FALCON 9 BOOSTER LANDING SUCCESS PREDICTION

Capstone

Shahadat Hussain

Researcher, Mechanical Engineering

April, 2024

https://sites.google.com/view/shahadathussain/

IBM Developer

SKILLS NETWORK

OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Conclusion
- Appendix

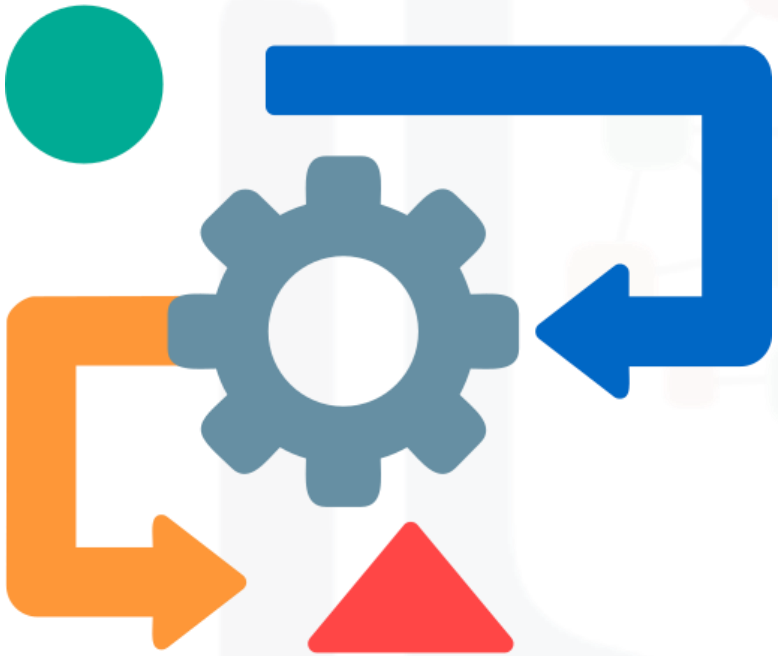
EXECUTIVE SUMMARY

- Analysis of space mission data reveals a positive trend of improving success rates over time.
- Key findings highlight:
  - Effectiveness of KSC LC-39A as a launch site.
  - Potential of CCAFS SLC-40 for handling heavier payloads.
  - Importance of strategic mission planning, evident in consistent success rates in specific orbits.
  - Observed increase in success rates beyond 2013 reflects advancements in technology and operational practices.
- Insights underscore the importance of continuous innovation and collaboration in achieving successful space missions.

INTRODUCTION

- The capstone project aims to predict the successful landing of the Falcon 9 first stage.
- SpaceX promotes Falcon 9 rocket launches on their website, priced at $62 million, significantly lower than competitors' costs of over $165 million, largely due to the reusable first stage.
- Predicting the first stage's landing success enables estimation of launch costs, crucial for competing bids against SpaceX.
- Using data science methods, variables like Payload Mass, Flight Number, Orbit Type, Launch Site location, and other factors were examined to gain insights into their behaviors and dependencies, particularly regarding landing success and mission success rates.

METHODOLOGY



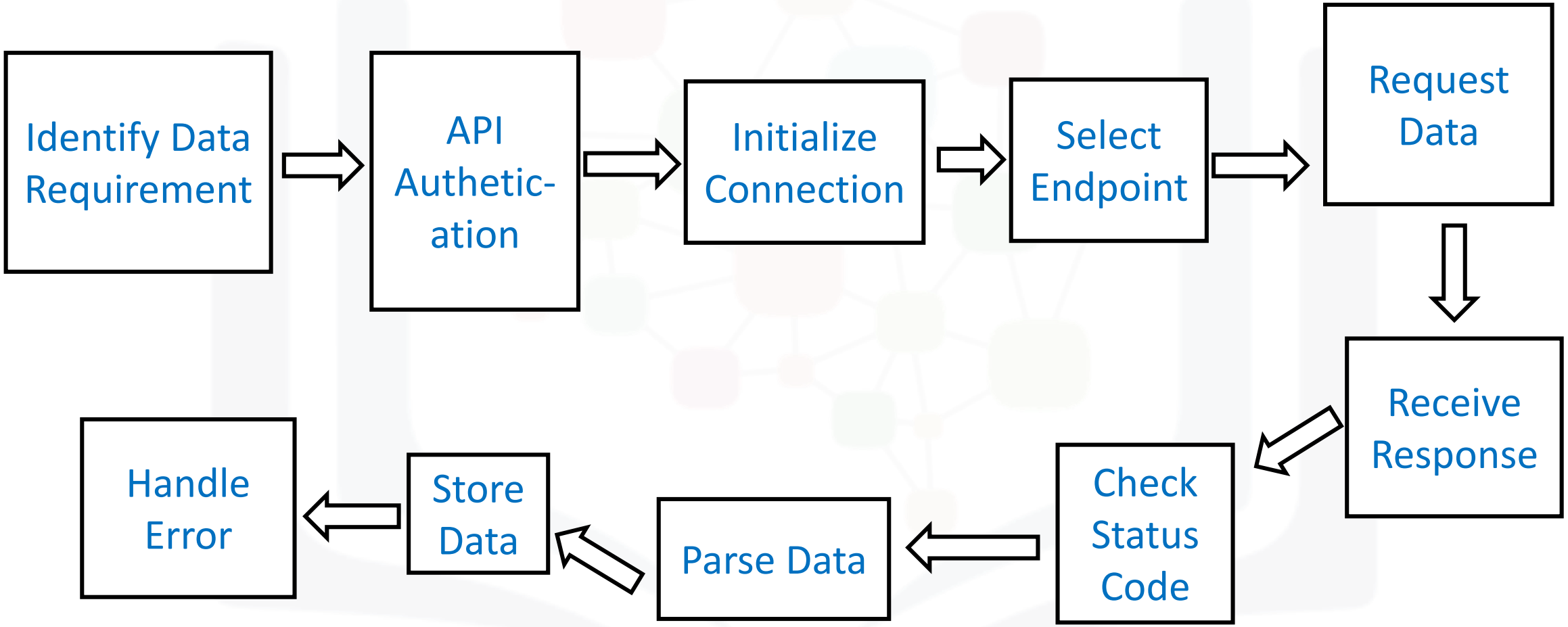
- Data Collection
  - APIs
  - Web scraping using BeautifulSoup library
- Data Wrangling
- Exploratory Data Analysis
  - Pandas, matplotlib, numpy, and scipy
  - SQL
- Interactive Visual Analytics
  - Using Folium map library
  - Dashboard using Plotly Dash
- Prediction Models
  - SVM
  - Classification Trees
  - Regression

DATA COLLECTION THROUGH SPACEX API

- Data collection through an API involves retrieving structured data from a remote server using predefined endpoints and protocols.
- Key steps include accessing the API, sending a request with parameters specifying the desired data, receiving and parsing the response, and storing the data for further analysis.
- APIs provide advantages such as real-time data access, automation of retrieval processes, and interoperability, but users must adhere to terms of use and respect usage limits.

https://github.com/drdataengg/Capstone_Applied_Data_Science_IBM/blob/05b10240338f1652fa38dbd7824a0e0b83e9285c/jupyter-labs-spacex-data-collection-api.ipynb

DATA COLLECTION THROUGH SPACEX API



DATA COLLECTION THROUGH SPACEX API

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

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

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

```
response.status_code
```

```
0]: 200
```

```
# Use json_normalize meethod to conver
response_json = response.json()
df = pd.json_normalize(response_json)
```

Using the dataframe `data` print the first 5 rows

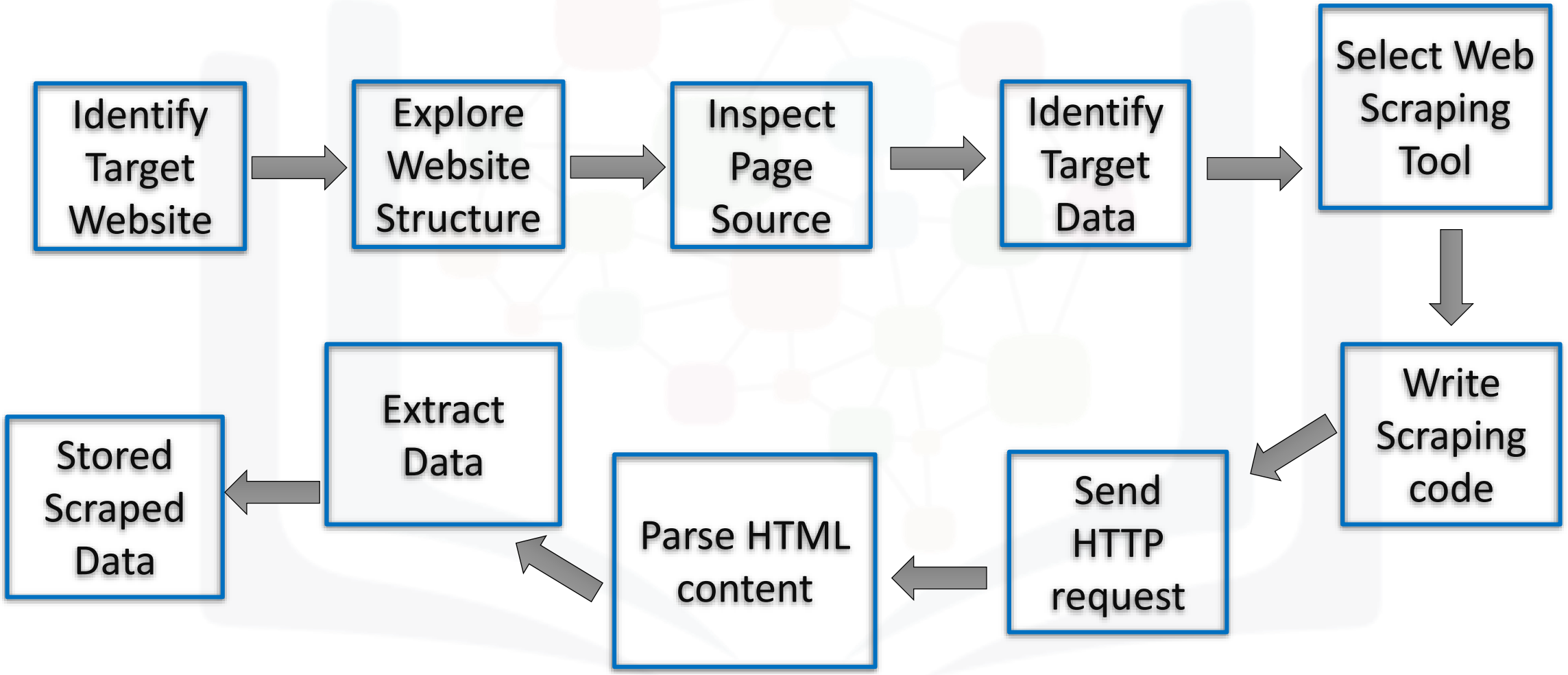
```
# Get the head of the dataframe
df.head()
```

DATA COLLECTION WITH WEB SCRAPING

- Web scraping is a method used to automatically extract data from websites.
- It involves accessing web pages programmatically, parsing their HTML or other markup languages, and extracting desired information.
- Techniques include using libraries or frameworks in programming languages like Python or JavaScript, parsing HTML, automating data collection, and adhering to legal and ethical guidelines.

https://github.com/drdataengg/Capstone_Applied_Data_Science_IBM/blob/05b10240338f1652fa38dbd7824a0e0b83e9285c/jupyter-labs-webscraping.ipynb

DATA COLLECTION WITH WEB SCRAPING



DATA COLLECTION WITH WEB SCRAPING

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)
response_text=response.text
```

```
# Use BeautifulSoup() to create a BeautifulSoup c
import html5lib
soup=BeautifulSoup(response_text, 'lxml')
```

Print the page title to verify if the `BeautifulSoup` object was cr

