IBM DATA SCIENCE CAPSTONE (ROCKET)

OMKAR SHITOLE

AUGUST 31, 2025

OUTLINE

- Executive Summary
- Introduction
- Methodology
- Data Collection
- Data Wrangling
- Exploratory Data Analysis (Python)
- Exploratory Data Analysis (SQL)
- Interactive Maps and Dashboards
- Predictive Analysis
- Conclusion

EXECUTIVE SUMMARY

Summary of Methodologies Used

- Data Collection through API
- Data Collection with Web Scraping
- Data Wrangling
- Exploratory Data Analysis with Data Visualization
- Exploratory Data Analysis with SQL
- Interactive Visual Analytics with Folium
- Machine Learning based Predictive Analysis

Summary of Results Shown

- Screenshots and Results of Exploratory Data Analysis
- Screenshots and Results of Interactive Analytics
- Screenshots and Results of Predictive Analytics

INTRODUCTION

- Project background:

  Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars while other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problem statements:
  - What factors determine a successful rocket landing?
  - How do the various features determine the success rate of a successful landing?
  - What operating conditions need to be ensured for a successful landing program?

METHODOLOGY

Data collection methodology:

1. Data was collected using SpaceX API and web scraping from Wikipedia.
2. Data Wrangling was performed and One-hot encoding was applied to categorical features
3. Performed exploratory data analysis (EDA) using python visualization and SQL
4. Performed interactive visual analytics using Folium and Plotly Dash
5. Performed ML-based predictive analysis using classification models

DATA COLLECTION

DATA COLLECTION

The data was collected using various methods

- Data collection was executed using ‘get requests’ to the SpaceX API.
- The response content was decoded as a Json using .json() function call and converted into a pandas DataFrame.
- Data cleaning was performed by checking for missing values and replacing them.
- In addition, web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup was performed.
- The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas DataFrame for future analysis.

DATA COLLECTION – SPACEX API

- Used the ‘get request’ to the SpaceX API to collect data, clean the requested data and perform some basic data formatting.
- The link to the notebook is https://github.com/omsh97/IBM-Capstone-Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

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

[6]
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

[7]
```
response = requests.get(spacex_url)
```

Check the content of the response

[8]
```
print(response.content)
```

b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/NN6Ph

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

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:

[9]
```
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=requests.get(static_json_url)
```

[11]
```
response.status_code
```

200

DATA COLLECTION – SCRAPING

- Conducted web scrapping with BeautifulSoup to collect Falcon 9 launch records
- Parsed the data and converted it into a pandas dataframe.
- The link to the notebook is https://github.com/omsh97/IBM-Capstone-Project/blob/main/jupyter-labs-webscraping.ipynb

[4]
```python
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

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.

[10]
```python
# use requests.get() method with the provided static_url
# assign the response to a object
headers = {'User-Agent': 'Mozilla/5.0'}
response = requests.get(static_url, headers = headers)
response.status_code
```

```
200
```

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

[11]
```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, "html.parser")
```

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

[12]
```python
# Use soup.title attribute
print(soup.title)
print(soup.title.string)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

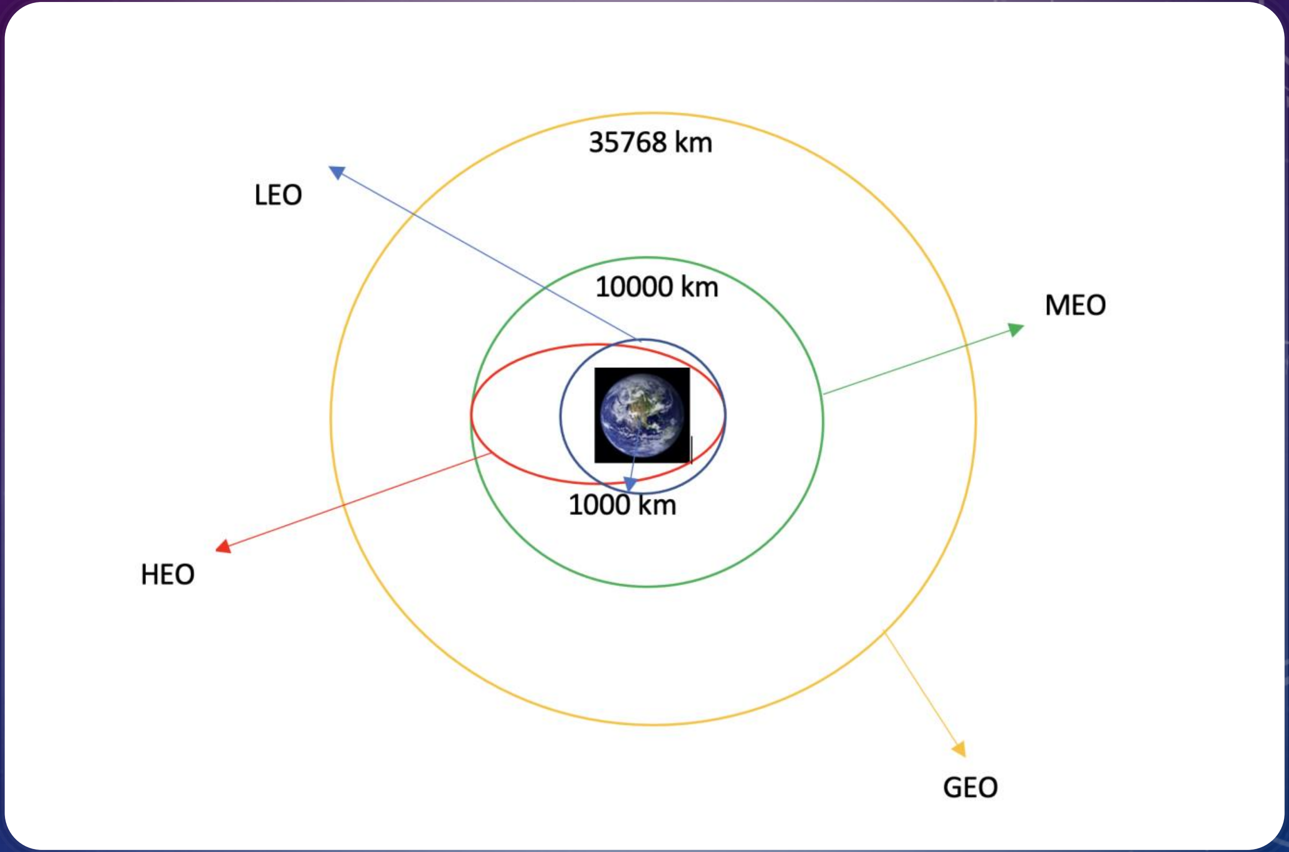
TASK 2: Extract all column/variable names from the HTML table header

