IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Walid KINI
23/12/2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

  SpaceX promotes Falcon 9 rocket launches on its website at a price of 62 million dollars, while other providers charge over 165 million dollars per launch. A significant portion of the cost savings stems from SpaceX's ability to reuse the first stage. Consequently, the ability to predict the successful landing of the first stage becomes crucial in estimating the overall launch cost. This predictive capability is valuable for potential competitors seeking to bid against SpaceX for rocket launches. The project's objective is to develop a machine learning pipeline dedicated to forecasting the likelihood of a successful first stage landing.

- Problems you want to find answers
  - **Data Reliability:** Ensure accurate and comprehensive data on past Falcon 9 launches, especially first stage landing outcomes.
  - **Effective Feature Selection:** Identify and select key features influencing first stage landings for precise model training.
  - **Model Generalization:** Develop a machine learning model that not only performs well on historical data but also generalizes effectively to new, unseen data for future launch predictions.

Section
1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- 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

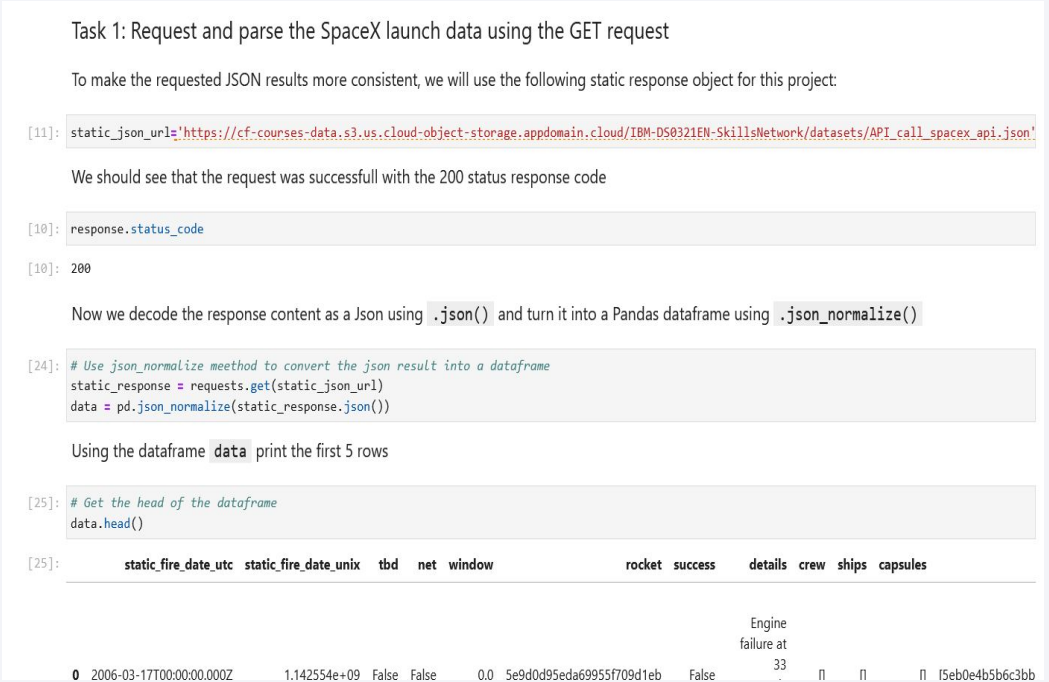
We gathered the data through diverse approaches:

- We utilized a GET request to the SpaceX API for data retrieval.
- Following this, we decoded the response content into JSON format using the .json() function and transformed it into a Pandas dataframe using .json_normalize().
- We conducted data cleaning, addressing missing values and filling them in as needed.
- We performed web scraping on Wikipedia for Falcon 9 launch records using BeautifulSoup.
- The goal was to extract launch records presented as an HTML table, parse the table, and convert the information into a Pandas dataframe for subsequent analysis.

Data Collection – SpaceX API

We used the get request to the SpaceX API to collect data, then proceeded to clean and wrangle the data.

https://github.com/walkidni/IBM-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



Task 1: Request and parse the SpaceX launch data using the GET request

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

```
[11]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

```
[10]: response.status_code
```

```
[10]: 200
```

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

```
[24]: # Use json_normalize meethod to convert the json result into a dataframe
static_response = requests.get(static_json_url)
data = pd.json_normalize(static_response.json())
```

Using the dataframe `data` print the first 5 rows

```
[25]: # Get the head of the dataframe
data.head()
```

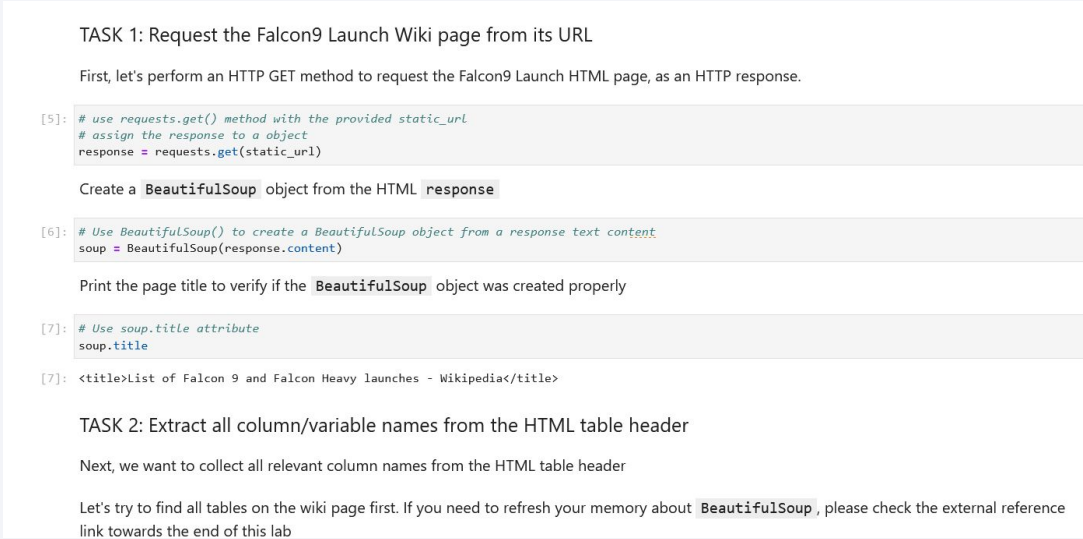
[25]:

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33	[]	[]	[]

Data Collection - Scraping

- Using BeautifulSoup, we scraped falcon 9 launch records from the wikipedia page.
- The table was parsed and turned into a pandas dataframe

https://github.com/walkidni/IBM-Data-Science-Capstone/blob/main/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.

[5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

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

[6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content)
```

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

[7]:
```
# Use soup.title attribute
soup.title
```

[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

Data Wrangling

- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.

https://github.com/walkidni/IBM-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- We investigated the data by visualizing correlations such as the relationship between flight number and launch site, payload and launch site, success rate for each orbit type, flight number and orbit type, as well as the annual trend in launch success.
    - https://github.com/walkidni/IBM-Data-Science-Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- We performed EDA on the data through SQL queries to fetch:
  - 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.

https://github.com/walkidni/IBM-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities.

https://github.com/walkidni/IBM-Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- https://github.com/walkidni/IBM-Data-Science-Capstone/blob/main/dashboard.py

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our

data into training and testing.