```
# Use soup.title attribute
soup.title
```

```
]: <title>List of Falcon 9 and Falcon Heavy launches
```

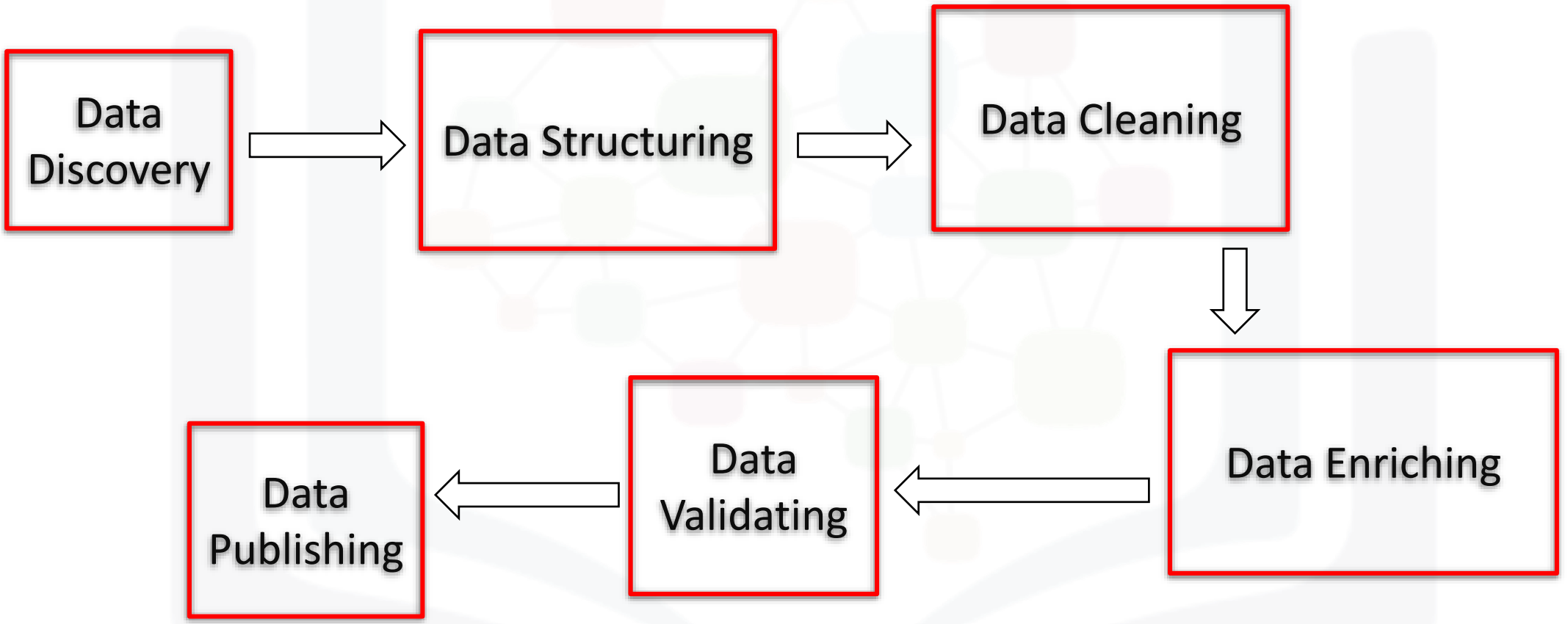
```
# Use the find_all function in the Beautiful
# Assign the result to a list called `html_t
html_tables = soup.find_all('table')
```

DATA WRANGLING

- Data wrangling, also known as data munging, involves cleaning, transforming, and preparing raw data for analysis.
- It encompasses steps such as data collection, cleaning, transformation, integration, feature engineering, and data splitting.
- Data wrangling ensures that the data is accurate, complete, and structured in a way that facilitates meaningful analysis and modeling.

https://github.com/drdataengg/Capstone_Applied_Data_Science_IBM/blob/05b10240338f1652fa38dbd7824a0e0b83e9285c/labs-jupyter-spacex-Data%20wrangling.ipynb

DATA WRANGLING



DATA WRANGLING

```
n [3]: df.isnull().sum()/len(df)*100

Out[3]: FlightNumber      0.000000
        Date              0.000000
        BoosterVersion    0.000000
        PayloadMass       0.000000
        Orbit             0.000000
        LaunchSite        0.000000
        Outcome           0.000000
        Flights           0.000000
        GridFins          0.000000
        Reused            0.000000
        Legs              0.000000
        LandingPad       28.888889
        Block             0.000000
        ReusedCount       0.000000
        Serial            0.000000
        Longitude         0.000000
        Latitude          0.000000
        dtype: float64
```

Identify which columns are numerical and

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

```
# Apply value_counts() on column LaunchSite
launch_counts = df['LaunchSite'].value_counts()
launch_counts
```

```
]: LaunchSite
   CCAFS SLC 40    55
   KSC LC 39A      22
   VAFB SLC 4E     13
   Name: count, dtype: int64
```

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise


landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]


print("Landing class:")
print(landing_class)
```

```
Landing class:
[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1]
```

IBM Developer

SKILLS NETWORK

EDA WITH DATA VISUALIZATION

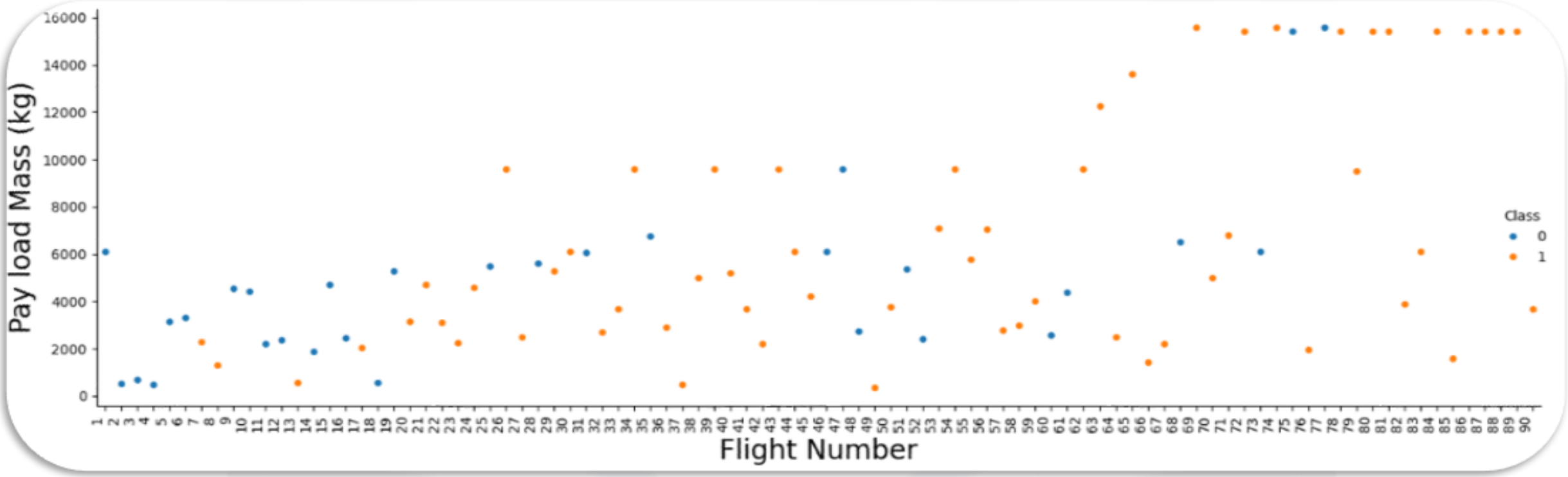
Exploratory data analysis was conducted on the extracted SpaceX data to uncover patterns. Key libraries including pandas (for data manipulation), matplotlib (a versatile plotting library), seaborn (a visualization library built on matplotlib), and numpy (for array support) were imported.

Generated multiple plots to explore the dataset:

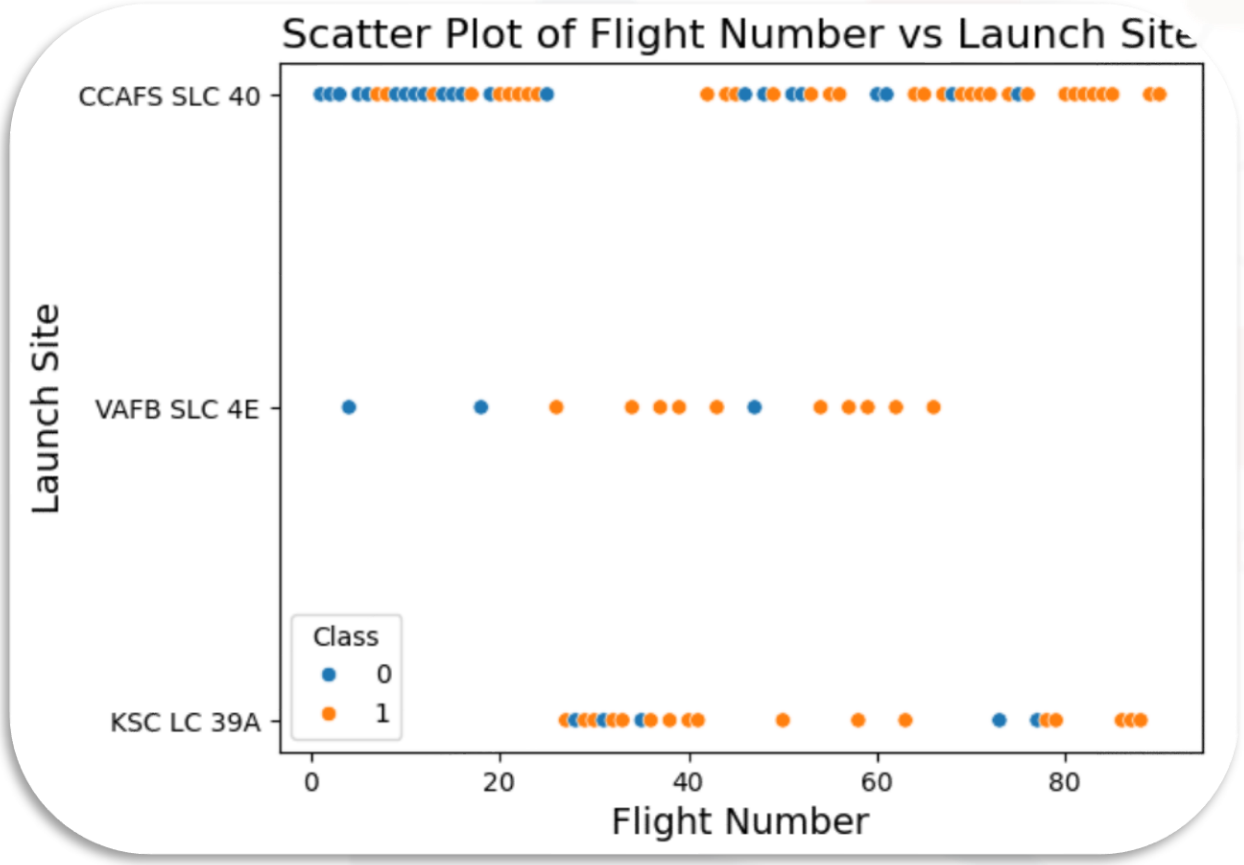
- Categorical plots comparing payload mass with flight number.
- Scatterplots depicting the relationship between payload mass and launch site.
- Scatterplots showing orbit versus flight number and orbit versus payload mass.
- Created a line plot to visualize the yearly launch success rate.