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

DATA WRANGLING

DATA WRANGLING

- Tried to find some patterns in the data.
- Calculated the number of launches at each site, and the number and occurrence of each orbits.
- Created landing outcome labels from outcome column and exported the results to a csv file.
- The link to the notebook is https://github.com/omsh97/IBM-Capstone-Project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

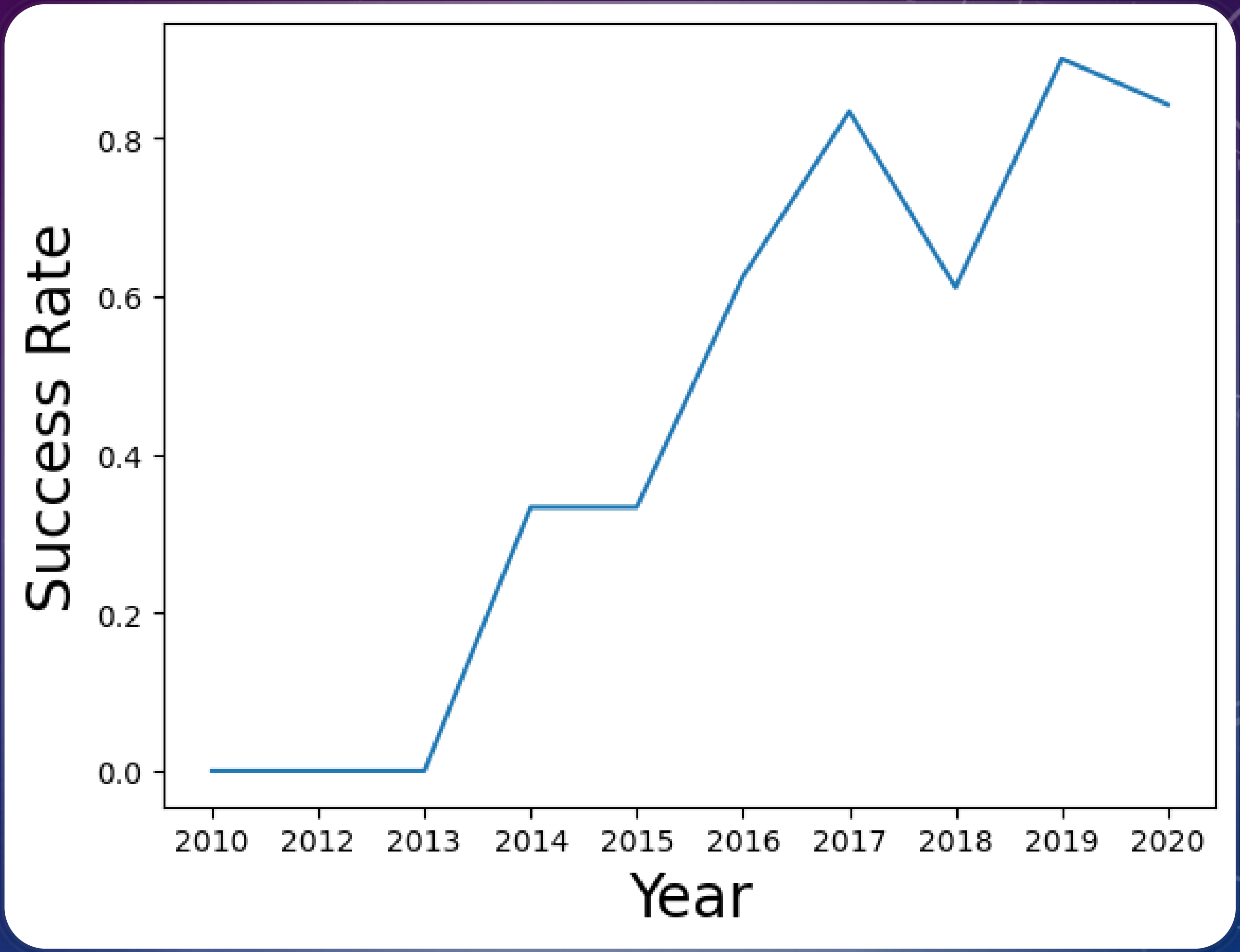


EXPLORATORY DATA ANALYSIS (EDA) WITH PYTHON VISUALIZATION

EDA WITH DATA VISUALIZATION

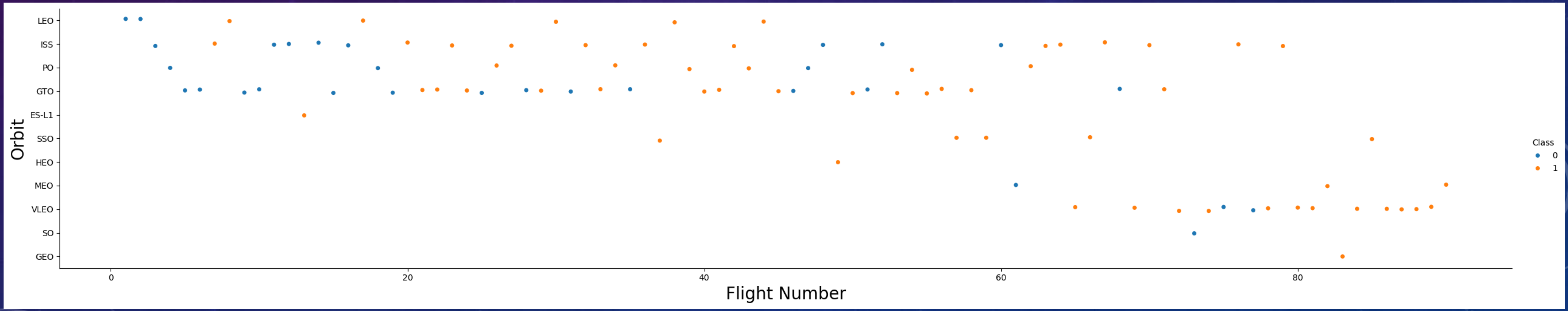
Performed several types of EDA plotting

The link to the notebook is https://github.com/omsh97/IBM-Capstone-Project/blob/main/edadataviz.ipynb



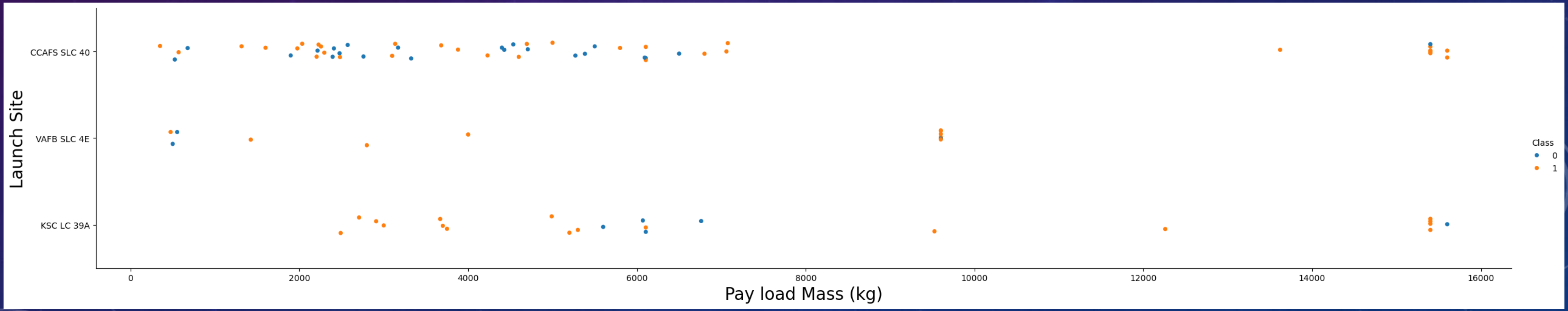
VISUALIZE THE RELATIONSHIP BETWEEN FLIGHT NUMBER AND ORBIT TYPE

The plot below shows the Flight Number vs. Orbit type. It is observed that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.

