IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

ABBA ONOJA

6th Nov, 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection with API
  - Data collection with web scrapping
  - Data wrangling
  - Exploratory Data Analysis using sql
  - Exploratory Data Analysis with visualization
  - Interactive Visual Analysis with folium
  - Building a Dashboard with Plotly Dash
  - Predictive analysis (Classification)
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive analytics screenshots
  - Predictive analysis results

Introduction

- **Project background and context**
  - SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch. In this module, you will be provided with an overview of the problem and the tools you need to complete the course.
- **Problems you want to find answers**
  - The project task is to determine if the first stage of the SpaceX Falcon 9 rocket will land successfully.

Section 1

Methodology

Methodology

Executive Summary

- **Data collection methodology:**
  - The data was gathered by utilizing the SpaceX API as well as conducting web scraping from Wikipedia to collect the data.
- **Perform data wrangling**
  - One-hot encoding was applied to categorical features
- **Perform exploratory data analysis (EDA) using visualization and SQL**
  - Exploratory data analysis using python library (Matplotlib and Seaborn) , 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

The following datasets was collected:

- SpaceX launch data that was gathered from the SpaceX REST API
- This API will give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.
- The SpaceX REST API endpoints or URL, starts with api.spacexdata.com/v4/.
- Web scraping was carried out on Wikipedia for Falcon 9 launch records using BeautifulSoup

Data Collection – SpaceX API

- **Data collection with SpaceX REST API**

1. Requesting rocket launch data from SpaceX API with the following URL

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

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

2. Use json_normalize meethod to convert the json result into a dataframe

```
# Use json_normalize meethod to convert the json result into a dataframe
if response.status_code == 200:
    data = response.json()
    df = pd.json_normalize(data)
    print("succesful")
else:
    print(f"API request failed with status code: {response.status_code}")
```

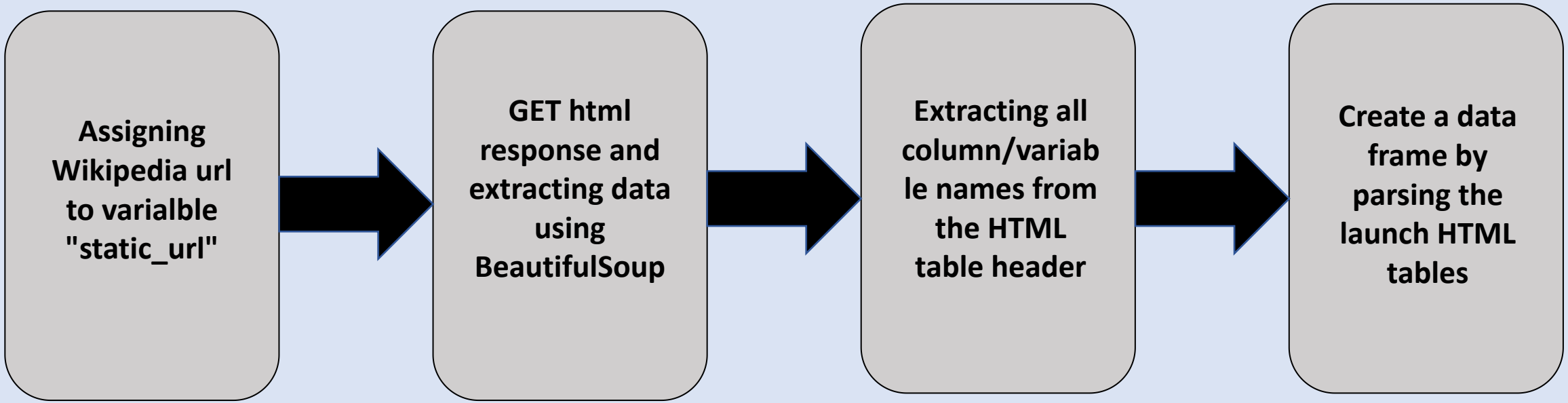
3. Use the API again to gather information about the launches using the IDs given for each launch

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.

data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]


# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a single r
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

- ***https://github.com/PROD-A33A/COURSERA-DATA-SCIENCE-CAPSTONE/blob/main/Data%20Collection%20API%20Lab.ipynb***