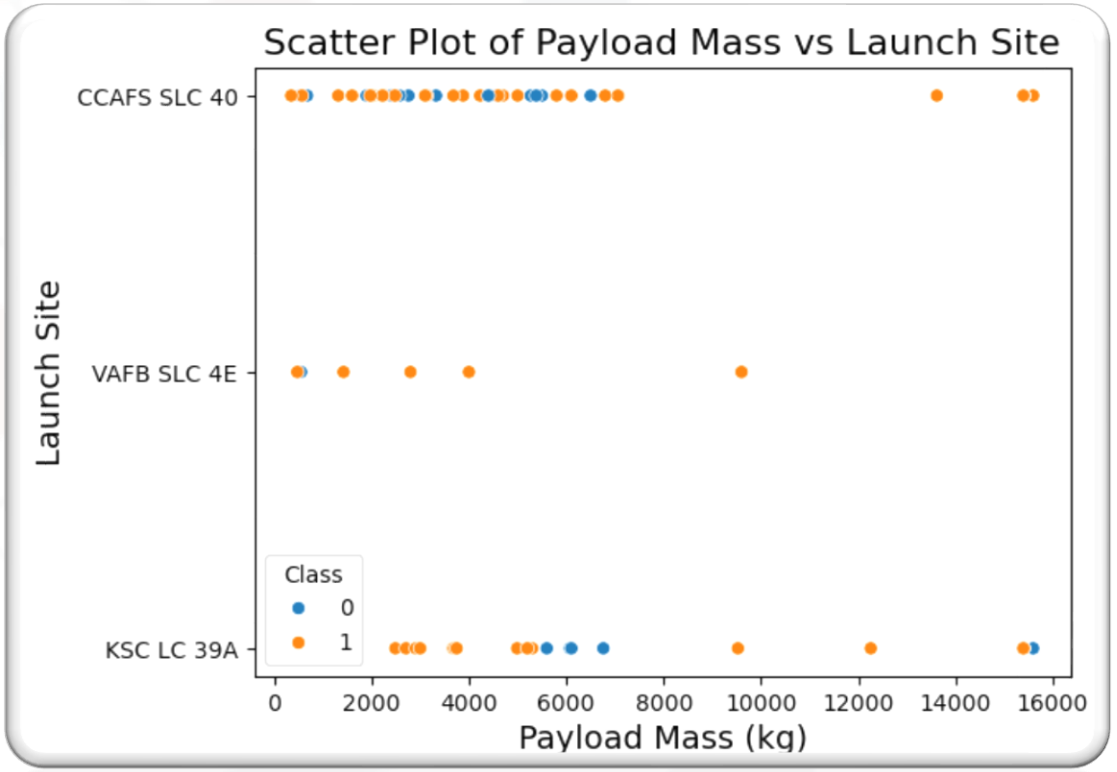
https://github.com/drdataengg/Capstone_Applied_Data_Science_IBM/blob/05b10240338f1652fa38dbd7824a0e0b83e9285c/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

PAYLOAD MASS VS FLIGHT NUMBER

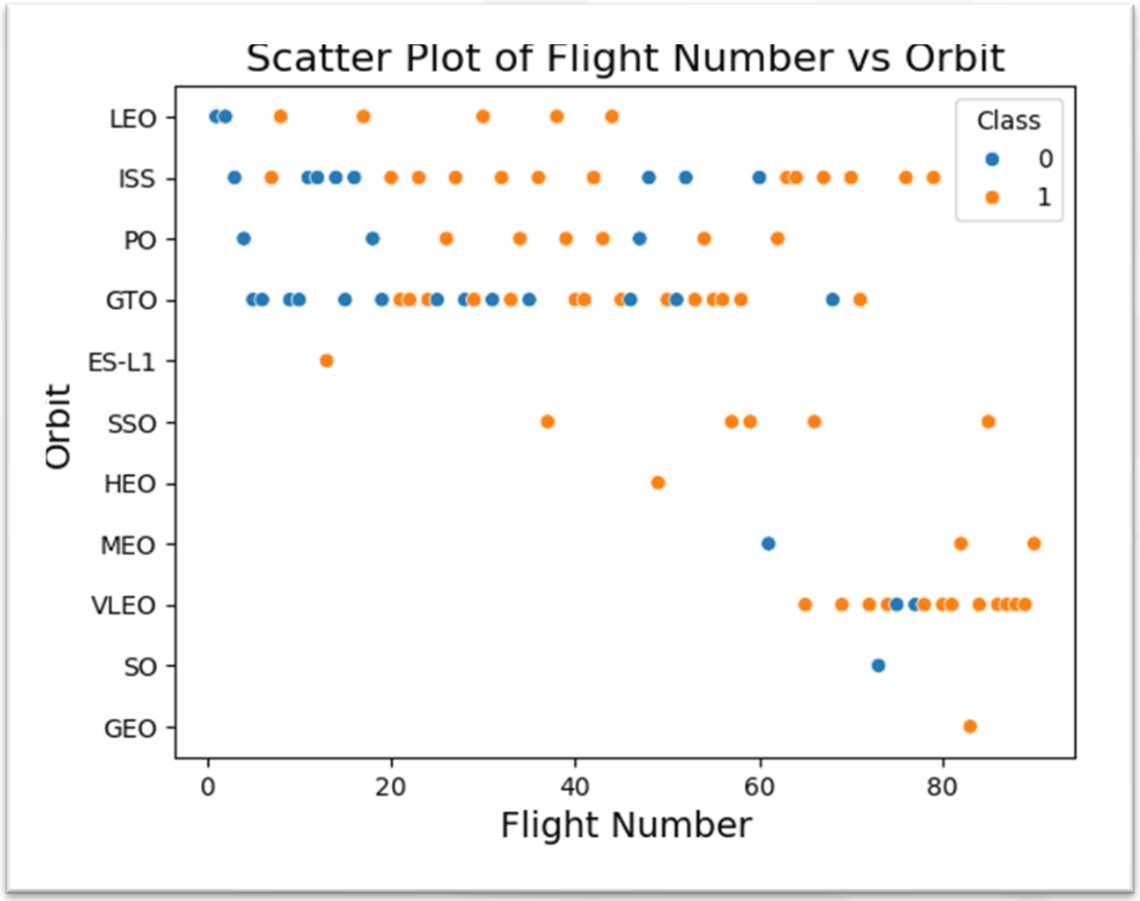


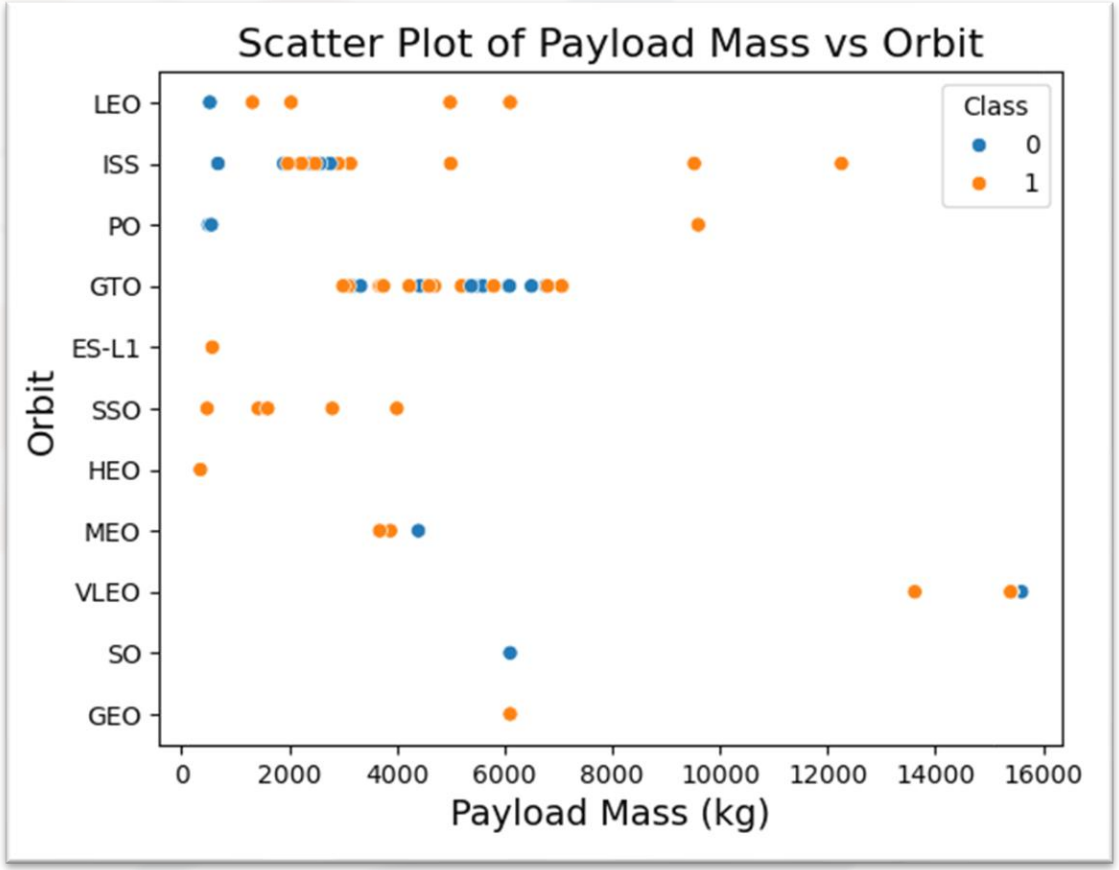
LAUNCH SITE VS FLIGHT NUMBER AND PAYLOAD MASS



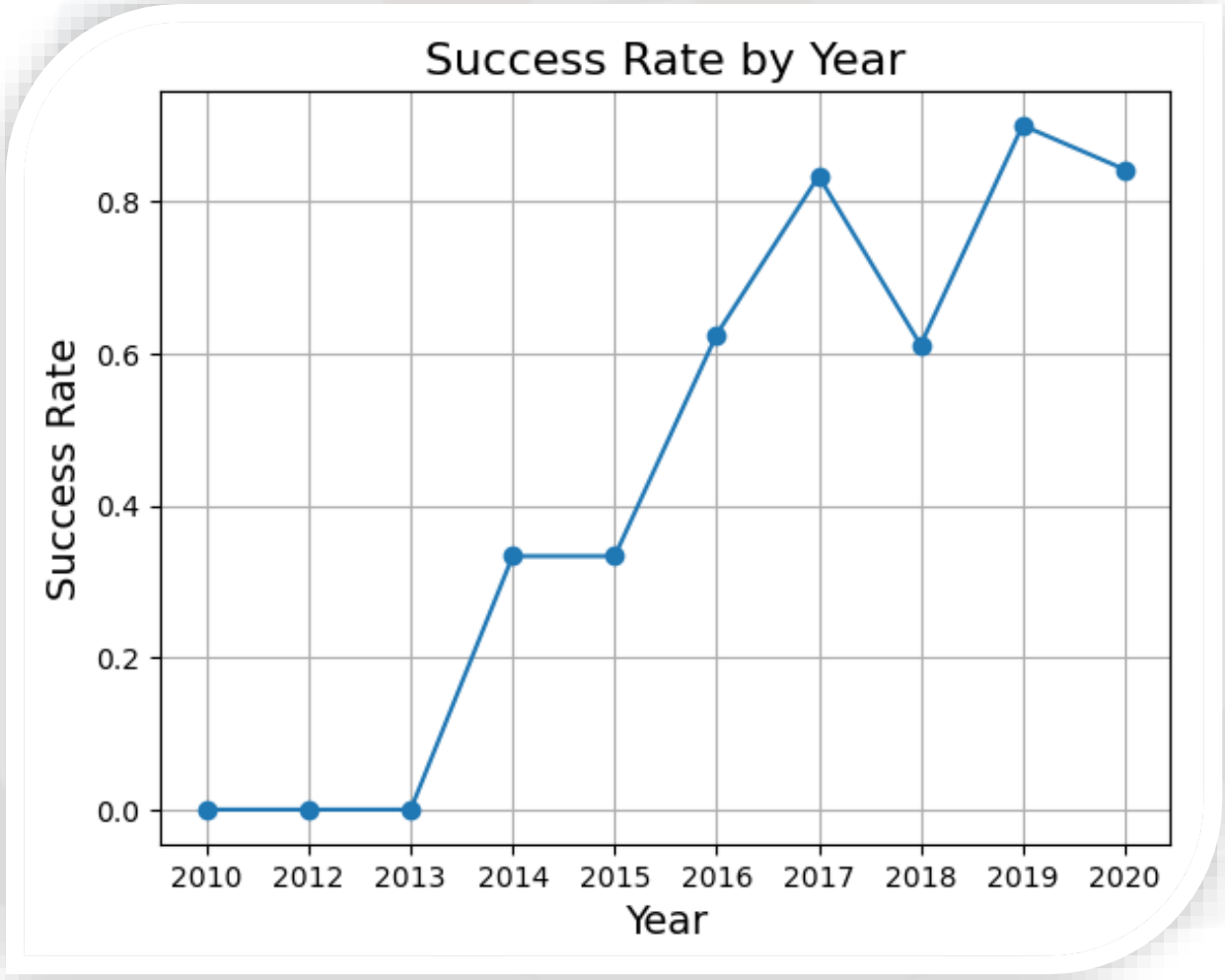


ORBIT TYPE VS FLIGHT NUMBER AND PAYLOAD MASS





YEARLY SUCCESS RATE



EDA WITH SQL

Exploratory Data Analysis (EDA) with SQL involves querying a database to explore and analyze datasets directly. This includes describing the data, sampling, handling missing values, exploring relationships, identifying patterns, and, while not its primary function, some basic visualization capabilities. EDA with SQL is an efficient way to gain insights into large datasets within the database environment.