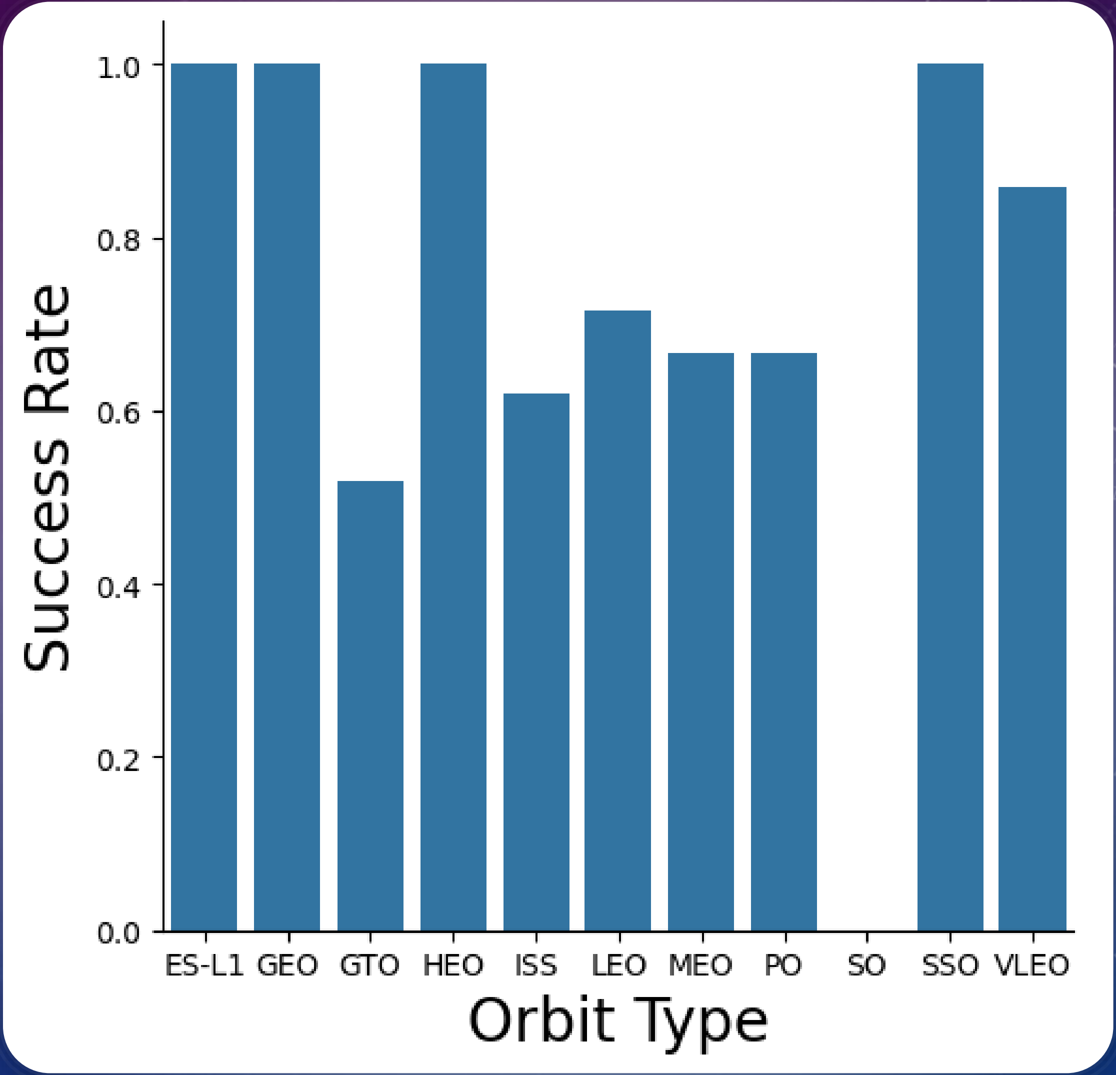


VISUALIZE THE RELATIONSHIP BETWEEN PAYLOAD MASS AND ORBIT TYPE

PO, LEO and ISS orbits achieve more successful landings with a heavier payload



VISUALIZATION OF SUCCESS RATES FOR EACH ORBIT TYPE



Orbits with highest success rates (100%):

- ES-L1
- GEO
- HEO
- SSO

EXPLORATORY DATA ANALYSIS (EDA) WITH SQL

EDA WITH SQL

- The SpaceX dataset was loaded into an SQLite database.
- EDA with SQL was used to get insights from the data.
- Some of the queries used are:
  - 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 link to the notebook is https://github.com/omsh97/IBM-Capstone-Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

ALL LAUNCH SITE NAMES

Used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

Task 1

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

```
%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

EDA (SQL) - LAUNCH SITE NAMES BEGINNING WITH 'CCA'

Used the query below to display **5 records** where launch sites begin with **`CCA`**

Task 2

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

```
%sql select * from SPACEXTABLE where launch_site like 'CCA%' limit 5;
```

[11]

```
 * 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

EDA (SQL) - TOTAL PAYLOAD MASS CARRIED BY BOOSTERS LAUNCHED BY NASA (CRS)

The total payload carried by boosters from NASA is **45596 kg**

Task 3

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

```
%sql select sum(payload_mass__kg_) as total_payload_mass from SPACEXTABLE where customer = 'NASA (CRS)';
```

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

total_payload_mass
45596

EDA (SQL) - AVERAGE PAYLOAD MASS CARRIED BY BOOSTER VERSION F9 V1.1

The average payload mass carried by booster version F9 v1.1 is **2534.67 kg**

Task 4

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

```
%sql select avg(payload_mass__kg_) as average_payload_mass from SPACEXTABLE where booster_version like '%F9 v1.1%';
```

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

average_payload_mass
2534.6666666666665

EDA (SQL) - DATE OF FIRST SUCCESSFUL LANDING OUTCOME IN GROUND PAD WAS ACHIEVED

The date of the first successful landing outcome on ground pad was **22nd December 2015**

Task 5

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

Hint:Use min function

```
%sql select min(date) as first_successful_landing from SPACEXTABLE where landing_outcome = 'Success (ground pad)';
```

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

first_successful_landing
2015-12-22

EDA (SQL) - NAMES OF SUCCESSFUL BOOSTERS WITH PAYLOAD MASS BETWEEN 4000 AND 6000

Used **WHERE** clause to filter for boosters which have successfully landed on drone ship, applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