- We built different machine learning models and explored different

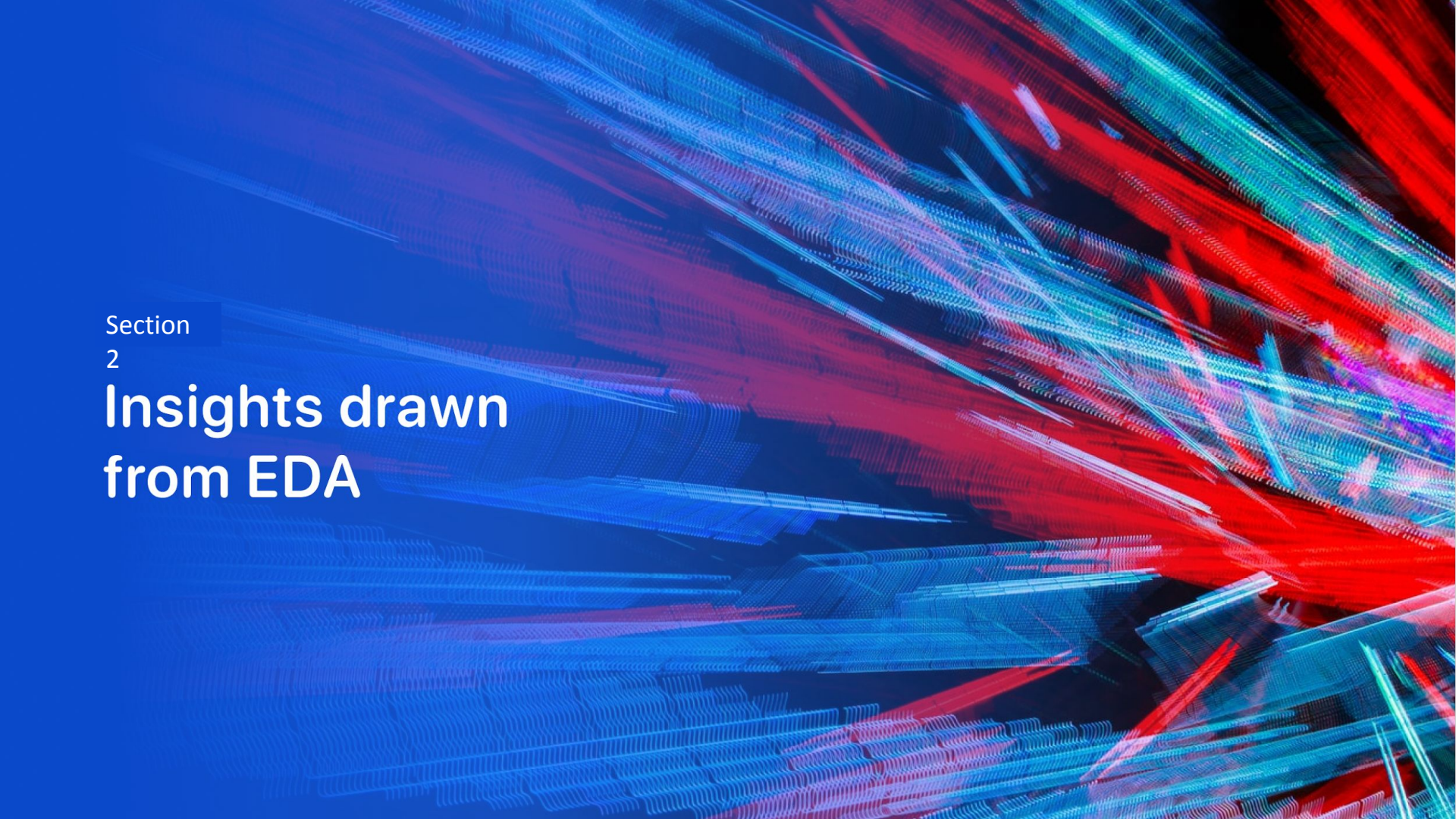
hyperparameters using GridSearchCV.

- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.

https://github.com/walkidni/IBM-Data-Science-Capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

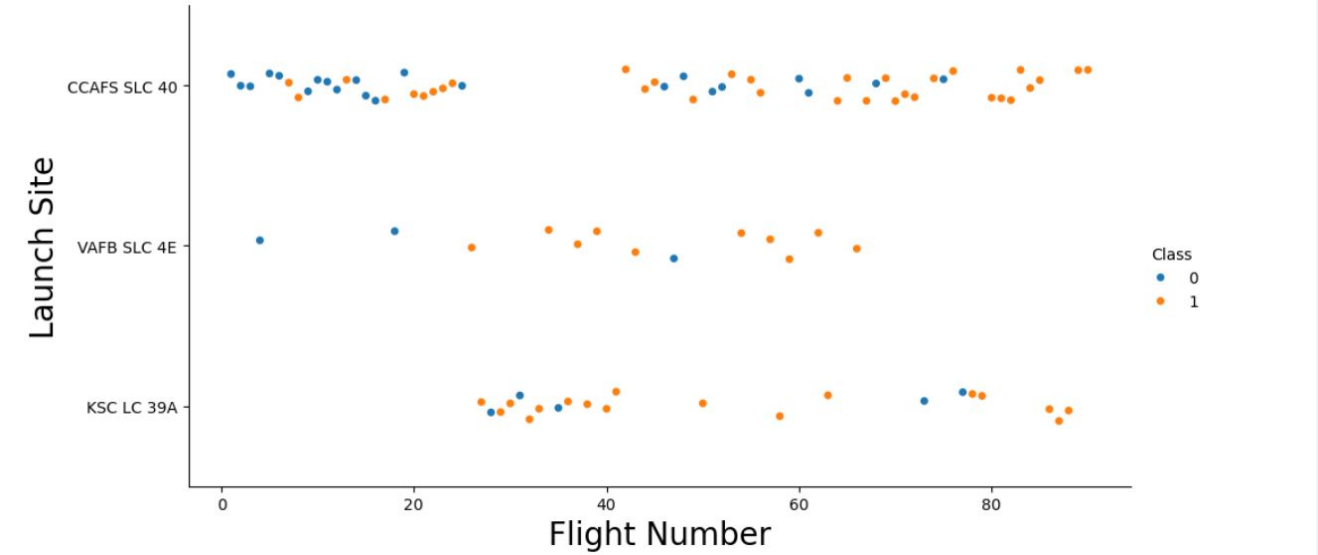
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