- Conducted EDA with SQL on SpaceX dataset.
- Tasks included showcasing unique launch site names and retrieving records with specific launch site strings.
- Calculated total payload mass for NASA's CRS program and average payload mass for booster version F9 v1.1.
- Identified date of first successful ground pad landing outcome.
- Listed boosters with successful drone ship landings within specified payload mass range.
- Tabulated total successful and failed mission outcomes.
- Identified booster versions with maximum payload mass using subquery.
- Compiled records to display month names, failure landing outcomes, booster versions, and launch sites for 2015.
- Ranked count of landing outcomes between specified dates.

EDA WITH SQL

```
[10]: %sql Select distinct Launch_Site from SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

Out[10]:

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

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

```
%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5
 * sqlite:///my_data1.db
Done.
```

5]:

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

https://github.com/drdataengg/Capstone_Applied_Data_Science_IBM/blob/d2e55cc601f0090e851ec66aeb9cce954f0962d5/jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA WITH SQL

Task 1

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

```
[10]: %sql Select distinct Launch_Site from SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

Out[10]:

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

Task 2

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

```
]: %sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5
 * sqlite:///my_data1.db
Done.
```

t[26]:

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

EDA WITH SQL

Task 3

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

```
]:  %%sql

    SELECT SUM(PAYLOAD_MASS__KG_) AS TotalPayloadMass
    FROM SPACEXTABLE
    WHERE Customer LIKE '%CRS%';
```

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

t[14]:

TotalPayloadMass
48213

Task 4

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

```
%%sql

SELECT AVG(PAYLOAD_MASS__KG_) AS AveragePayloadMass
FROM SPACEXTABLE
WHERE Booster_Version = 'F9 v1.1';
```

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

l5]:

AveragePayloadMass
2928.4

EDA WITH SQL

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 FirstSuccessfulLandingDate
FROM SPACEXTABLE
WHERE Landing_Outcome LIKE 'Success%Ground Pad%';
```

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

[18]:

FirstSuccessfulLandingDate
2015-12-22

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 DISTINCT Booster_Version
FROM SPACEXTABLE
WHERE Landing_Outcome LIKE 'Success%Drone Ship%'
AND PAYLOAD_MASS__KG_ > 4000
AND PAYLOAD_MASS__KG_ < 6000;
```

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

]:

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

EDA WITH SQL

Task 7

List the total number of successful and failure mission outcomes

```
%%sql

SELECT Mission_Outcome, COUNT(*) AS TotalCount
FROM SPACEXTABLE
WHERE Mission_Outcome LIKE 'Success%' OR Mission_Outcome LIKE 'Failure%'
GROUP BY Mission_Outcome;
```

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

2]:

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

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%%sql

SELECT Booster_Version
FROM SPACEXTABLE
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
```

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

23]:

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4

EDA WITH SQL

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,boos
2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to ge

```
%%sql

SELECT
    CASE substr(Date, 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
```

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

MonthName	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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 LandingOutcomeCount
FROM SPACEXTABLE
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY LandingOutcomeCount DESC;
```

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

5]:

Landing_Outcome	LandingOutcomeCount
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 VISUAL ANALYTICS: FOLIUM MAP

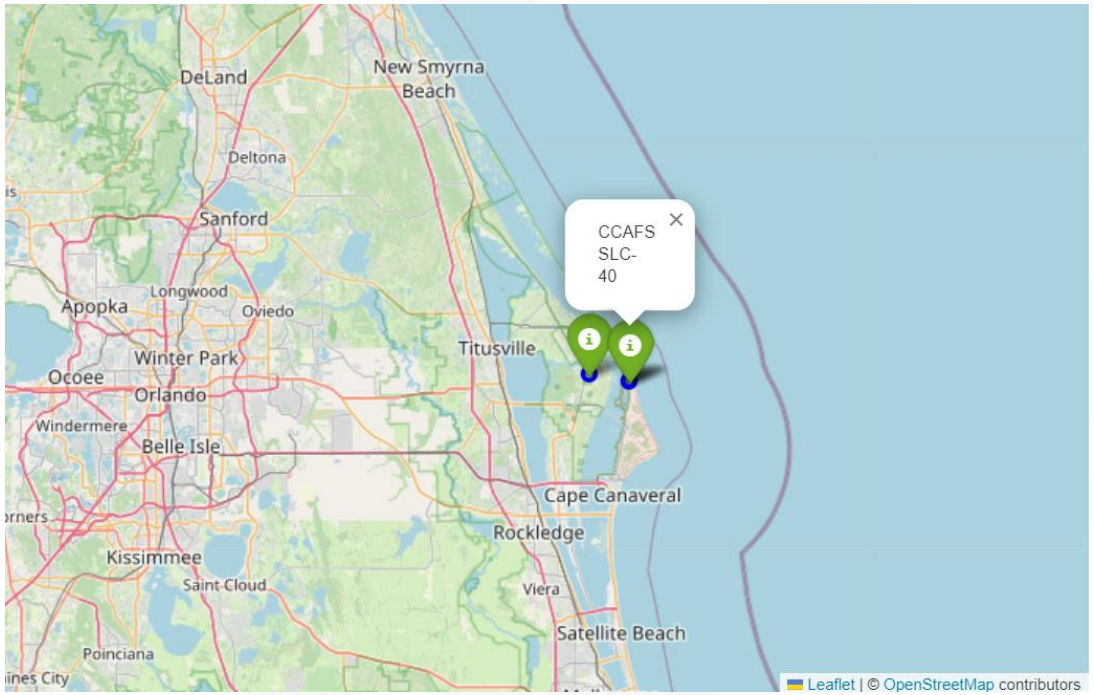
Folium enables interactive visual analytics in Python by creating dynamic maps for geospatial data exploration. Users can plot data points, customize map styles, and interact with map elements like markers and polygons. Integration of visualizations and widgets allows for deeper insights into spatial patterns and relationships, enhancing geospatial data analysis.

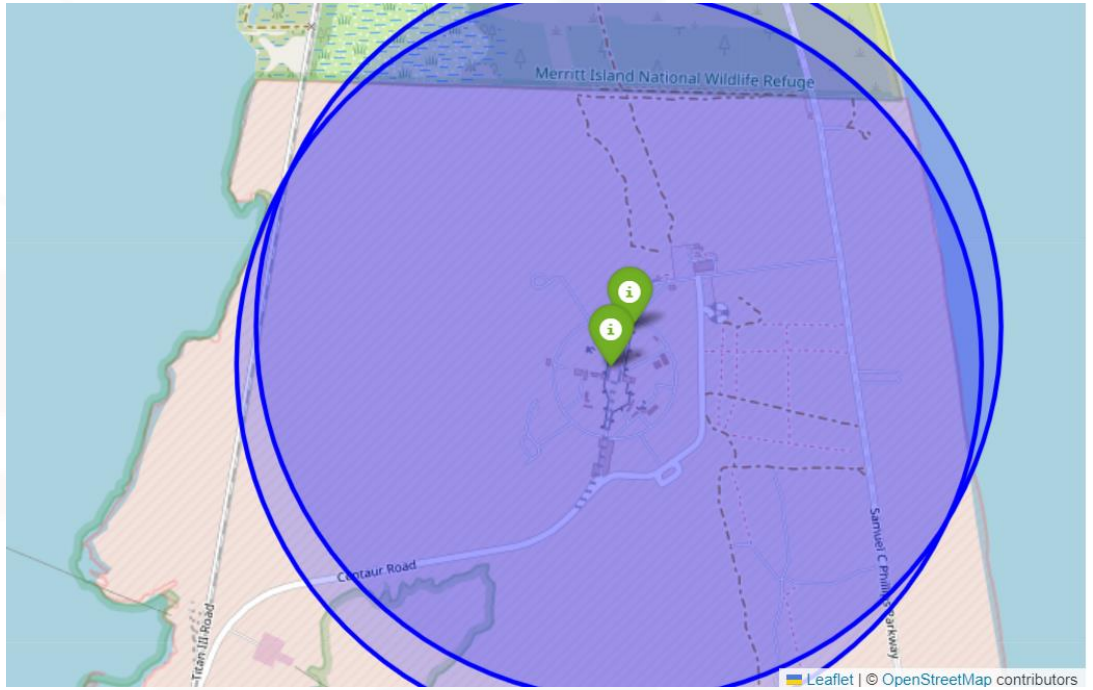
Using Folium maps, the following actions were performed to discover geographical patterns related to launch sites:

- Marking all launch sites on a map.
- Indicating the success or failure of launches for each site on the map.
- Calculating the distances between a launch site and its proximities.

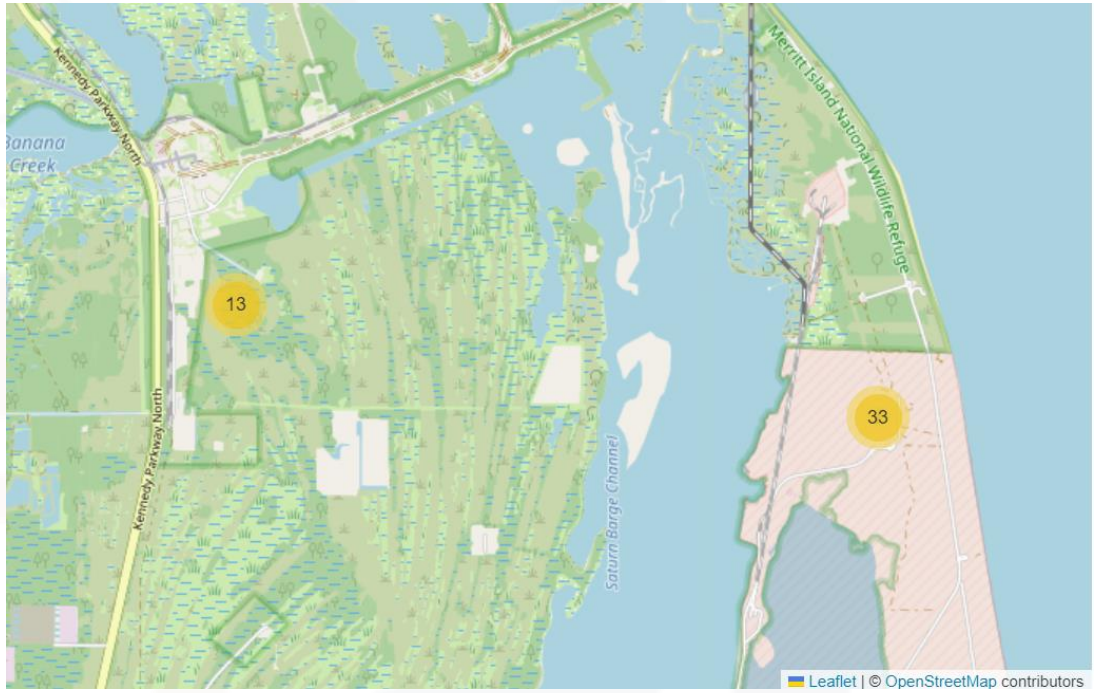
https://github.com/drdataengg/Capstone_Applied_Data_Science_IBM/blob/d2e55cc601f0090e851ec66aeb9cce954f0962d5/lab_jupyter_launch_site_location%20(1).ipynb