Task 6

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 SPACEXTABLE where landing_outcome = 'Success (drone ship)' and payload_mass__kg_ between 4000 and 6000;
```

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

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

EDA (SQL) - LIST THE NUMBER OF SUCCESSFUL AND FAILURE MISSION OUTCOMES

Used **GROUP BY** to show mission outcomes

Task 7

List the total number of successful and failure mission outcomes

```
%sql select mission_outcome, count(*) as total_number from SPACEXTABLE group by mission_outcome;
```

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

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

EDA (SQL) - LIST ALL THE BOOSTER VERSIONS THAT HAVE CARRIED THE MAXIMUM PAYLOAD MASS

Determined the booster versions that carried the maximum payload using a **subquery** in the **WHERE** clause and the **MAX()** function.

Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

```
%sql select booster_version from SPACEXTABLE where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTABLE);
```

```
 * 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

EDA (SQL) - LIST THE RECORDS FOR YEAR 2015

Used a combination of the **WHERE** clause, **FROM**, **AS**, and **SUBSTR** to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

Task 9

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, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%sql select substr(date, 6, 2) as month, date, booster_version, launch_site, landing_outcome from SPACEXTABLE where landing_outcome = 'Failure (drone ship)' and substr(date, 1, 4) = '2015';
```

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

month	Date	Booster_Version	Launch_Site	Landing_Outcome
01	2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

EDA (SQL) - RANK THE COUNT OF LANDING OUTCOMES BETWEEN GIVEN DATES

Selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN 2010-06-04 to 2010-03-20**, applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

Task 10

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 count_outcomes from SPACEXTABLE where date between '2010-06-04' and '2017-03-20' group by landing_outcome order by count_outcomes desc;
```