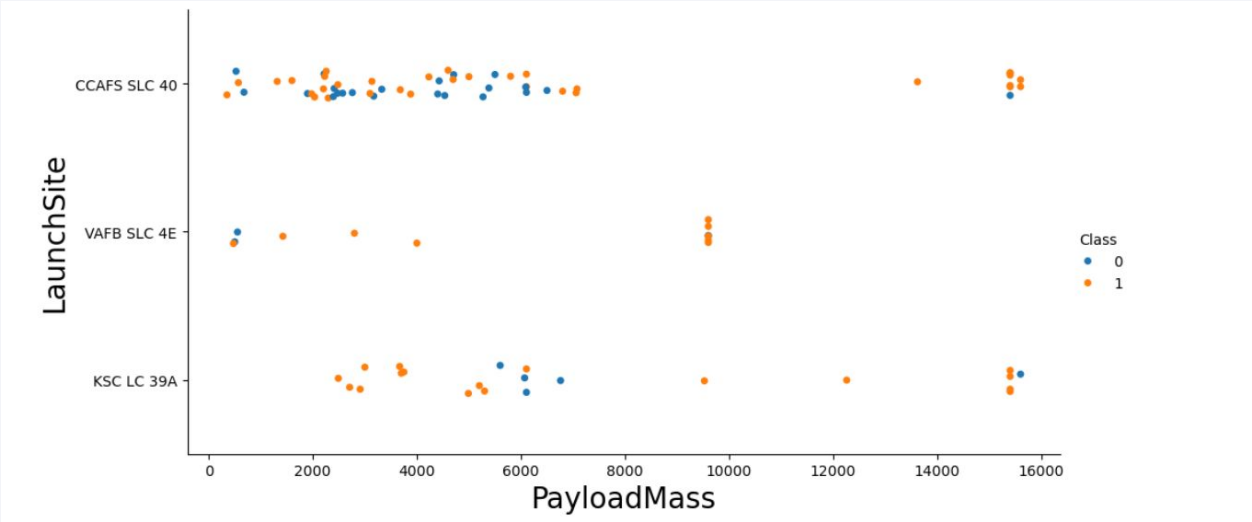
Insights drawn from EDA

Flight Number vs. Launch Site



We noticed that with time, each launch site success rate increases especially for CCAFS SLC 40.

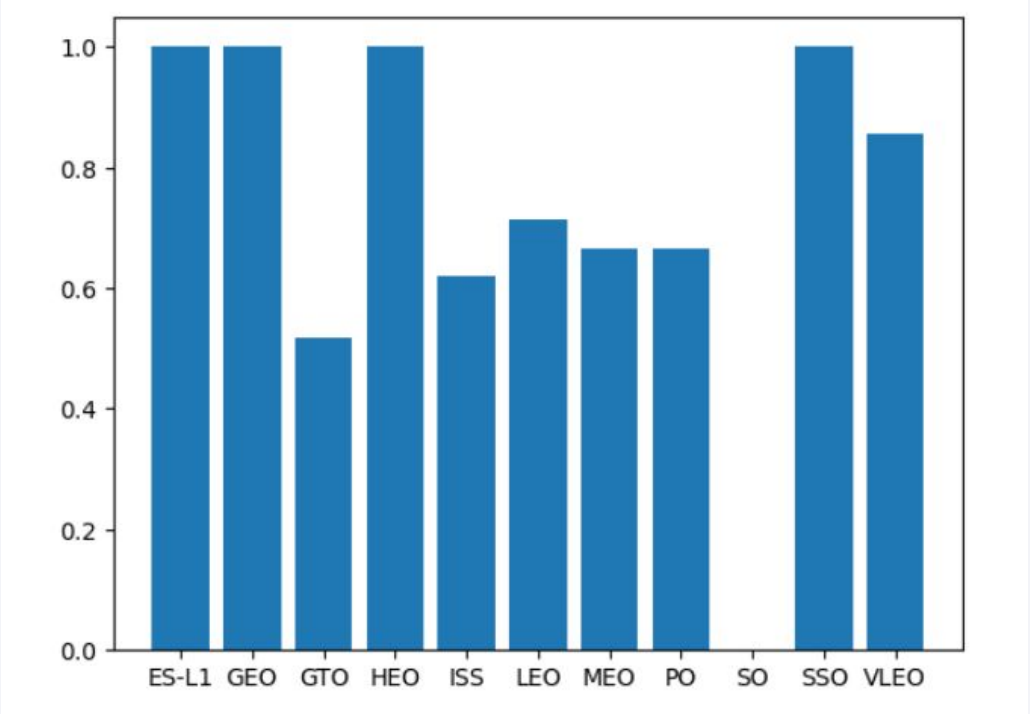
Payload vs. Launch Site



- in KSC LC 39A, the rockets launched have a payload mass over 2500 kg
- In CCAFS SLC 40, the rocket launched have either a payload mass lesser than 8000 kg or greater than 13 000 kg but nothing in between
- In VAFB SLC 4E, the rockets launched have a payload mass lesser than 10 000 kg

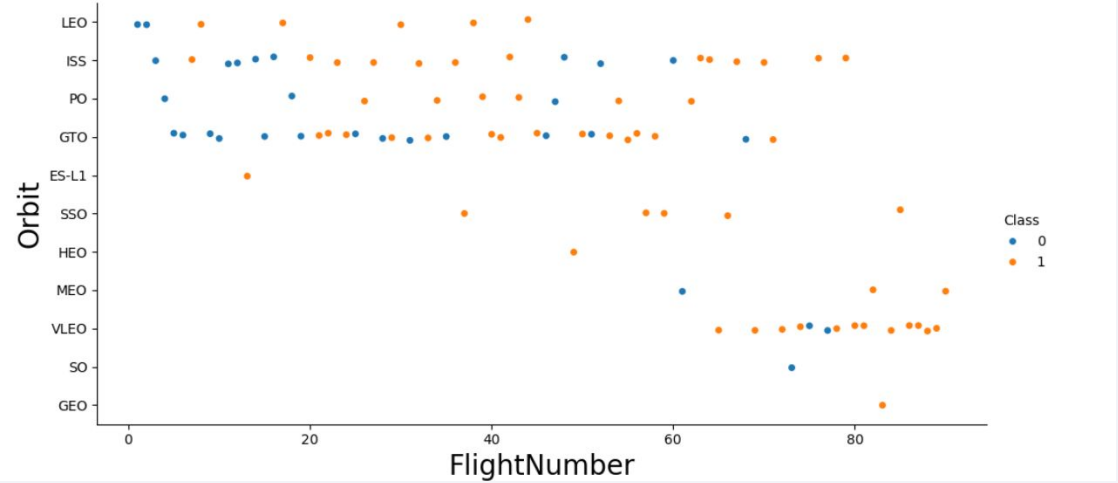
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO and SSO orbits have a perfect success rates