LAUNCH SITES MARKED WITH MARKERS AND CIRCLE

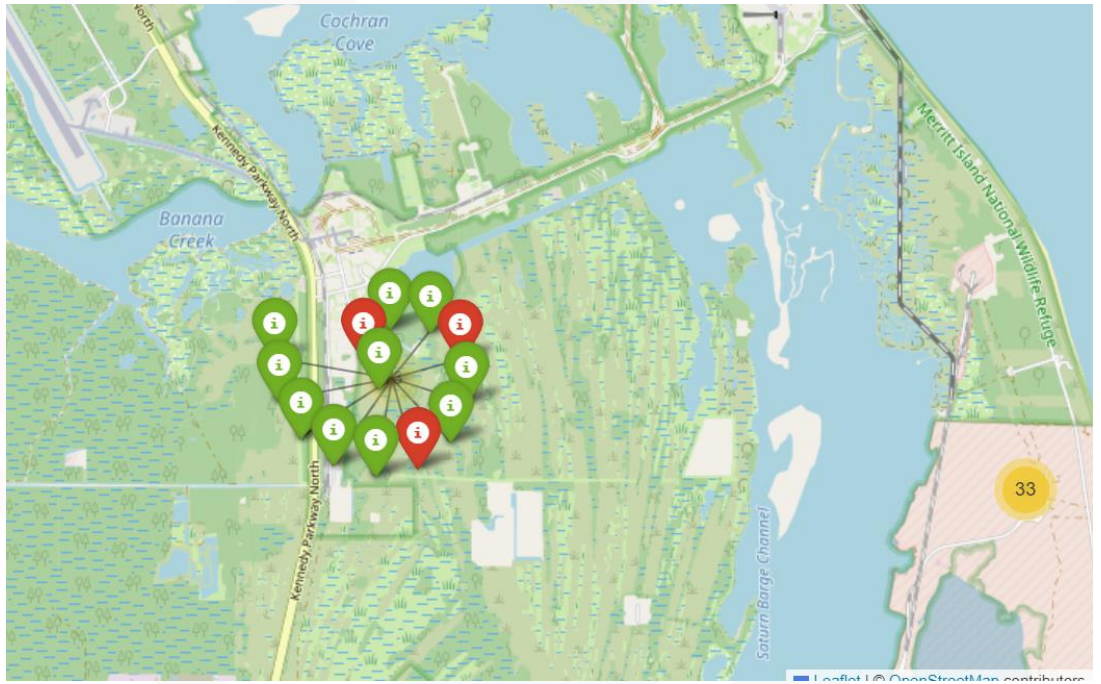




TOTAL NUMBER OF LAUNCHES PER SITE

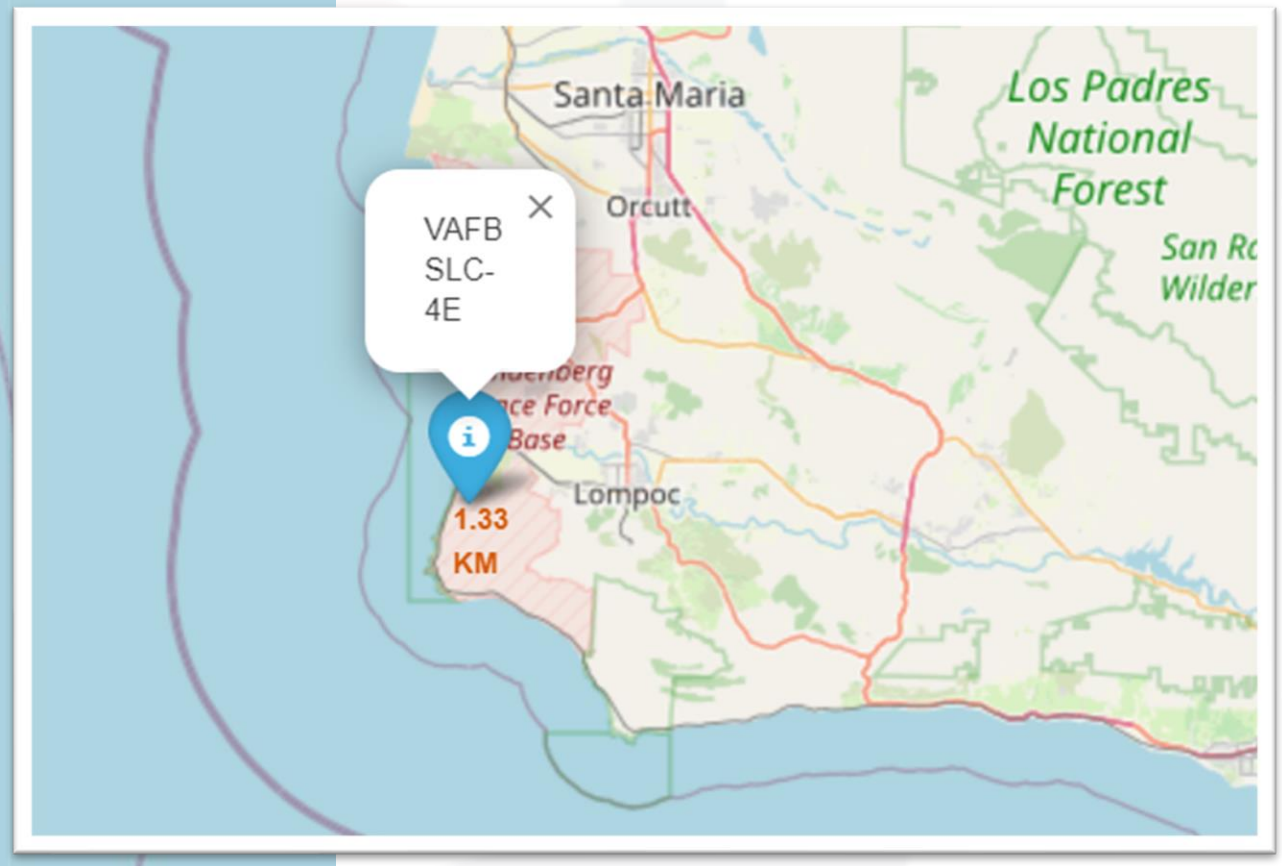


SUCCESSFUL AND FAILED LAUNCHED MARKED WITH MARKER OF DIFFERENT COLORS



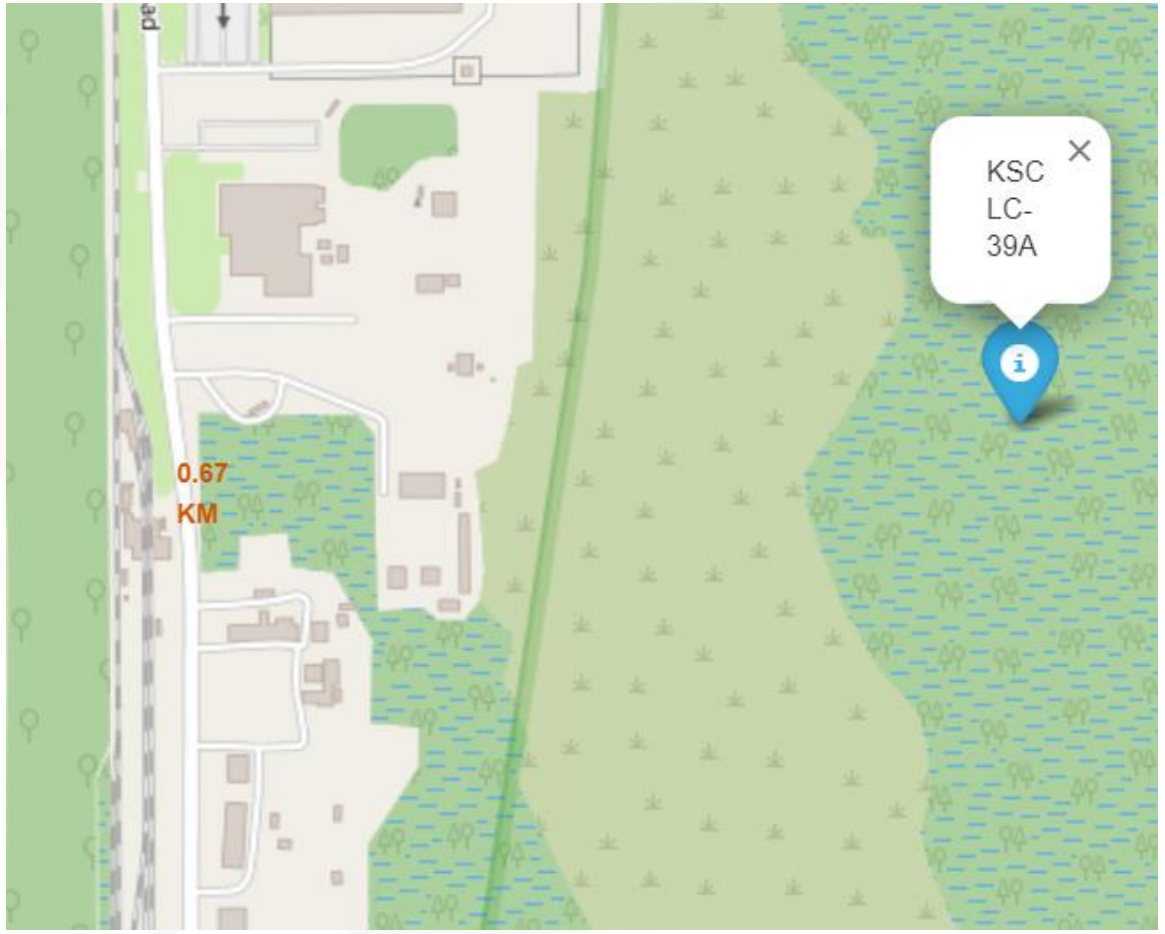
SUCCESSFUL AND FAILED LAUNCHED MARKED WITH MARKER OF DIFFERENT COLORS AND POPUP LABELS WITH LAUNCH SITE NAME