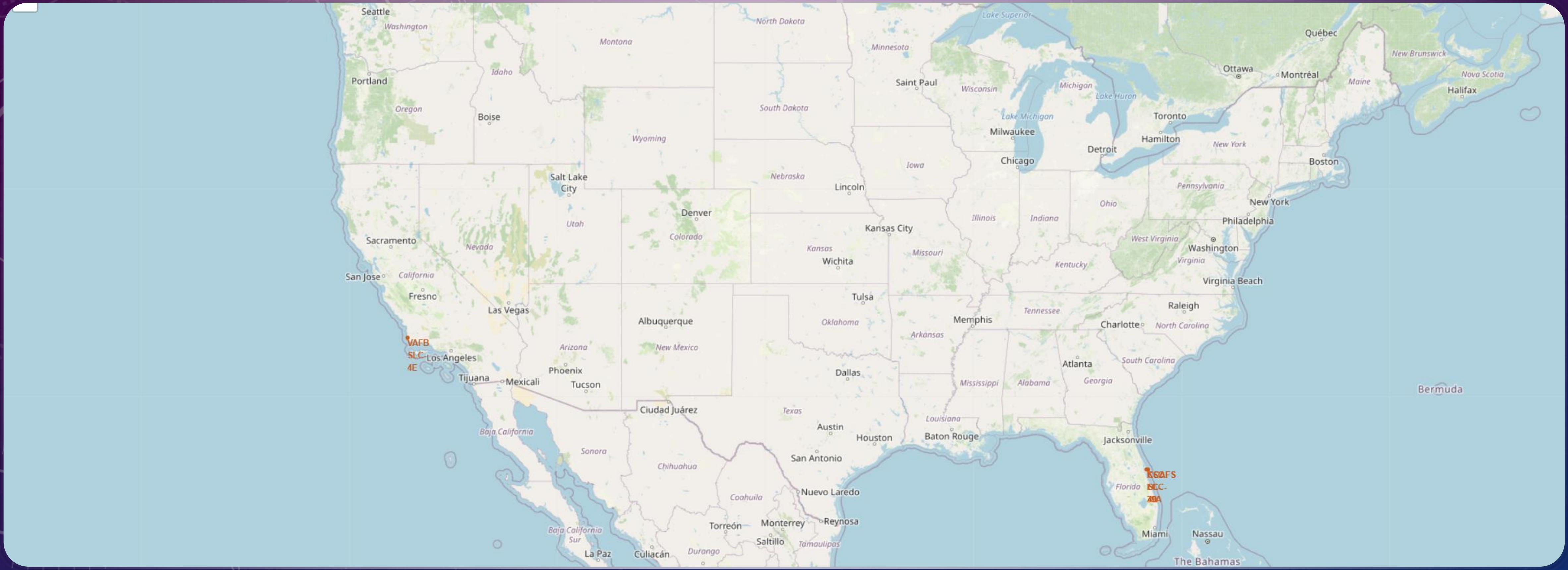
```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	count_outcomes
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

INTERACTIVE LAUNCH SITE MAPS

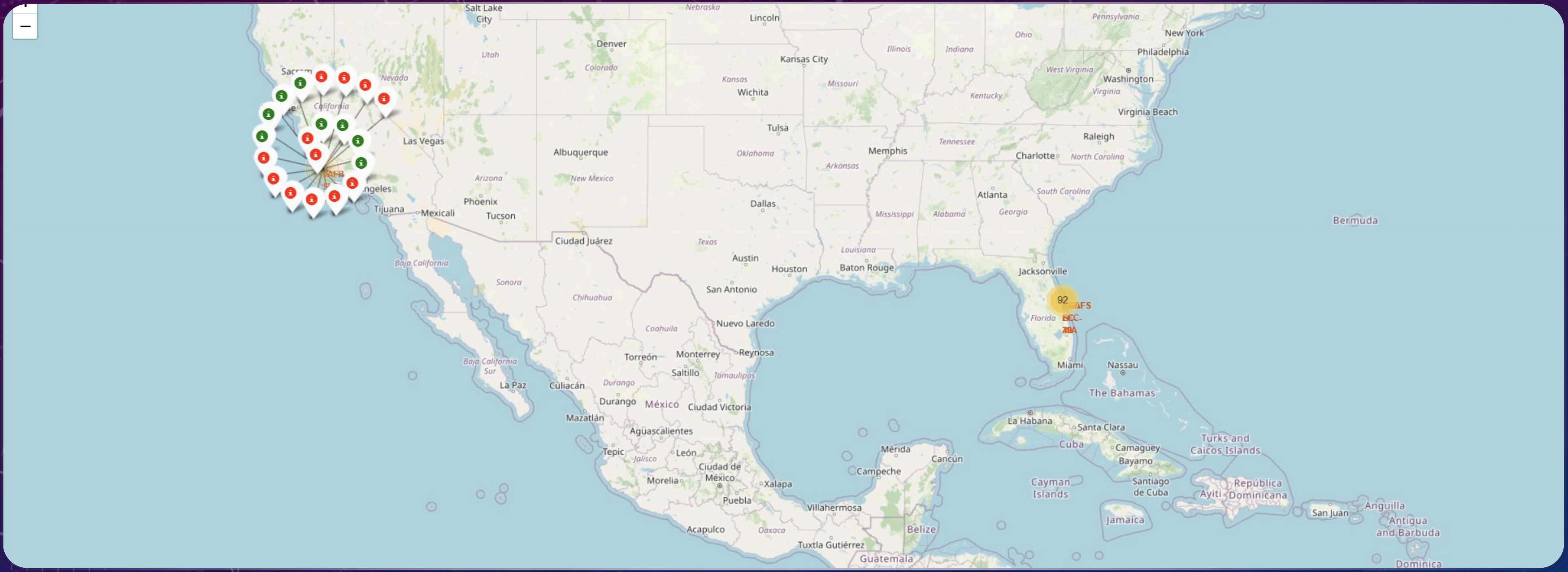
BUILD AN INTERACTIVE MAP WITH FOLIUM

- 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.
- Assigned the feature launch outcomes (failure or success) to class 0 for failure, and class 1 for success.
- Used the color-labeled marker clusters to identify which launch sites have a relatively high success rate.
- Calculated the distances between a launch site to its proximities.
- The link to the notebook is https://github.com/omsh97/IBM-Capstone-Project/blob/main/lab_jupyter_launch_site_location.ipynb



LAUNCH SITES GLOBAL MAP MARKERS

WE CAN SEE THE SPACEX LAUNCH SITES AT THE COAST OF FLORIDA AND CALIFORNIA