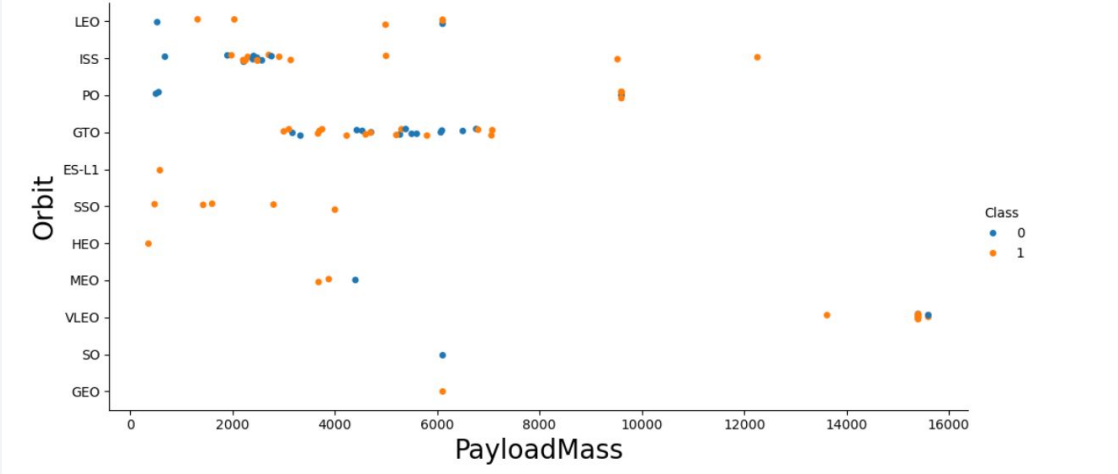
Flight Number vs. Orbit Type

- We notice that the orbits with a perfect success rate have very limited number of launches
- At the beginning of the launches there are more failed operations



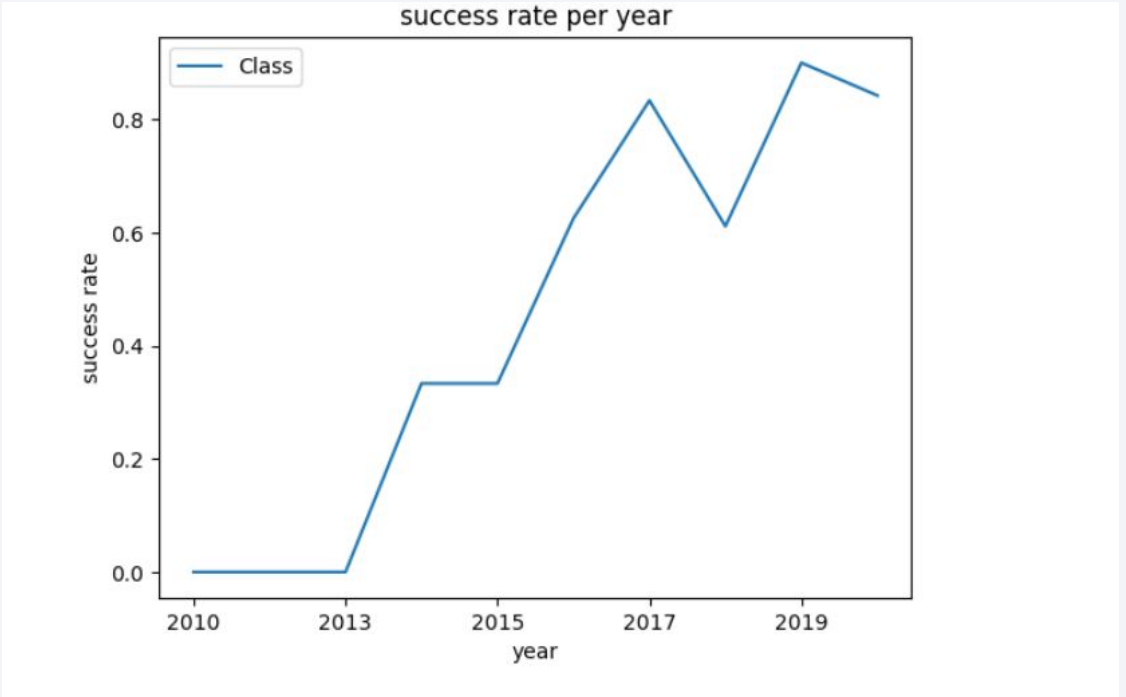
Payload vs. Orbit Type

- VLEO orbit type is only used for payload masses over 13 000 kg
- SSO, HEO and MEO orbit types are used for payload masses less than 5000 kg
- SO and GEO only have one launch
- The remaining Orbit types have a more even distribution of payload mass



Launch Success Yearly Trend

- We notice a growing success rate over the years



All Launch Site Names

- Using the keyword DISTINCT we can select the unique names of the launch sites as shown in the screenshot

```
%sql select distinct Launch_Site from SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

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

Launch Site Names Begin with 'CCA'

- Using the keyword Like we can choose only the launch sites with a name starting with ‘CCA’
- The keyword LIMIT makes sure we only get 5 records

```
%sql select * from SPACEXTABLE 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
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- The SUM function is used to calculate the total payload mass
- The WHERE keyword followed by the condition ensures that the function is only applied to boosters launched by NASA (CRS)

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

```
%sql select SUM(PAYLOAD_MASS__KG_) as TotalPayloadMass from SPACEXTABLE where Customer="NASA (CRS)"
```

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

TotalPayloadMass
45596

Average Payload Mass by F9 v1.1

- The AVG function is applied to the boosters with the version F9 v1.1

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

```
%sql select AVG(PAYLOAD_MASS__KG_) as AveragePayloadMass from SPACEXTABLE where Booster_Version like "F9 v1.1%"
```

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

AveragePayloadMass
2534.6666666666665

First Successful Ground Landing Date

- Using the function MIN on the Date with a condition on the outcome being successful, we queried the first successful ground landing date

```
%sql select MIN(Date) from SPACEXTABLE where Mission_Outcome = "Success"
```

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

MIN(Date)
2010-06-04

Successful Drone Ship Landing with Payload between 4000 and 6000

- We can combine two WHERE clauses with the AND operator to retrieve the names of the boosters with successful drone ship that have a payload between 4000 and 6000

```
[15]: %sql select Booster_Version from SPACEXTABLE where (PAYLOAD_MASS__KG_ between 4000 and 6000) and landing_outcome='Success (drone ship)'
```

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

[15]:

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

Total Number of Successful and Failure Mission Outcomes

- In order to retrieve the total number of success and failures we used a query combining COUNT function with a GROUP BY statement.

```
%sql select Mission_Outcome, COUNT(*) as Total from SPACEXTABLE group by Mission_Outcome;
```

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

Mission_Outcome	Total
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- In order to retrieve the boosters carrying the maximum payload, we used a subquery to obtain the maximum payload mass using a MAX function.

```
%%sql select Booster_Version, PAYLOAD_MASS__KG_
from spacextable
where PAYLOAD_MASS__KG_ = (
    select MAX(PAYLOAD_MASS__KG_)
    from spacextable
)
```

```
 * sqlite:///my_data1.db
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
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

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

```
%%sql select
    strftime('%m', Date) as Month,
    Booster_Version,
    Launch_Site,
    Landing_Outcome
from spacextable
where substr(Date, 0, 5) = '2015'
    and Landing_Outcome = 'Failure (drone ship)';
```

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

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

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.

