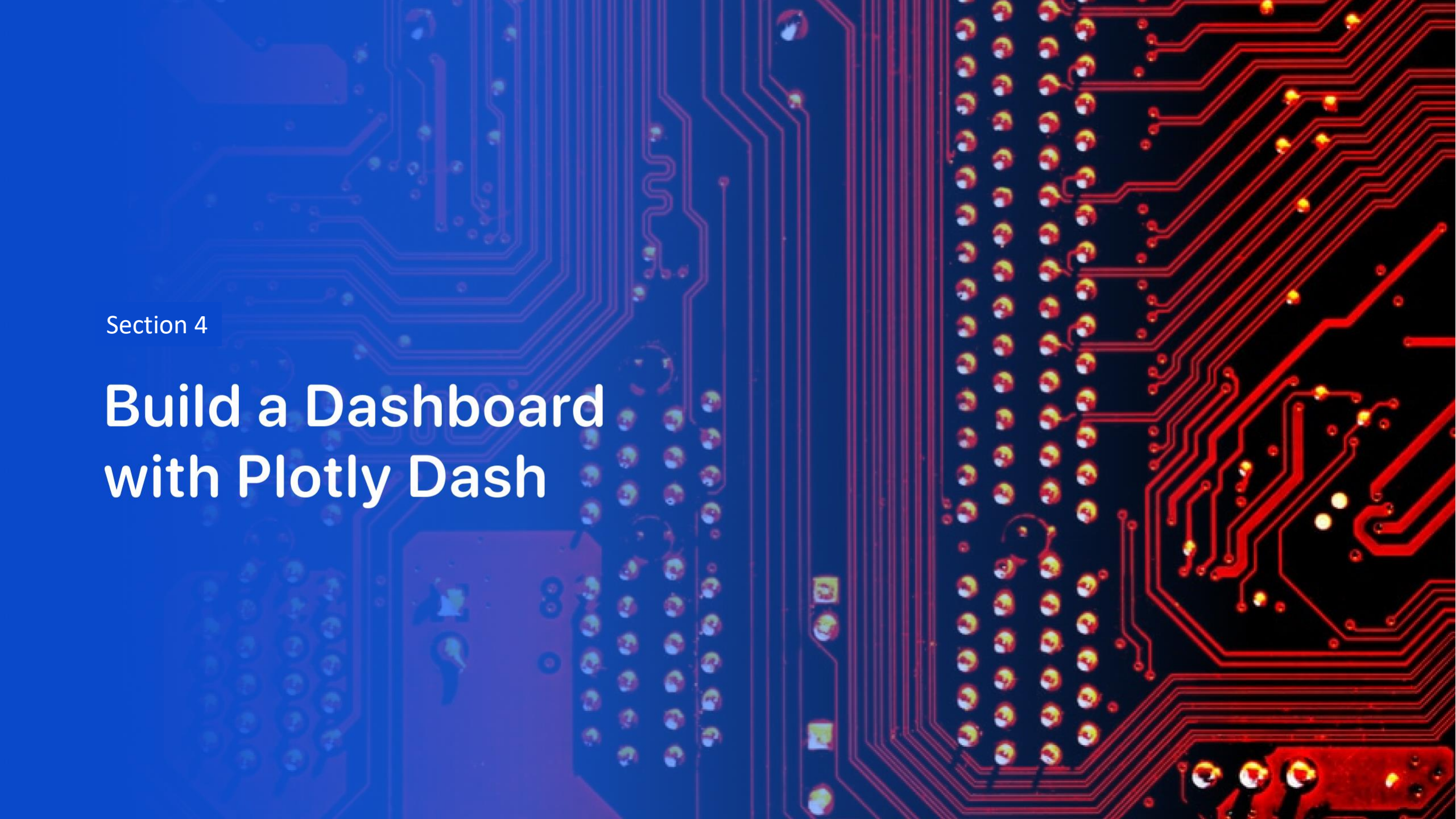
Map with outcome representation

- In addition to the first map, we know can also see if the launch was successful or not. If we zoom in more we can see the different launch sites. We used marker cluster.