LAUNCH SITE MARKERS WITH COLOR LABELS

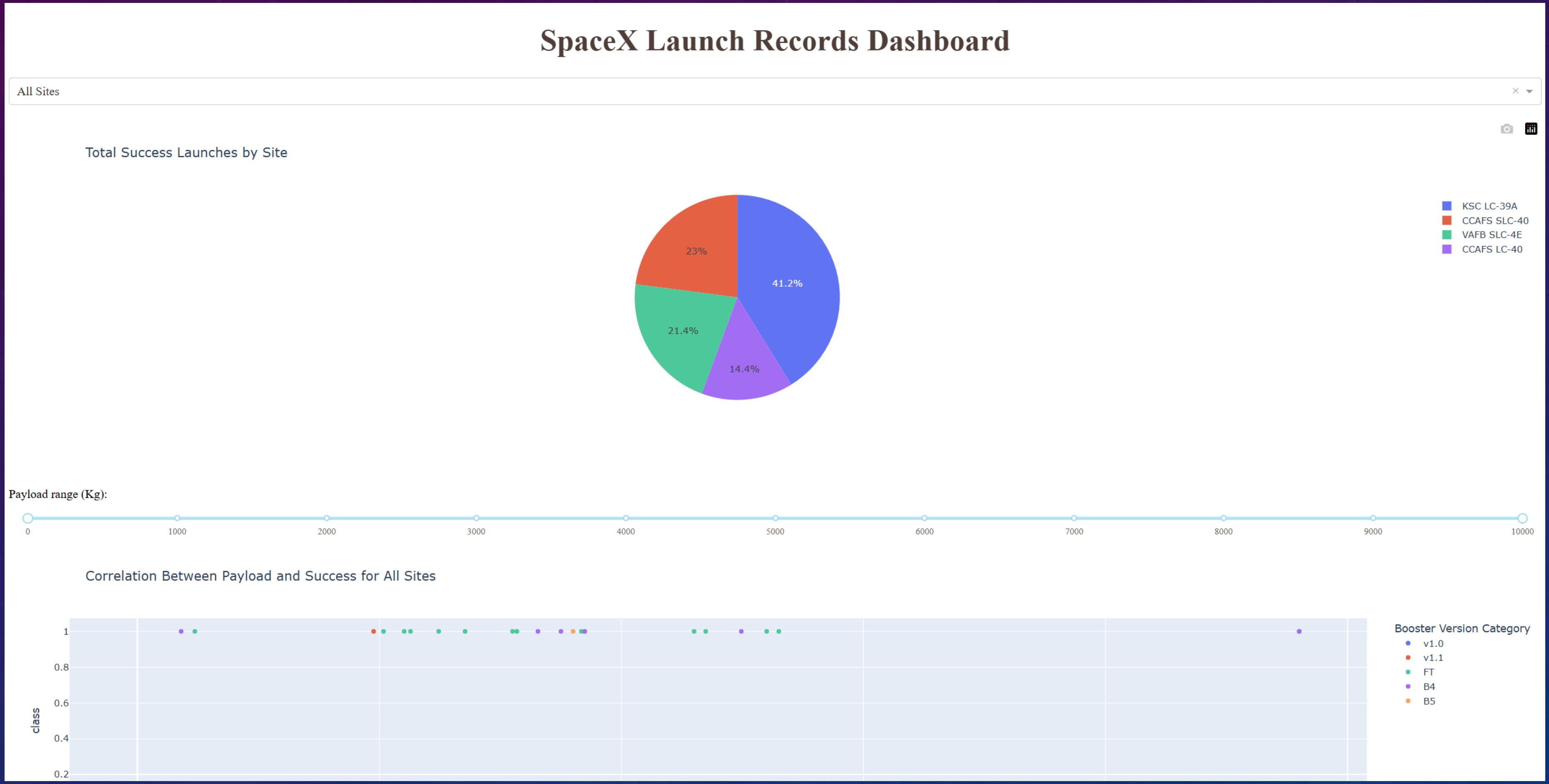
GREEN MARKERS – SUCCESSFUL LAUNCHES | RED MARKERS – FAILED LAUNCHES

BUILDING A PLOTLY DASHBOARD

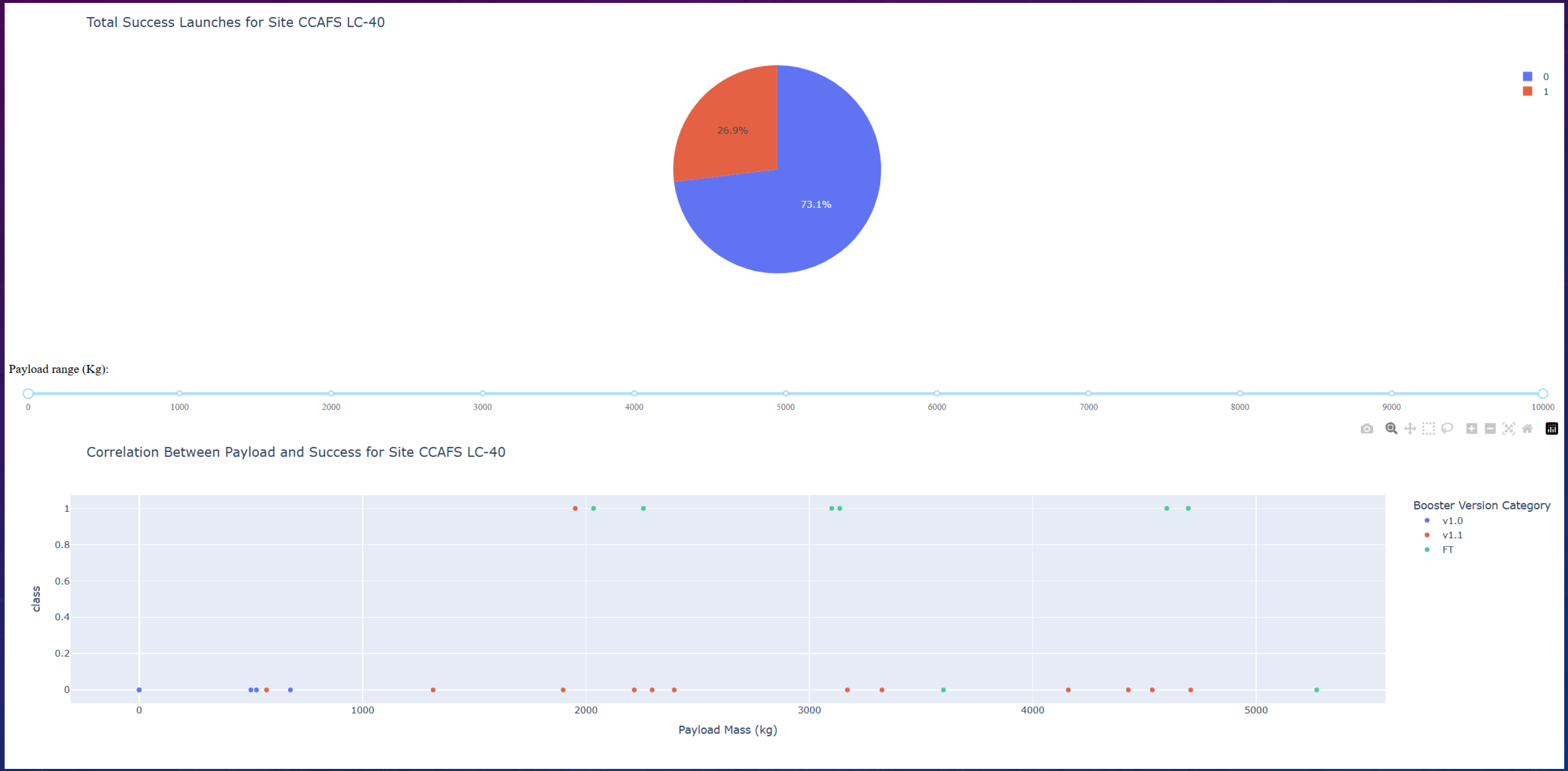
BUILD A DASHBOARD WITH PLOTLY DASH

- Built an interactive dashboard with Plotly dash using IBM Skills Network Labs
- Implemented dynamic pie charts plots showcasing the total launches by the selected sites
- Implemented dynamic scatter graphs with a slider showing the relationship with Outcome and Payload Mass (kg) for the different booster version.
- The link to the python app code is https://github.com/omsh97/IBM-Capstone-Project/blob/main/spacex-dash-app.py

PLOTLY DASHBOARD: HOME



PLOTLY DASHBOARD: CCAFS LC-40



PREDICTIVE ANALYSIS USING MACHINE LEARNING