```
%%sql
select
    Landing_Outcome,
    count(*) as OutcomeCount
from spacextable
where Date between '2010-06-04' and '2017-03-20'
group by Landing_Outcome
order by OutcomeCount desc;
```

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

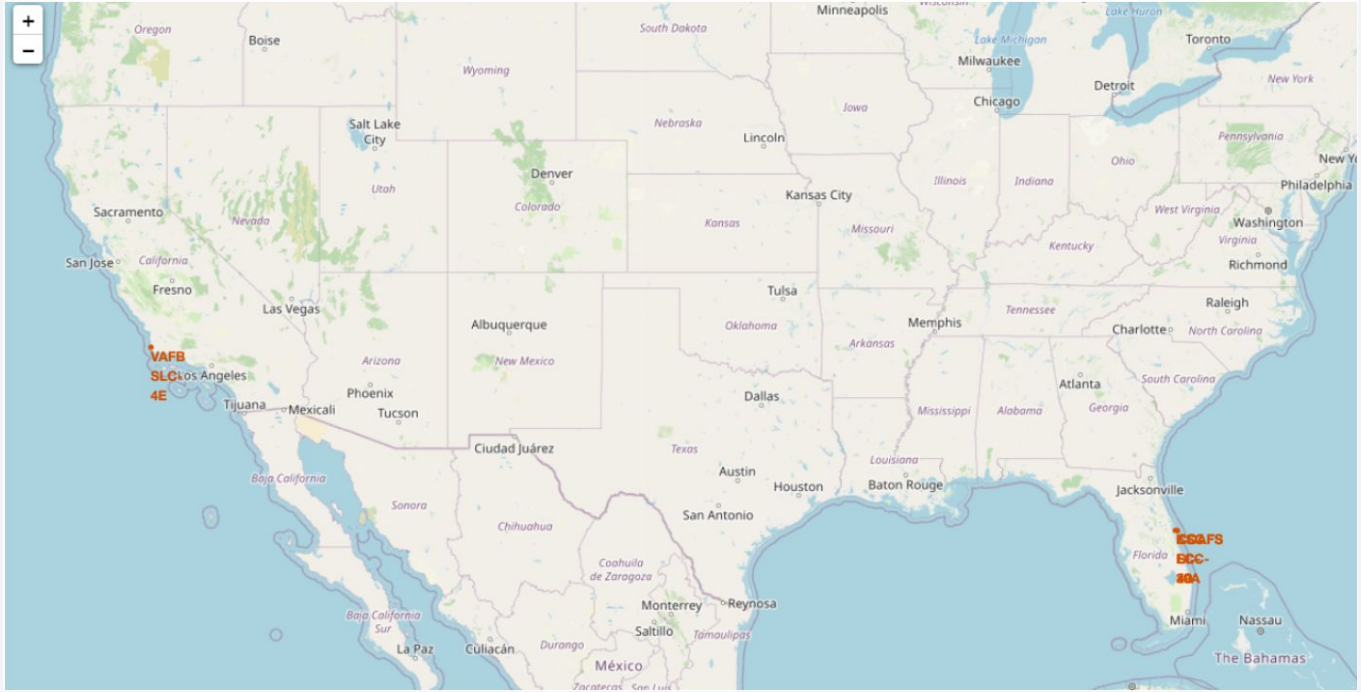
Landing_Outcome	OutcomeCount
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

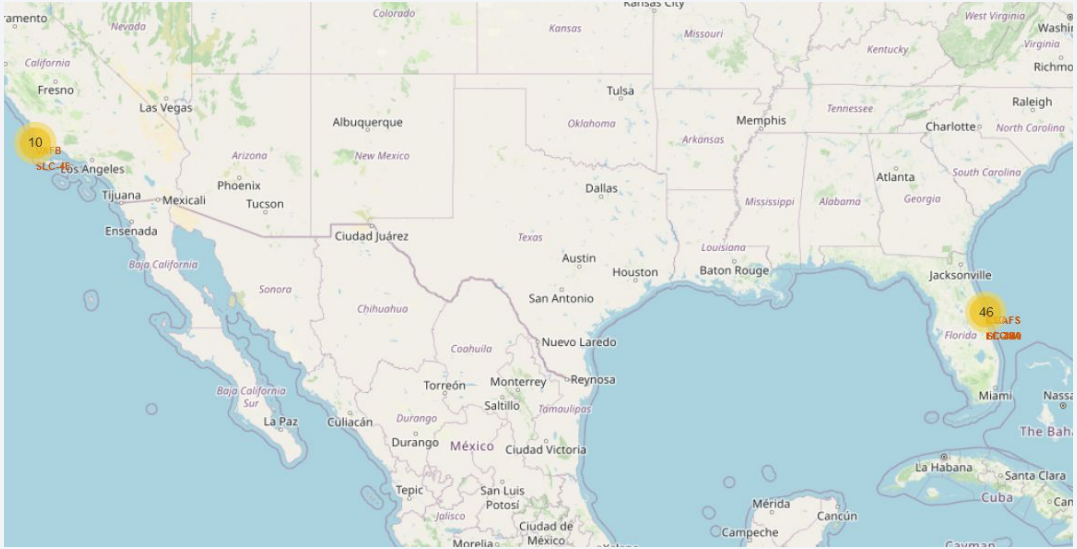
All Launch Sites

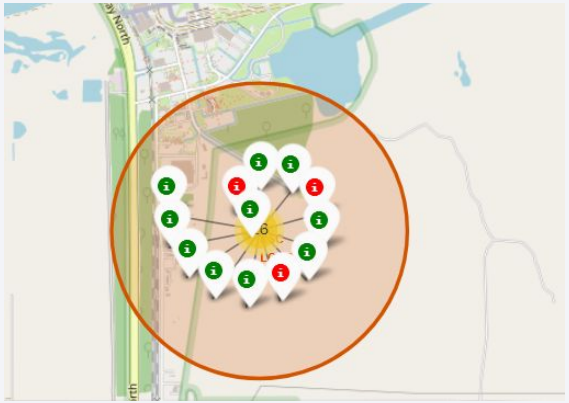
All launch sites are located near coastlines in restricted areas