DISTANCE BETWEEN LAUNCH SITES AND NEAREST COASTLINE

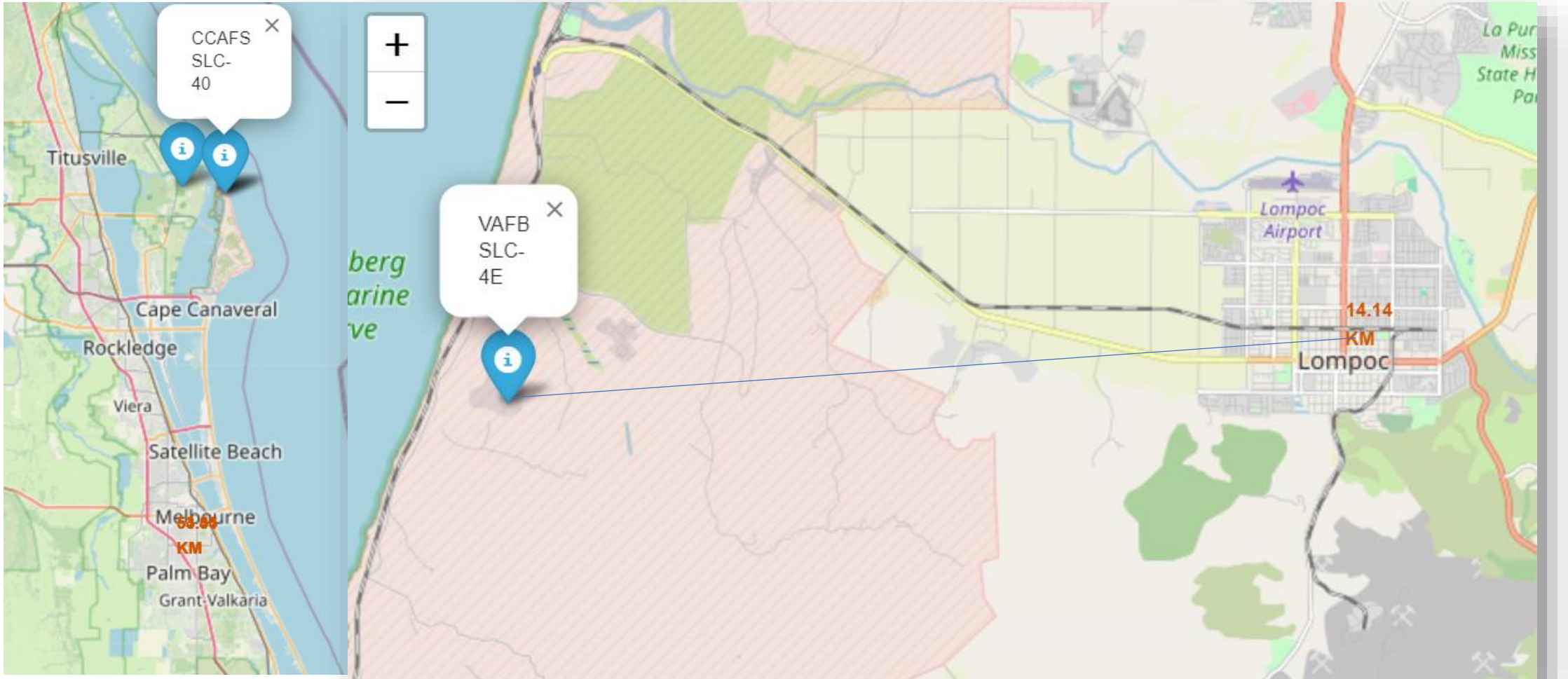


DISTANCE BETWEEN LAUNCH SITES AND NEAREST RAILWAY

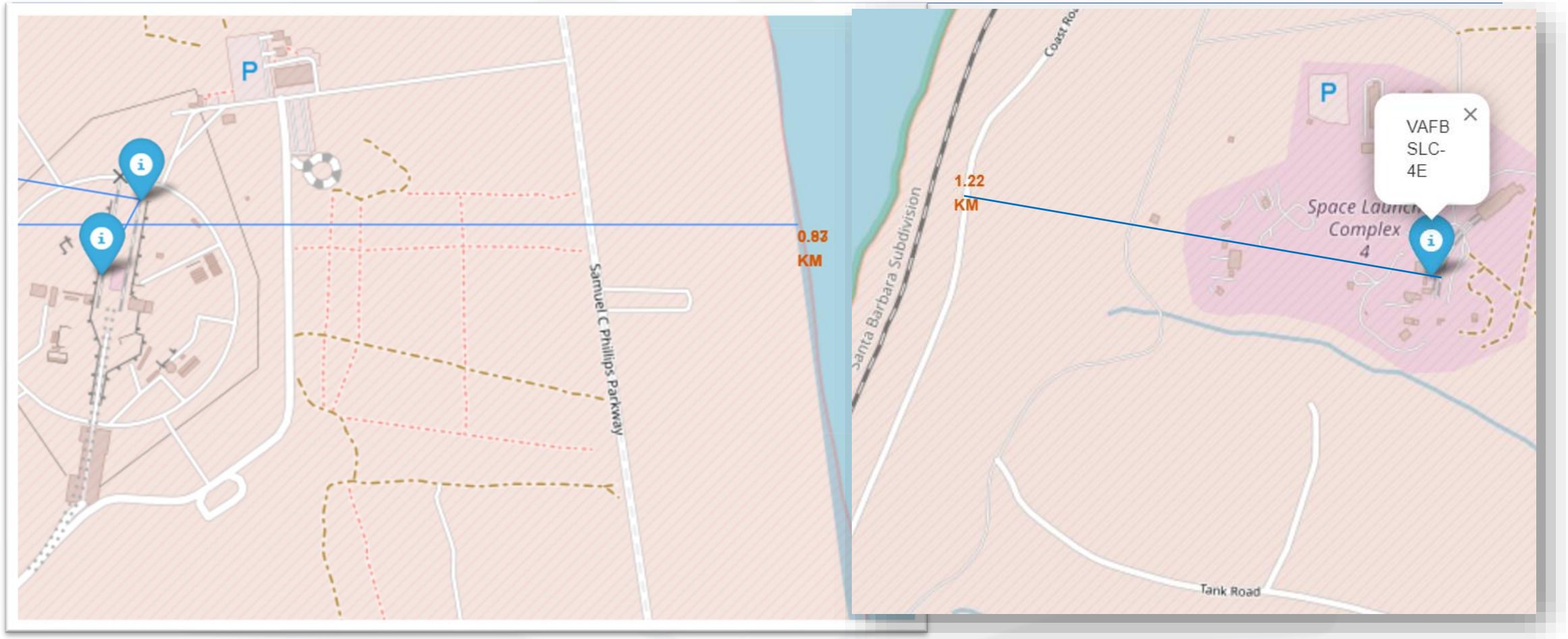
DISTANCE BETWEEN LAUNCH SITES AND NEAREST HIGHWAY



DISTANCE BETWEEN LAUNCH SITES AND NEAREST CITY AREA



DISTANCE BETWEEN LAUNCH SITES AND NEAREST COAST WITH POLYLINE



INTERACTIVE VISUAL ANALYTICS: PLOTLY DASH

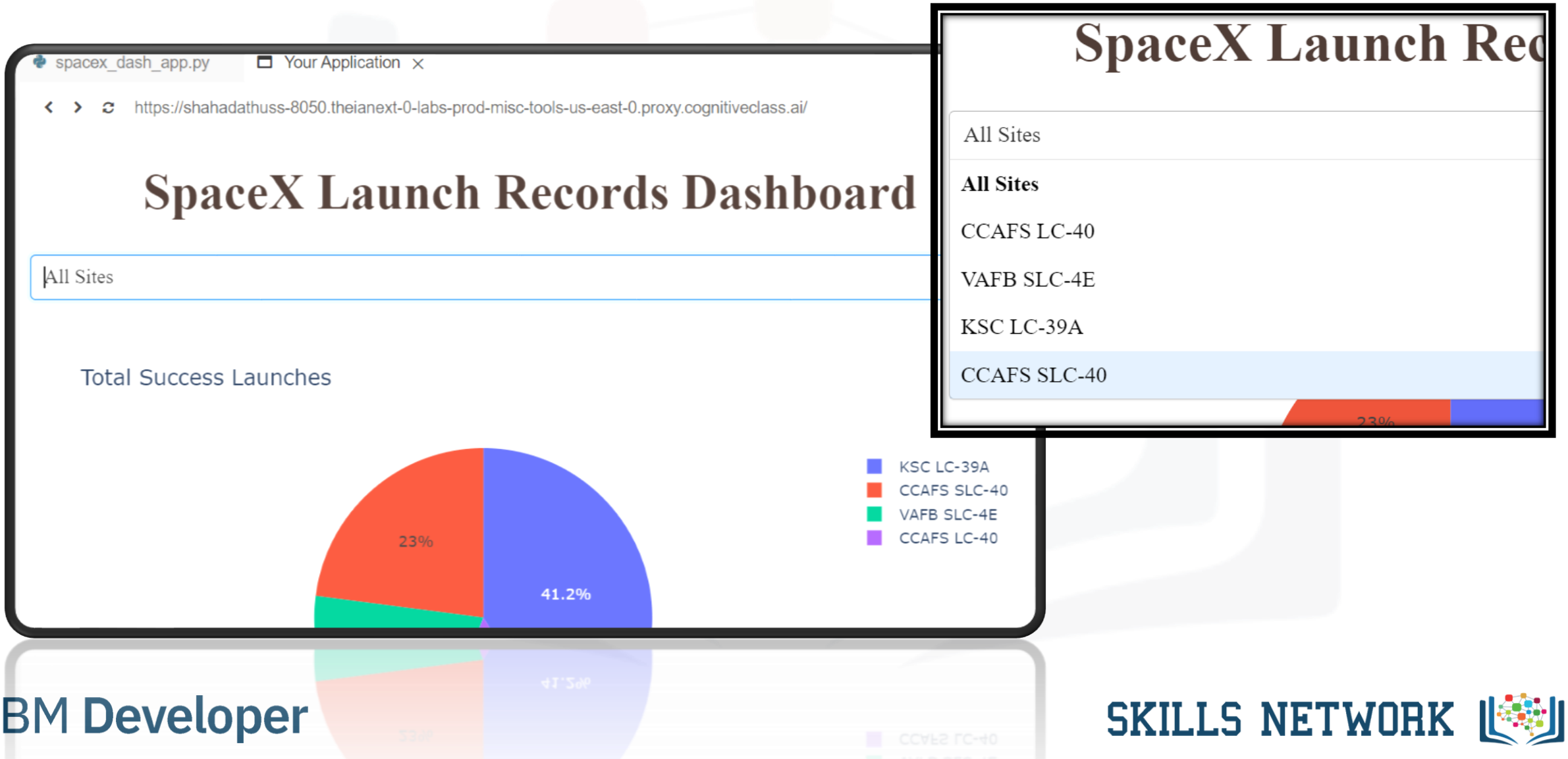
Dash and Plotly Express empower users to create interactive web-based dashboards in Python without the need for additional web development skills. These dashboards feature interactive components like dropdowns and sliders, responsive layouts, and customizable styling.

A Plotly Dash application was created for interactive visual analytics on real-time SpaceX launch data.

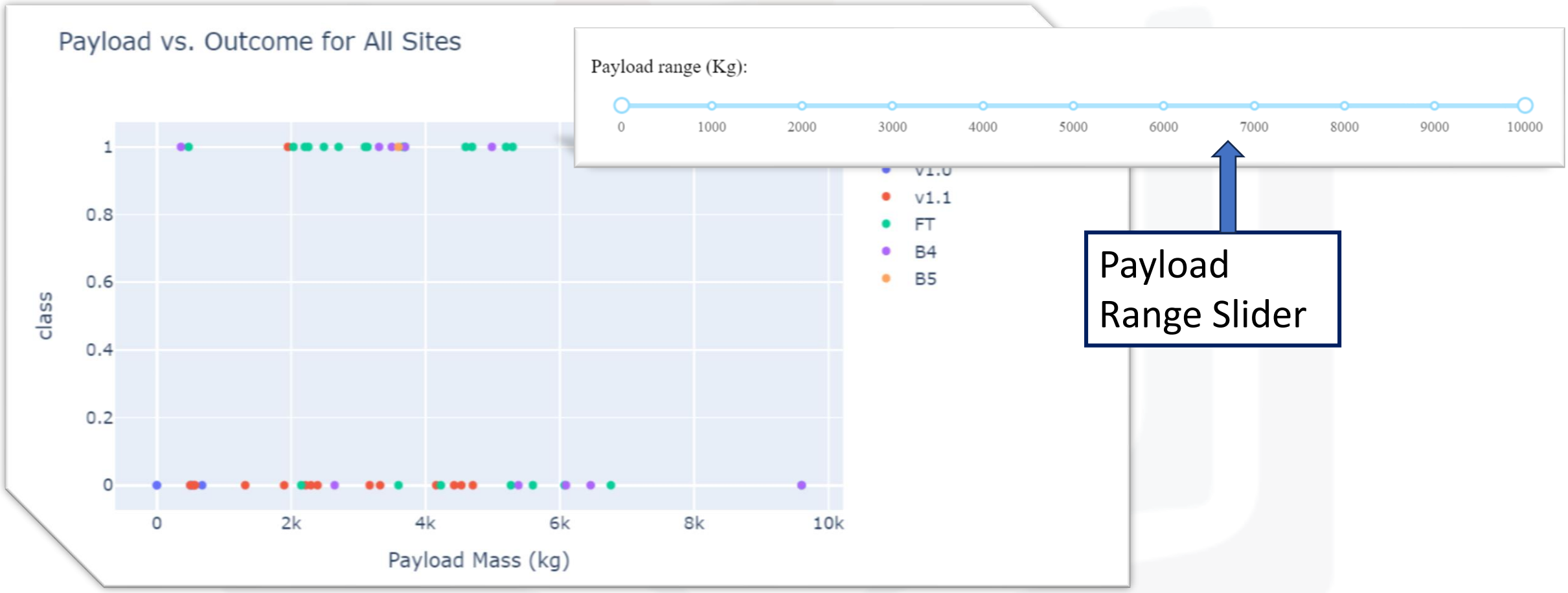
- Input components like a dropdown list and range slider were included to interact with visualizations.
- Tasks involved adding a launch site dropdown and range slider, and implementing callback functions for rendering pie and scatter plots.
- The dashboard enables users to explore SpaceX launch data dynamically through interactive elements.
- The tasks guided participants through building various components of the dashboard for comprehensive data analysis.

https://github.com/drdataengg/Capstone_Applied_Data_Science_IBM/blob/082882e665b1711d0e0ac46fcc36b2a4bf52a37c/spacex_dash_app.py