PREDICTIVE ANALYSIS (CLASSIFICATION)

- Loaded the data using Numpy and Pandas, transformed the data and split it into training and testing data sets.
- Built different machine learning models and tuned different hyperparameters using GridSearchCV.
- Accuracy was used as the metric for selecting the best model.
- All models performed equally on the Test Data.
- The link to the notebook is https://github.com/omsh97/IBM-Capstone-Project/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

CLASSIFICATION ACCURACY

Surprisingly, all models return the same accuracy with Test Data

TASK 12

Find the method performs best:

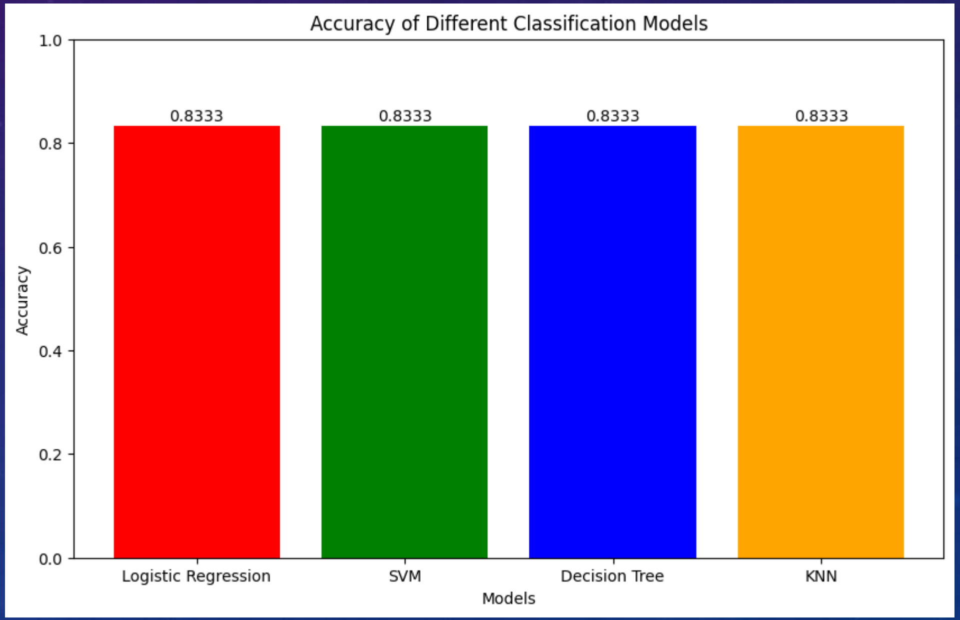
```python
models = ['Logistic Regression', 'SVM', 'Decision Tree', 'KNN']
accuracies = [logreg_accuracy, svm_accuracy, tree_accuracy, knn_accuracy]