Successful/Failed Launches For Each Site

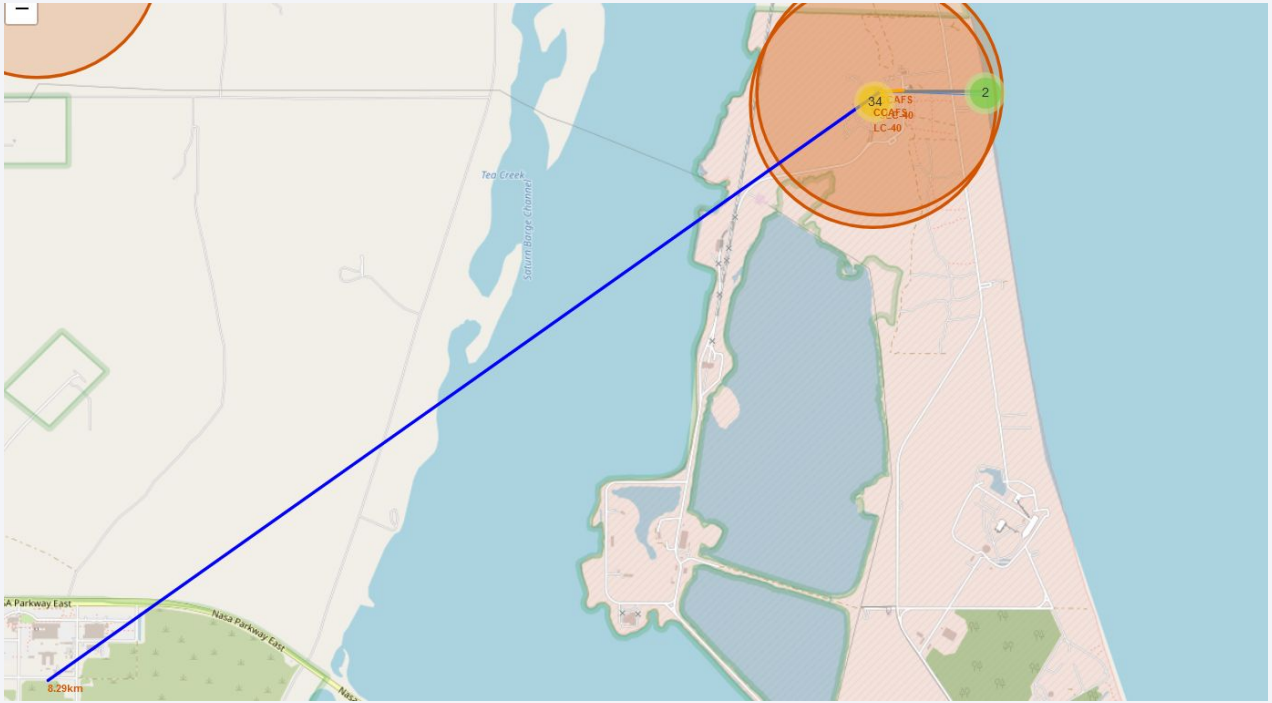
We then added markers of launches in each site. The marker color is red for failed launches and green for successful ones.

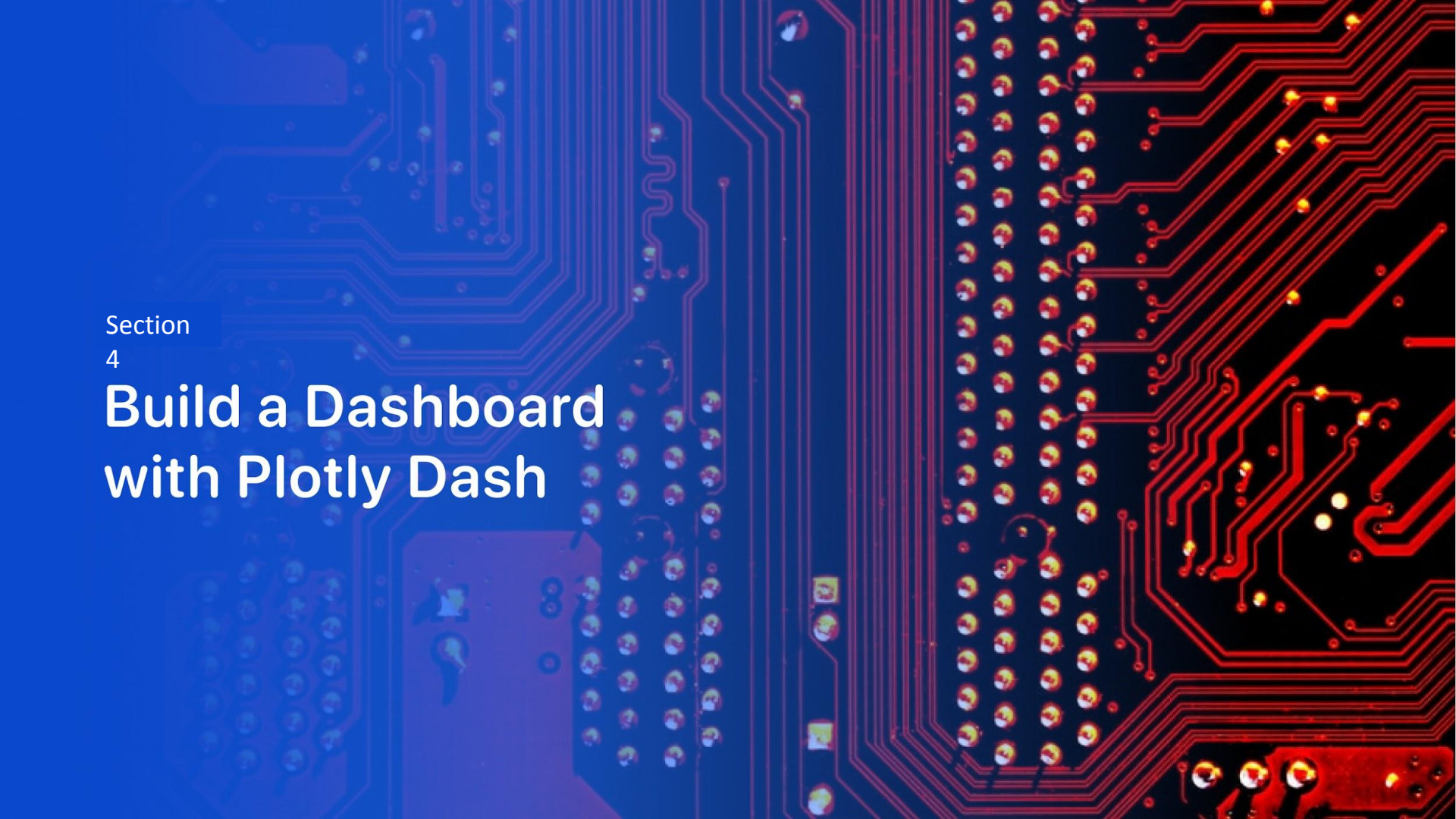




Proximities Of Selected Launch Site

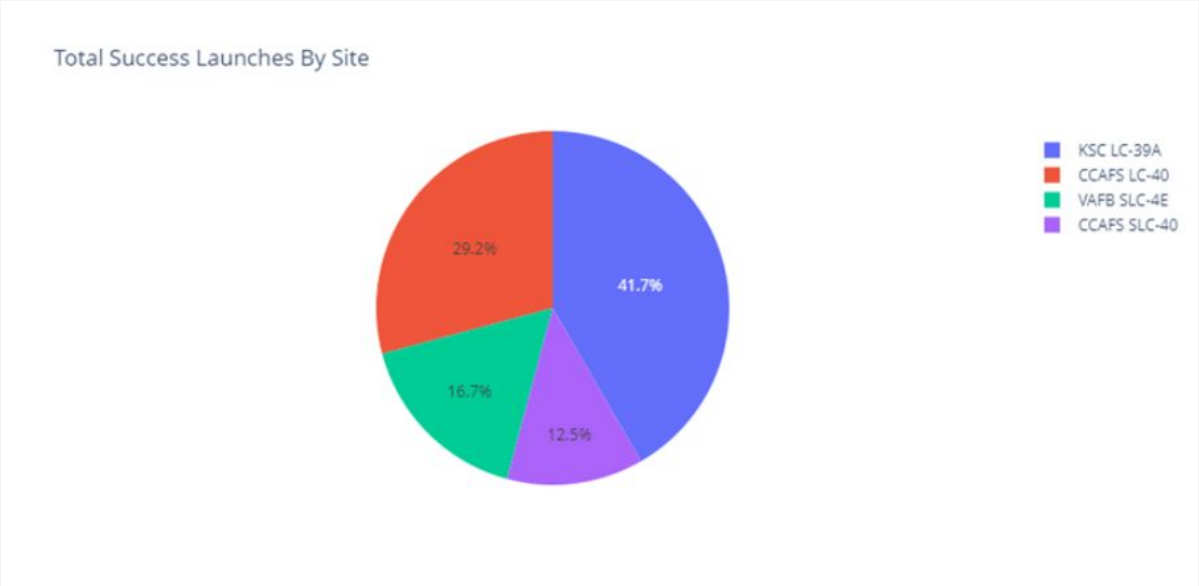
For a selected launch sites we traced lines from railways, coastlines and city .




Section
4
Build a Dashboard
with Plotly Dash

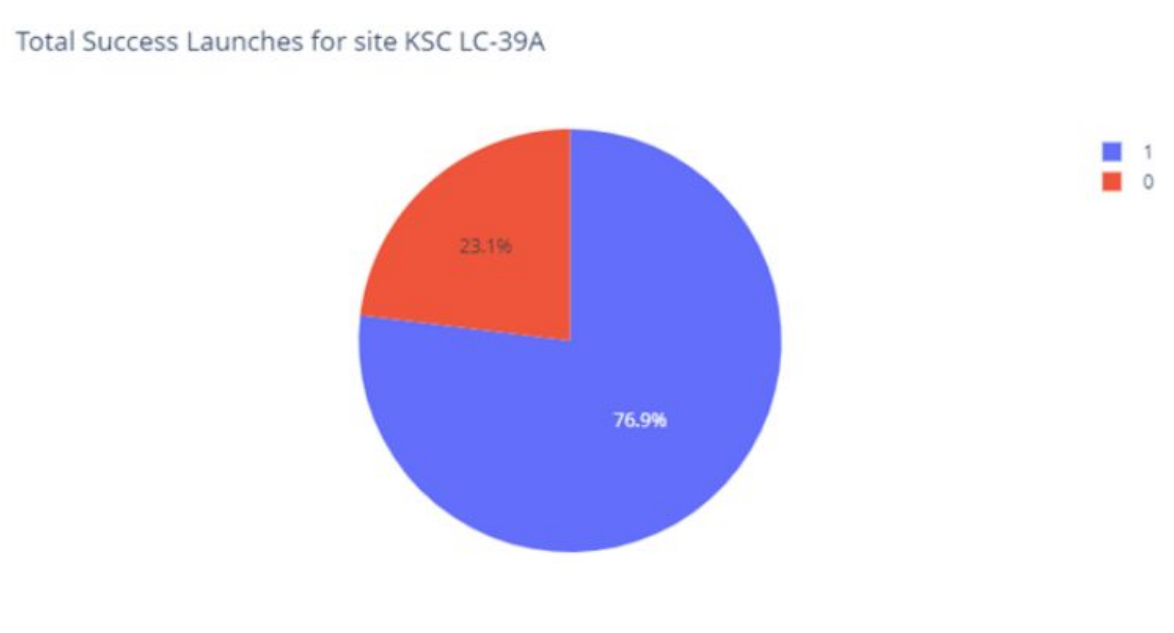
Total Success Launches For Each Site