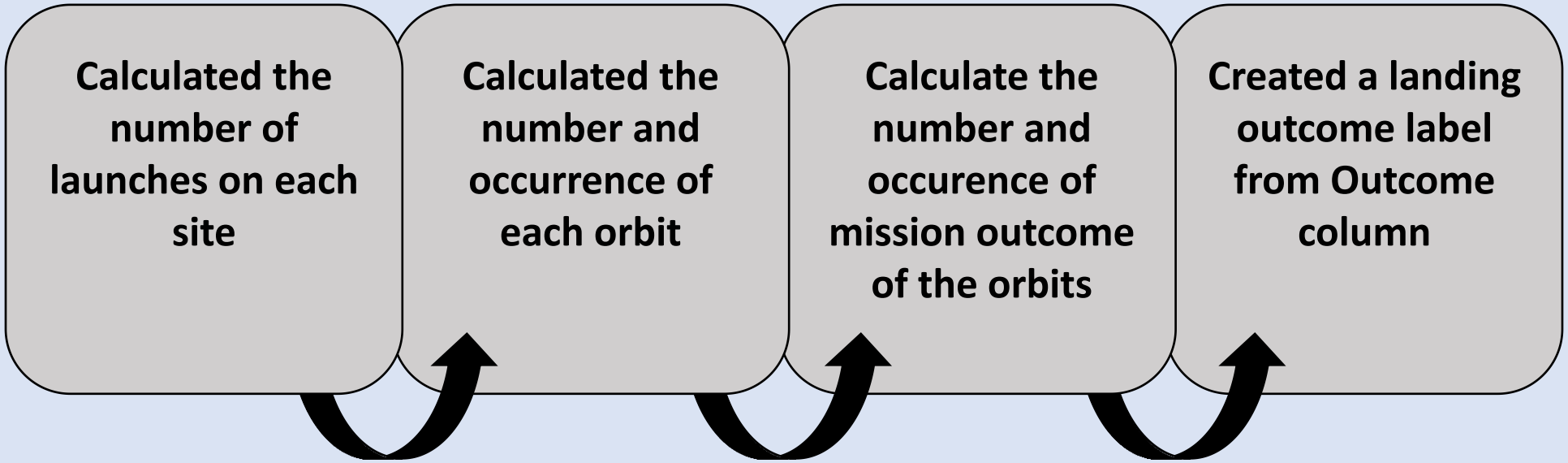
Data Collection - Scraping

- Web Scrapping from Wikipedia



- *https://github.com/PROD-A33A/COURSERA-DATA-SCIENCE-CAPSTONE/blob/main/Data%20Collection%20with%20Web%20Scraping%20lab.ipynb*

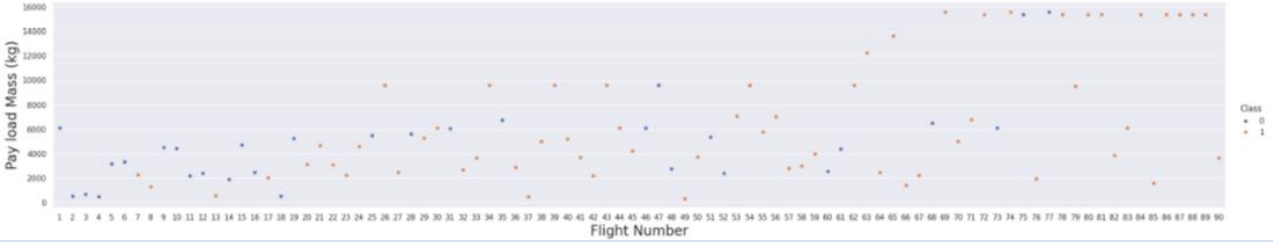
Data Wrangling



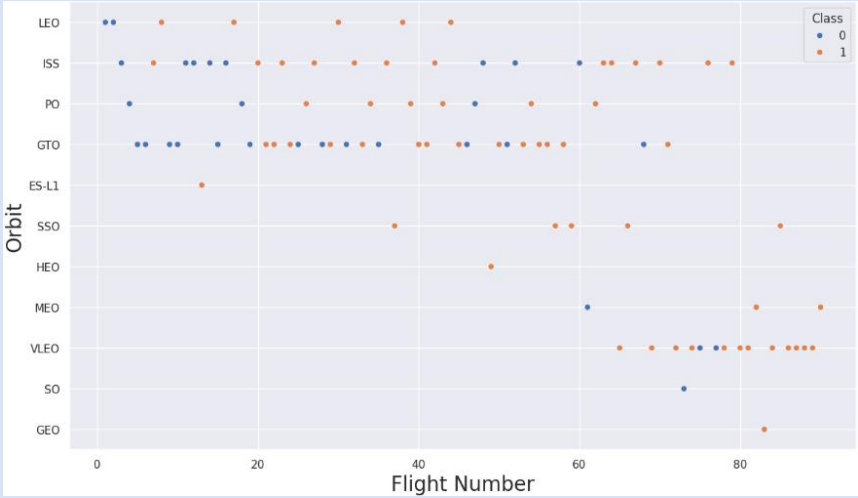
- https://github.com/PROD-A33A/COURSERA-DATA-SCIENCE-CAPSTONE/blob/main/Data%20Wrangling.ipynb