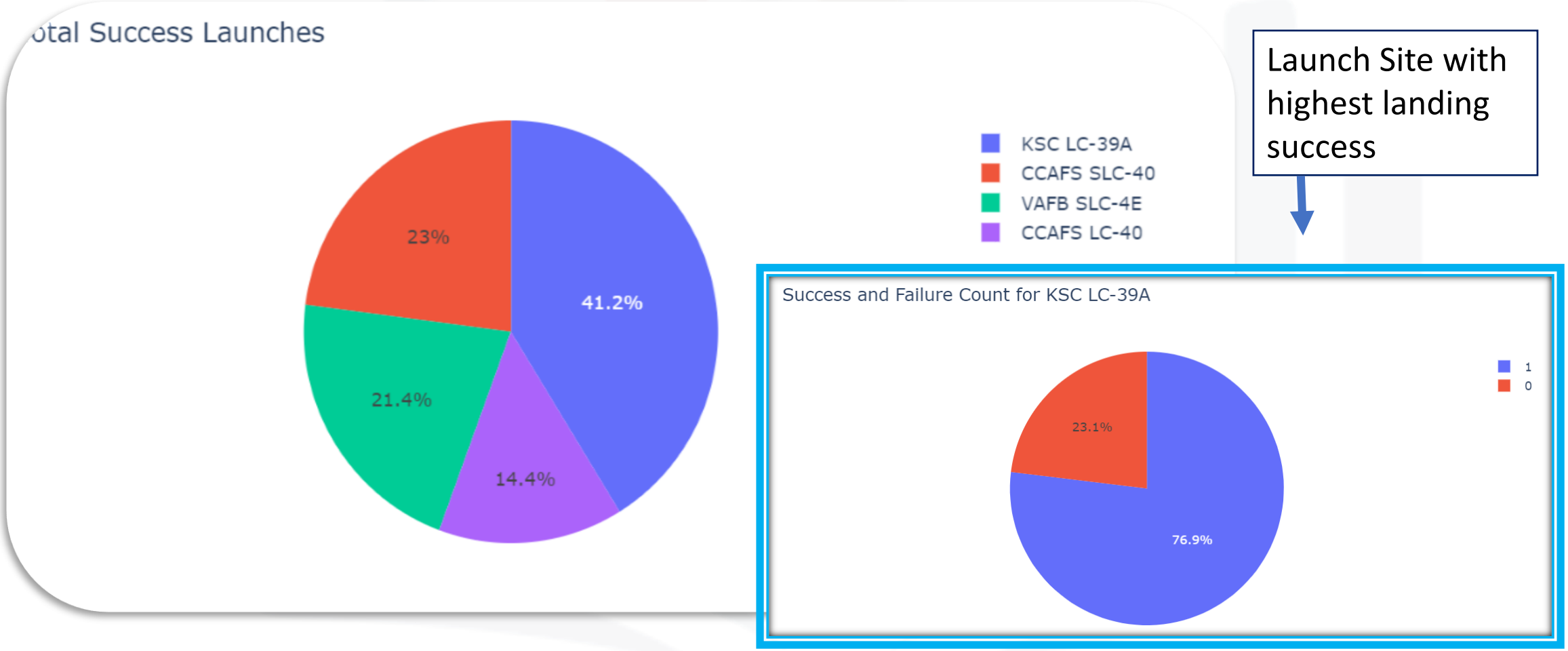
DASHBOARD TAB 1



DASHBOARD TAB 2



DASHBOARD TAB 3



MACHINE LEARNING PREDICTION MODELS

Machine learning prediction models utilize historical data to identify patterns and make predictions about future outcomes or classify new data points. They are applied across diverse domains for tasks like stock price forecasting, customer behavior prediction, disease diagnosis, and more. These models encompass various types, including regression, classification, clustering, and deep learning models, tailored to different data and tasks.

The following tasks were completed:

- Conducted exploratory data analysis to determine training labels.
- Established a column for the class.
- Standardized the data.
- Partitioned the dataset into training and test data.
- Identified the optimal hyperparameters for Support Vector Machines, Classification Trees, and Logistic Regression.
- Evaluated the performance of each method using the test dataset.

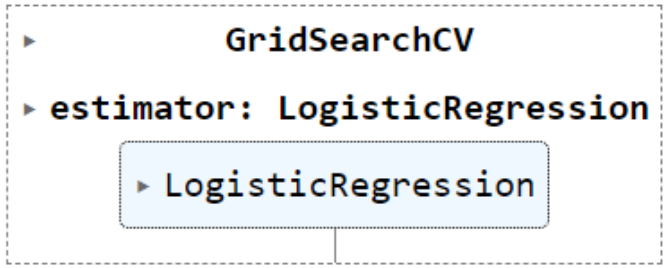
https://github.com/drdataengg/Capstone_Applied_Data_Science_IBM/blob/aa627719d92ad8815b40dd5a4f9eb4e10fbac2d3/SpaceX_Machine%20Learning%20Prediction_Part_5%20(1).ipynb

LOGISTIC REGRESSION ALGORITHM

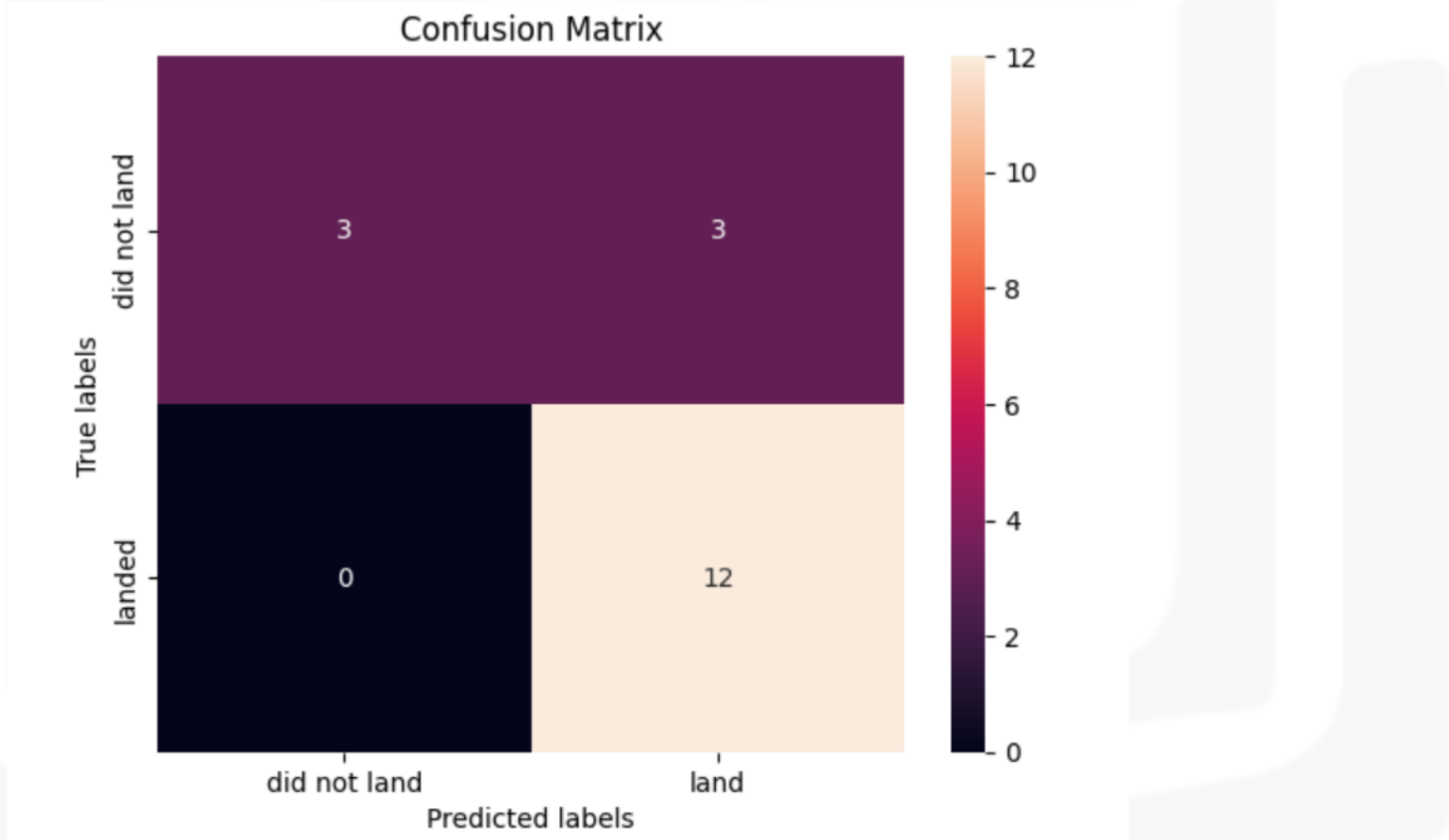
Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

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

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```



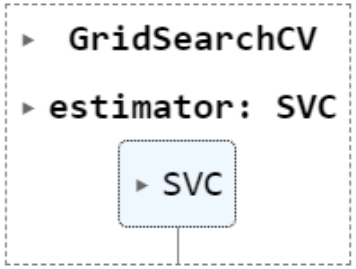
LOGISTIC REGRESSION CONFUSION MATRIX



SVM ALGORITHM

```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

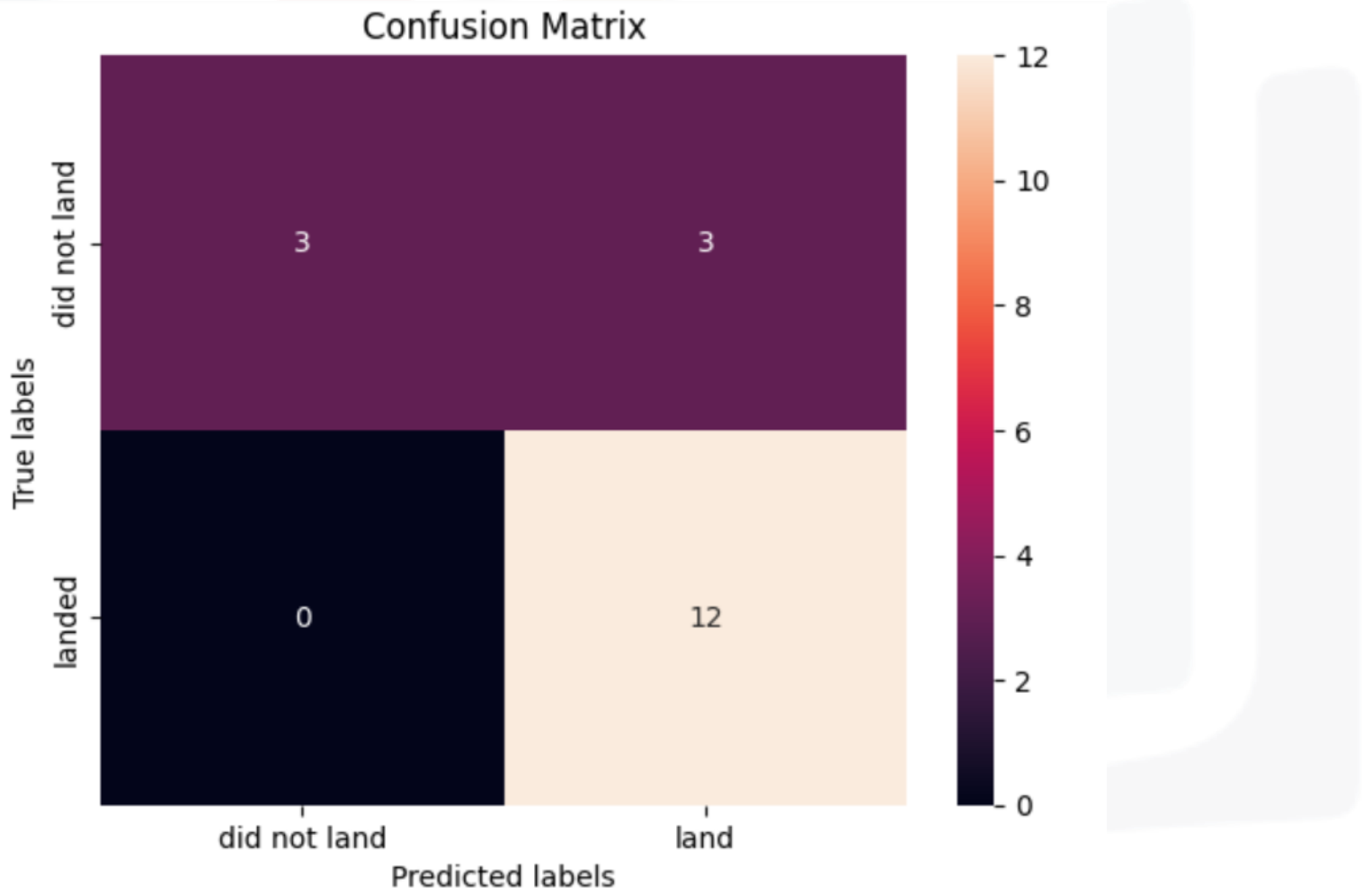
```
svm_cv = GridSearchCV(svm, parameters, cv=10)
svm_cv.fit(X_train, Y_train)
```



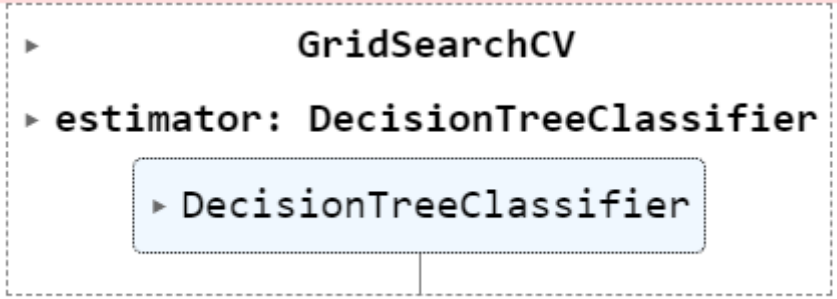