The launch site with the most successful launches is KSC LC-39A



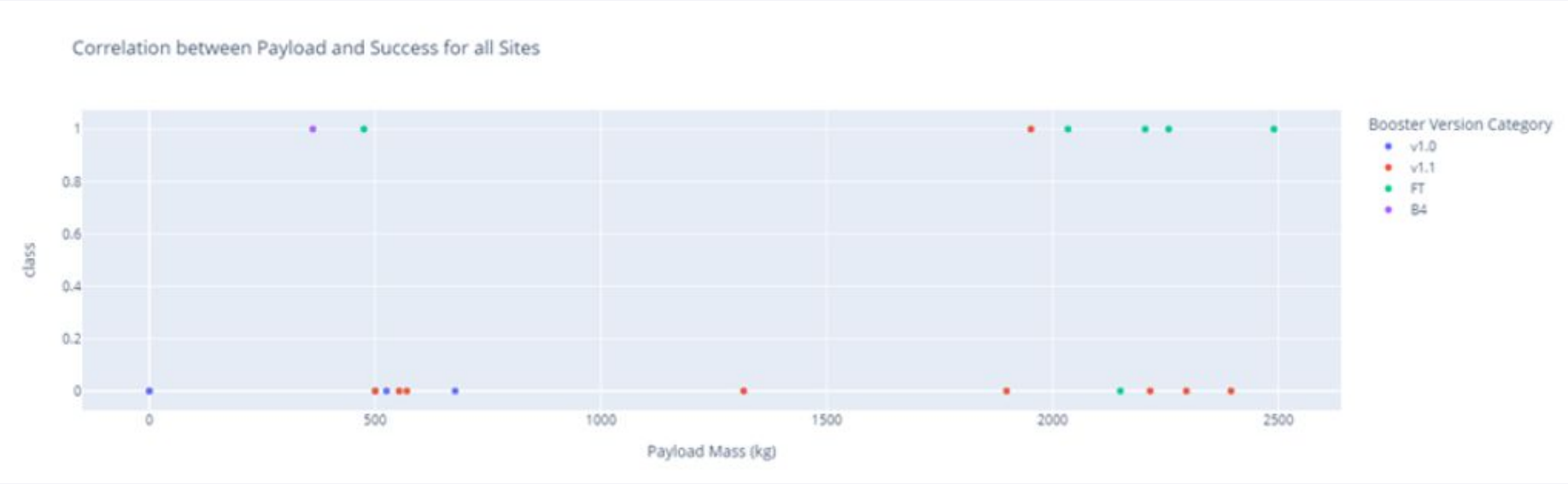
Success and Failure Rates For KSC LC-39A Launches

76.9% of launches in KSC LC-39A launch site are successful

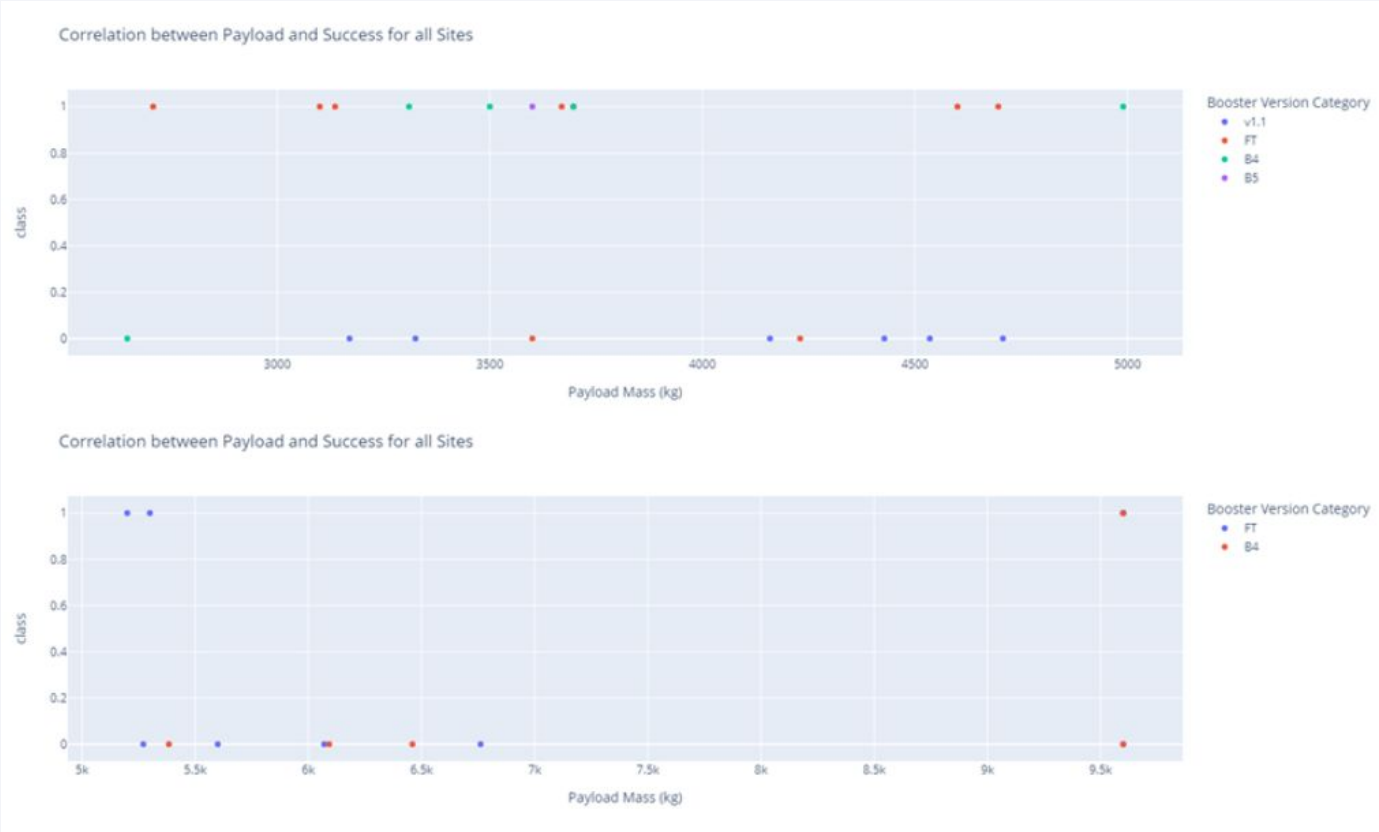


How are Payload and Launch Outcomes correlated

Most successful launches are concentrated in the 2500-5000(kg) payload range



How are Payload and Launch Outcomes correlated

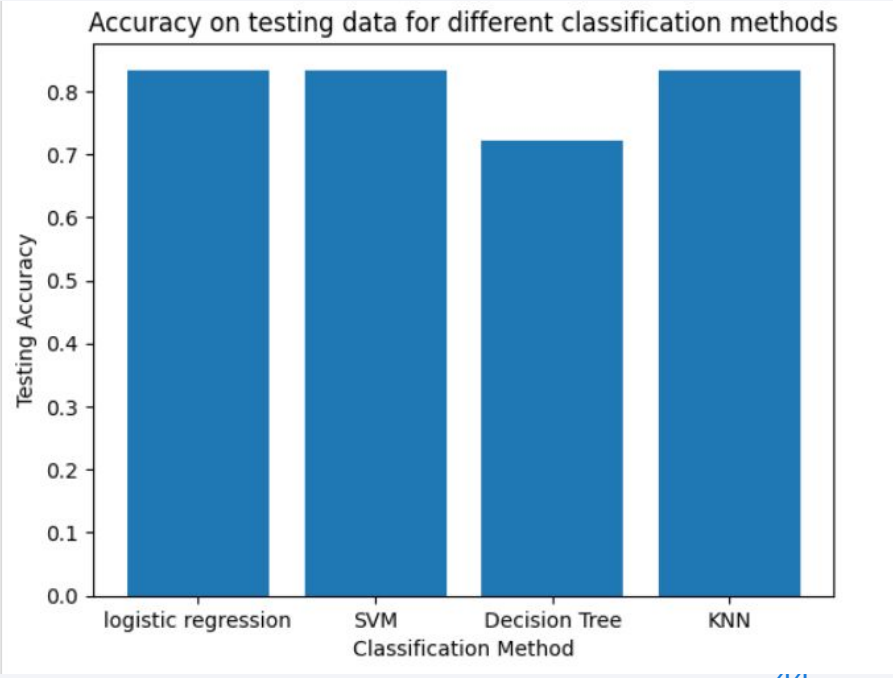


Section
5

Predictive Analysis (Classification)