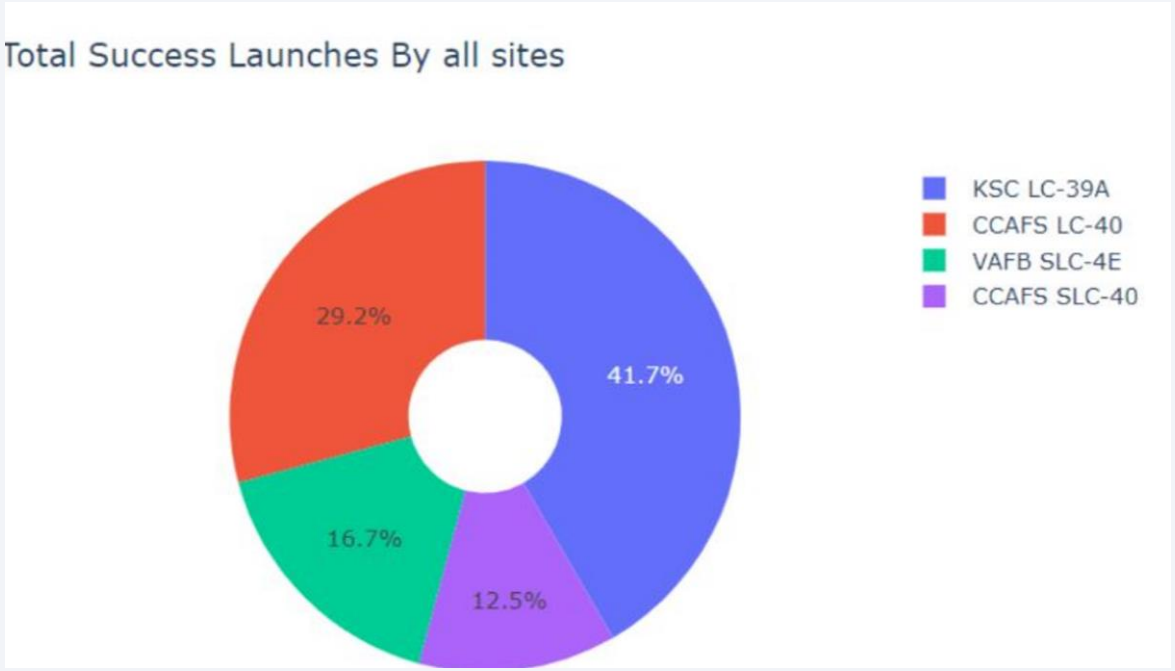
Map with distances



- Now we can also see the distance to important points on the map. In this screenshot for example we see the distance from launch site to the closest railroad (1,31 KM).


Section 4
Build a Dashboard with Plotly Dash

Piechart launch success



We can see the which launch site had the most successfull launches. It is easy to see that KSC performed the best

<Dashboard Screenshot 2>

- Replace <Dashboard screenshot 2> title with an appropriate title
- Show the screenshot of the piechart for the launch site with highest launch success ratio
- Explain the important elements and findings on the screenshot