```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

SVM CONFUSION MATRIX



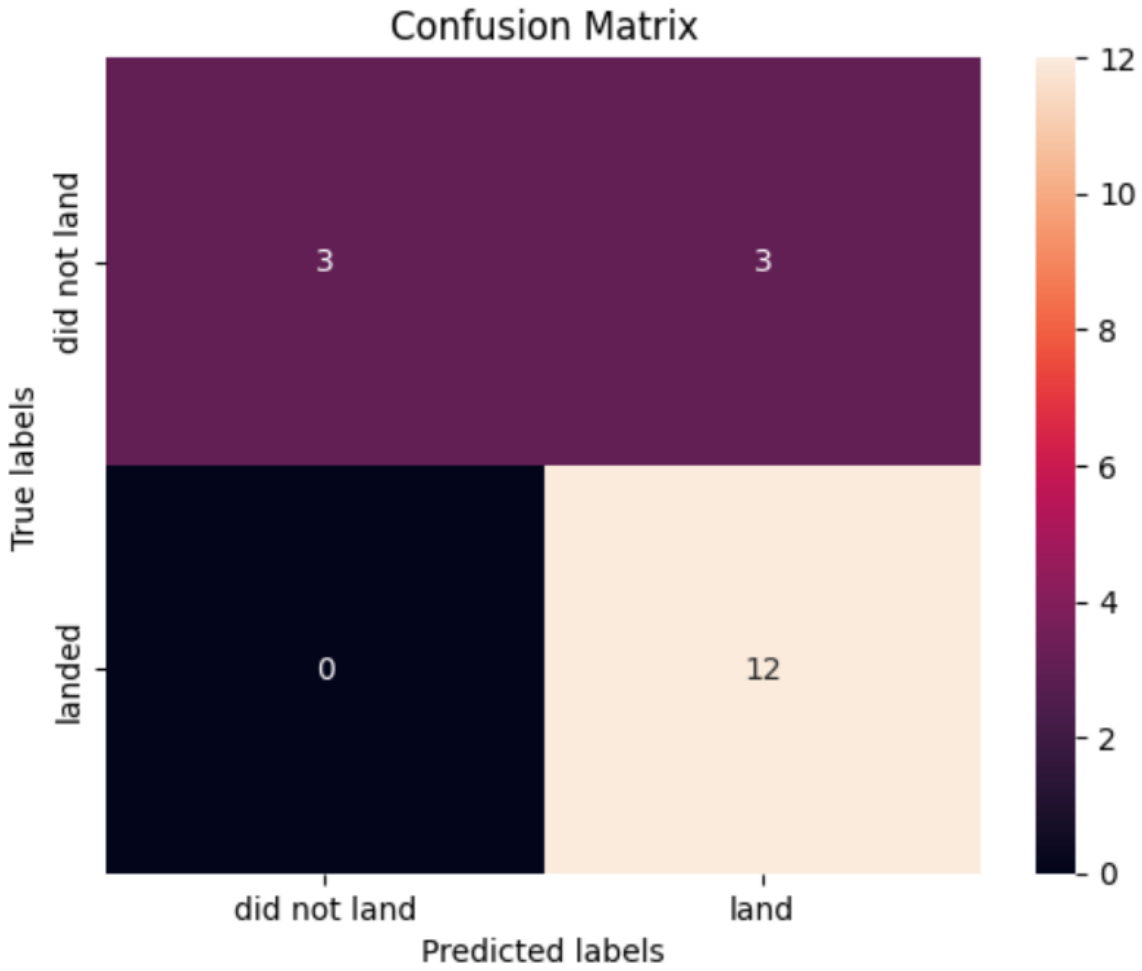
DECISION TREE ALGORITHM




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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt',
'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
accuracy : 0.9017857142857144
```

DECISION TREE CONFUSION MATRIX



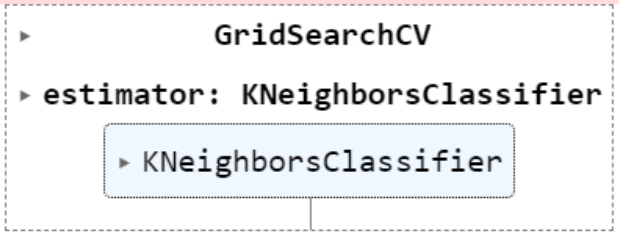
KNEIGHBORS ALGORITHM

```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

```
knn_cv = GridSearchCV(KNN, parameters, cv=10)
knn_cv.fit(X_train, Y_train)
```

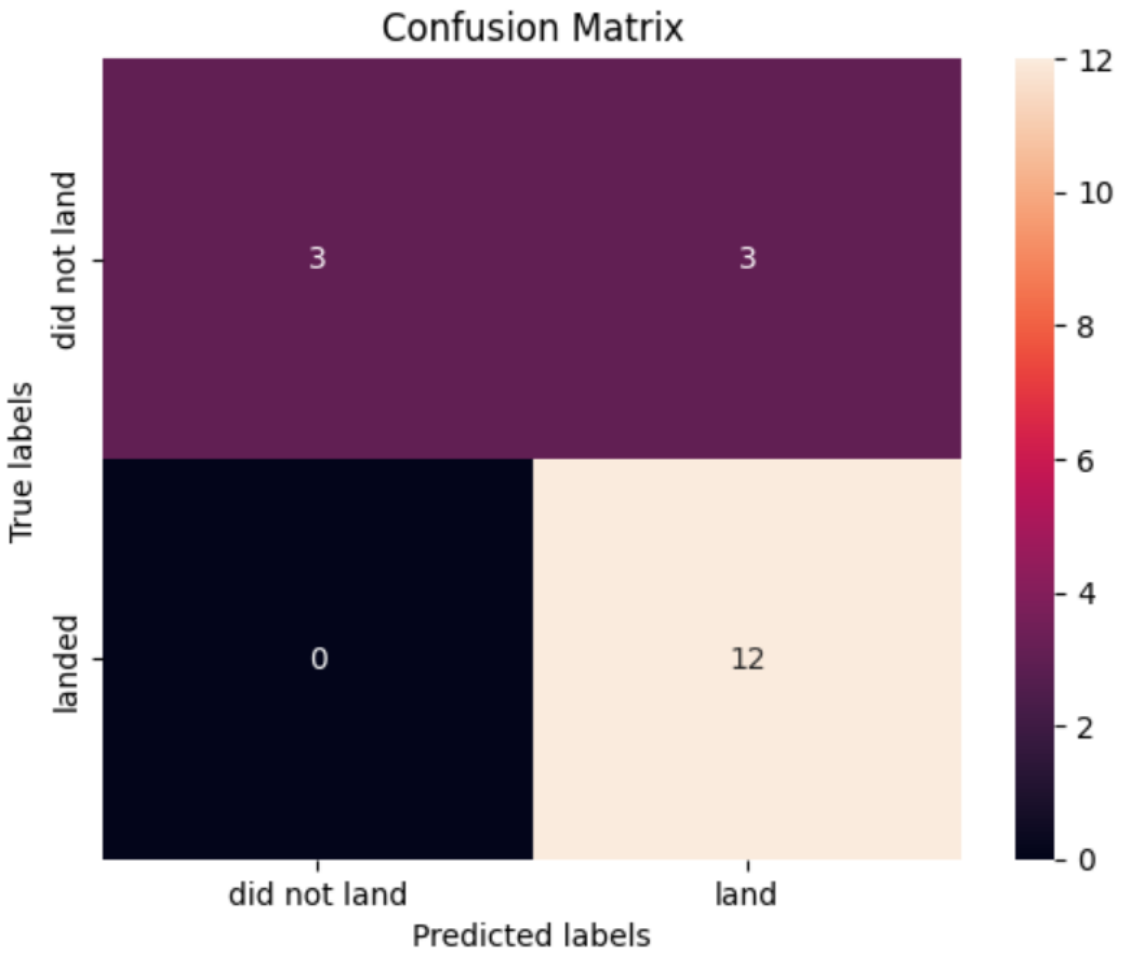
```
/lib/python3.11/site-packages/threadpoolctl.py:1019: RuntimeWarning: libc not found. The ct
Python 3.11 is maybe too old for this OS.
  warnings.warn(
```




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

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

KNEIGHBORS CONFUSION MATRIX



BEST PERFORMING ALGORITHMS

TASK 12

Find the method performs best:

```
max_accuracy = max(accuracy_scores.values())
best_methods = [method for method, accuracy in accuracy_scores.items() if accuracy == max_accuracy]
print("Best performing method(s):", ', '.join(best_methods))
print("Accuracy of the best performing method(s):", max_accuracy)
```

```
Best performing method(s): Logistic Regression, Support Vector Machine, Decision Tree, K-Nearest Neighbors
Accuracy of the best performing method(s): 0.8333333333333334
```

DISCUSSION

- Increasing flight numbers tend to correlate with a higher rate of successful landings, indicating a potential continuous learning process among engineers involved in the project, building on previous results.
- The success rate at each launch site tends to increase as the flight number increases, suggesting a learning curve and improvement in processes over time.
- KSC LC 39A exhibits the highest success rate among all launch sites.
- The majority of payload masses sent are below 7000 kg.
- For payloads exceeding 12000 kg, CCAFS SLC-40 demonstrates a better success rate compared to other launch sites.
- SSO orbit shows a 100% success rate, with LEO orbit also demonstrating higher success rates.
- Beyond 2013, there is an observed increase in success rate with each passing year.
- All four launch sites are situated closer to the coastline, railway, and highway, while being farther from urban areas. This placement ensures safety in case of failure, as launch sites are kept away from densely populated areas. Additionally, proximity to the coast, railway, and highway provides logistical advantages for transportation of heavy rocket or payload objects.

OVERALL FINDINGS & IMPLICATIONS

Findings

- The success rates at each launch site tend to improve with increasing flight numbers, reflecting a learning curve and refinement of processes.
- Beyond 2013, there is a notable increase in success rates, indicating advancements in technology and operational practices.
- The strategic positioning of launch sites away from urban areas, closer to the coastline, railway, and highway, ensures public safety in the event of failure and facilitates efficient logistics for transporting heavy payloads.

Implications

- The findings suggest that the aerospace industry is evolving, with a focus on continuous improvement and learning from past experiences.
- The observed increase in success rates over time underscores the importance of ongoing research, development, and collaboration within the space exploration community.
- The strategic placement of launch sites highlights the significance of safety considerations and logistical efficiency in mission planning and execution.

CONCLUSION

- The analysis indicates a positive trend of improving success rates in space missions, driven by continuous learning and refinement of engineering practices.
- KSC LC 39A emerges as an effective launch site, while CCAFS SLC-40 shows promise for handling heavier payloads.
- Consistent success rates in specific orbits highlight the significance of strategic mission planning.
- The observed increase in success rates beyond 2013 reflects advancements in technology and operational practices, emphasizing the dynamic nature of the aerospace industry.

APPENDIX

Calculation of distance between launch sites and other points of interests

	Launch Site	Coordinates	Total Launches	Success	Coastline	Railway	Highway	City	Distance to Coastline	Distance to Railway	Distance to Highway	Distance to City
0	CCAFS LC-40	[28.562302, -80.577356]	26	7	[28.56288, -80.56789]	[28.56178, -80.58718]	[28.56235, -80.57063]	[28.08773, -80.65567]	0.926991	0.961492	0.657104	53.34058
1	CCAFS SLC-40	[28.563197, -80.57682]	7	3	[28.56288, -80.58794]	[28.56178, -80.58718]	[28.56235, -80.57063]	[28.08773, -80.65567]	1.086910	1.024296	0.612011	53.44665
2	KSC LC-39A	[28.573255, -80.646895]	13	10	[28.58015, -80.64205]	[28.57317, -80.65404]	[28.57303, -80.6537]	[28.51456, -81.35834]	0.901196	0.698008	0.665203	69.82107
3	VAFB SLC-4E	[34.632834, -120.610745]	10	4	[34.63487, -120.62503]	[34.63519, -120.62373]	[34.63225, -120.59916]	[34.64874, -120.45742]	1.326832	1.216936	1.062264	14.14223

APPENDIX

	Launch Site	Total No. of Launches	Successful Landing of First Stage
1	CCAFS LC-40	26	7
2	CCAFS SLC-40	7	3
3	KSC LC-39A	13	10
4	VAFB SLC-4E	10	4