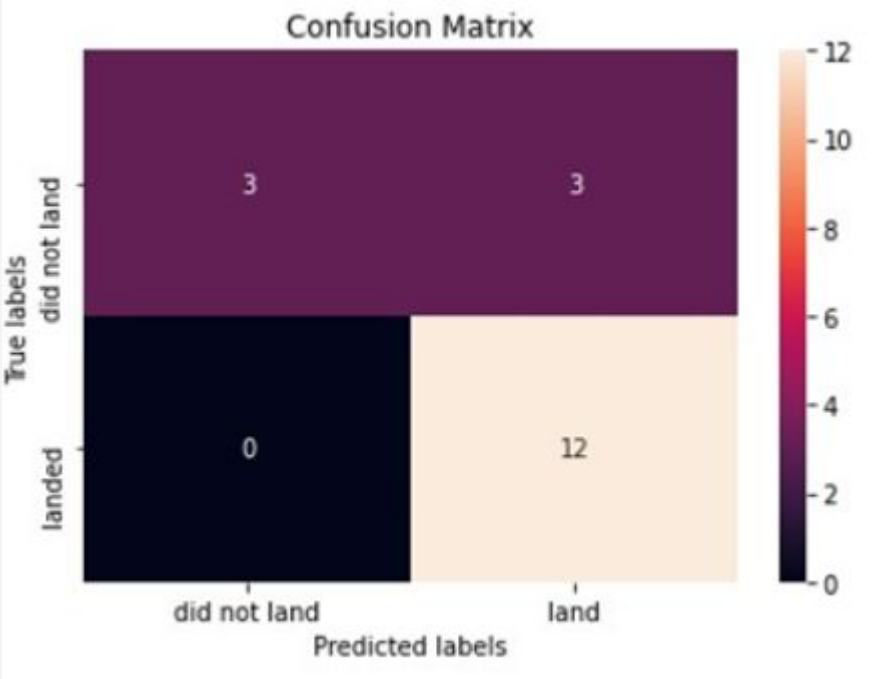
Classification Accuracy

The Decision tree method accuracy is the lowest while all other methods perform the same.



Confusion Matrix

The confusion matrix is the same for the three performing models



Conclusions

- Three methods : Logistic regression, SVM and KNN had the highest accuracy on testing set
- These methods had the same confusion matrix
- We can choose any method out of these three to predict if the first stage of the competitor is successful .

Thank you!