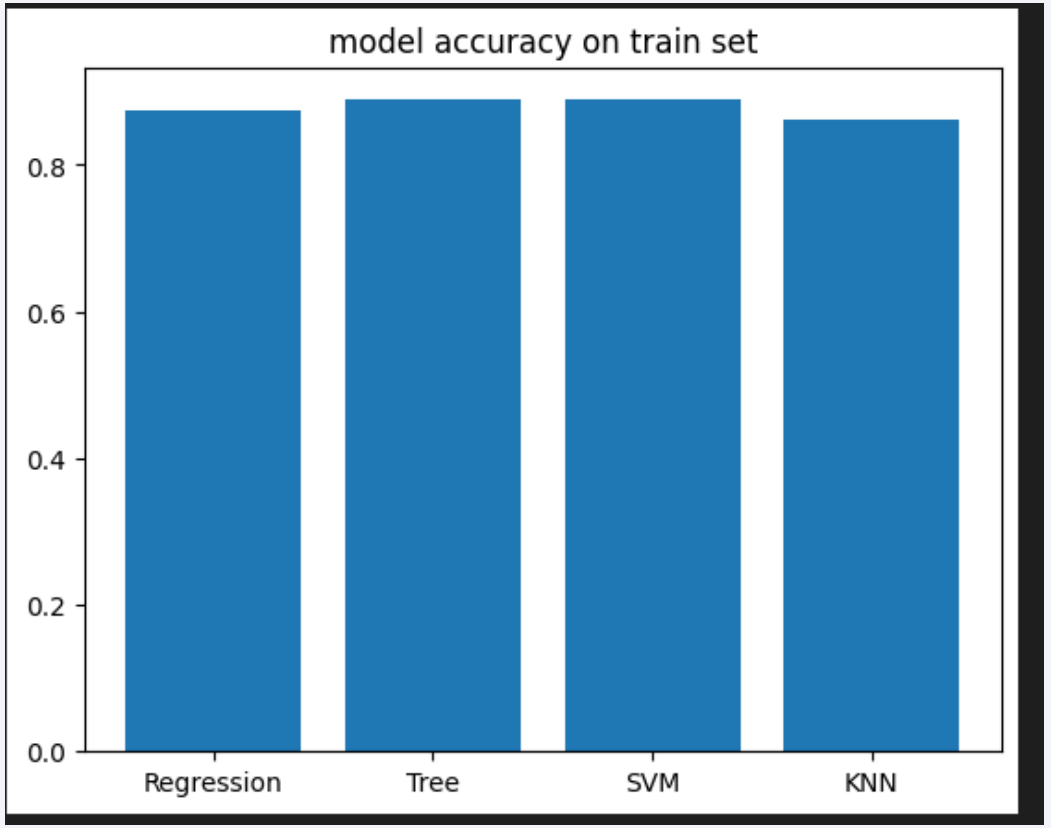
<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title
- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

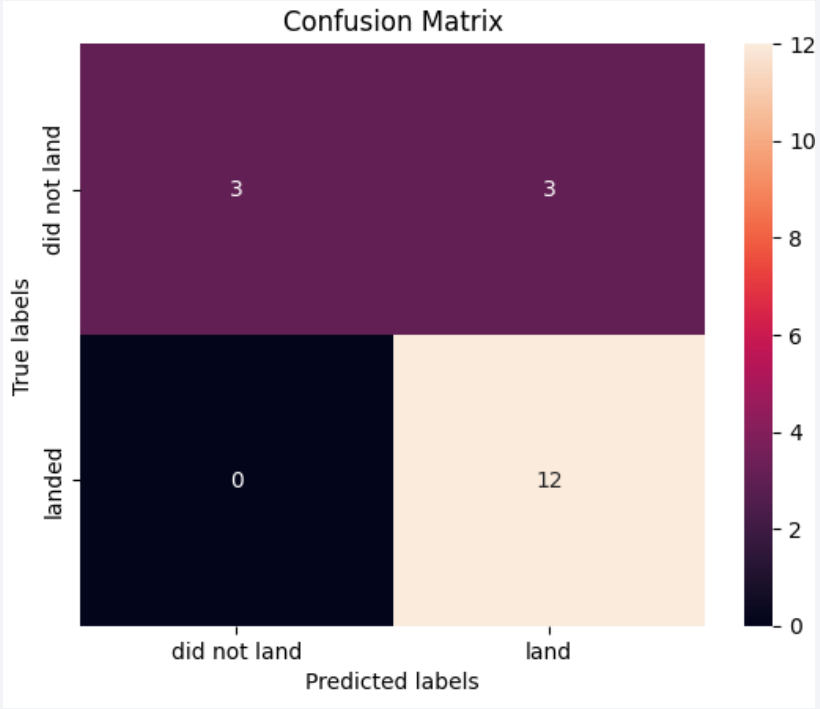
Predictive Analysis (Classification)

Classification Accuracy



- In the plot we can see that the TREE and the SVM models are the best performing models (0.8888)

Confusion Matrix



- We can take a look at the confusion matrix of the best performing model (SVM). The matrix shows how many labels are correctly predicted (land vs. not landed)

Conclusions

- We managed to get a better understanding of the data and put it into a from (folium, …) that other people can understand as well.
- Our models to predict the outcome performed well, but we should keep in mind that the data was very small.

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

Thank you!