plt.figure(figsize=(10, 6))
plt.bar(models, accuracies, color=['red', 'green', 'blue', 'orange'])

plt.title('Accuracy of Different Classification Models')
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.ylim(0, 1)  # Setting the y-axis range from 0 to 1

for i, accuracy in enumerate(accuracies):
    plt.text(i, accuracy + 0.01, f'{accuracy:.4f}', ha='center')

plt.show()
```



CLASSIFICATION ACCURACY

Decision Tree Classifier, however, returned the best results with Training Data

TASK 8

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
#     'max_features': ['auto', 'sqrt'],
     'max_features': ['sqrt', 'log2', None],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

```
tree_cv = GridSearchCV(tree, parameters, cv=10)
tree_cv.fit(X_train, Y_train)
```

```
GridSearchCV
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['sqrt', 'log2', None],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
estimator: DecisionTreeClassifier
DecisionTreeClassifier()
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'random'}
accuracy : 0.8732142857142857
```

CONCLUSIONS

- As the number of launches goes up, the rate of failure and therefore the probability of failure seems to decrease
- Launch success rate started to increase from 2013 until 2020.
- Launch Site CCAFS SLC 40 has the highest number of launches and the maximum number of failures as well.
- Orbits ES-L1, GEO, HEO, SSO had a success rate of 100%.
- KSC LC-39A had the most successful launches of any sites.
- All classifiers perform equally with current data.
- The Decision tree classifier might be the best machine learning algorithm for this task w.r.t training data accuracy.

THANK YOU