EDA with Data Visualization

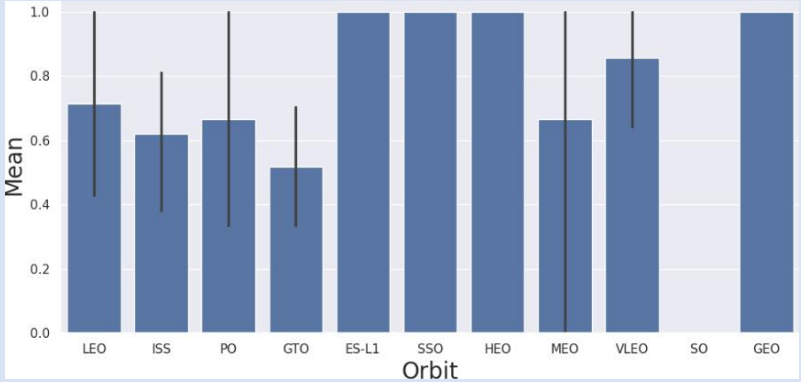
1. Categorical plot



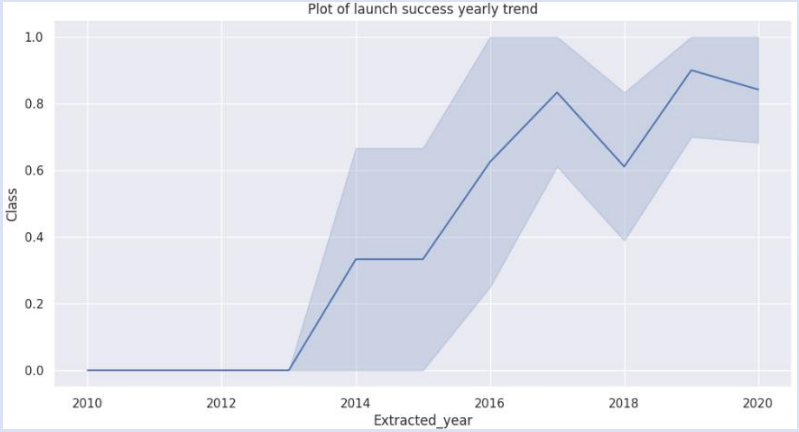
2. Scatter plot



4. Bar chart



4. Line chart



These charts best highlight the relationship between the variables

https://github.com/PROD-A33A/COURSERA-DATA-SCIENCE-CAPSTONE/blob/main/EDA%20with%20Visualization%20Lab.ipynb

EDA with SQL

- **SQL queries performed include:**
  - Displaying the names of the unique launch sites in the space mission
  - Displaying 5 records where launch sites begin with the string 'KSC’
  - Displaying the total payload mass carried by boosters launched by NASA (CRS)
  - Displaying average payload mass carried by booster version F9 v1.1
  - Listing the date where the successful landing outcome in drone ship was achieved.
  - Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000.
  - Listing the total number of successful and failure mission outcomes
  - Listing the names of the booster_versions which have carried the maximum payload mass.
  - Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
  - Ranking the count of successful landing_outcomes between the date 2010 06 04 and 2017 03 20 in descending order.

- **https://github.com/PROD-A33A/COURSERA-DATA-SCIENCE-CAPSTONE/blob/main/EDA%20with%20SQL.ipynb**

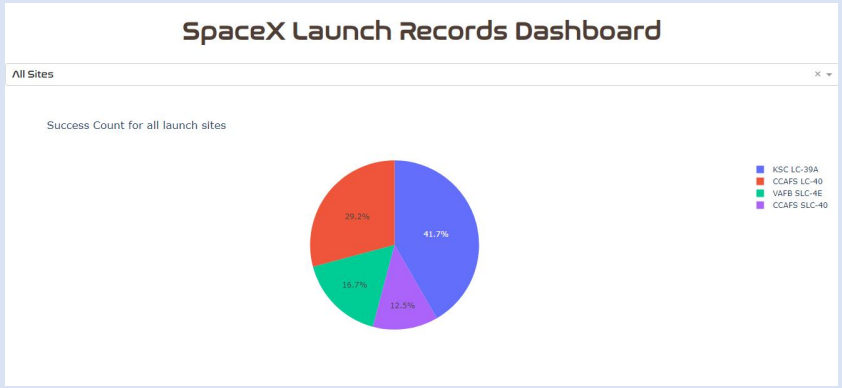
Build an Interactive Map with Folium

- **Marker:** These markers were used to represent the launch sites on the map.
- **Circle:** For indicating distances to features like railways, highways, and cities.
- **Lines**: To represent distance between launch sites and features such as coastline, railway, closest city, etc.
- **MarkerCluster:** Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.

- https://github.com/PROD-A33A/COURSERA-DATA-SCIENCE-CAPSTONE/blob/main/Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

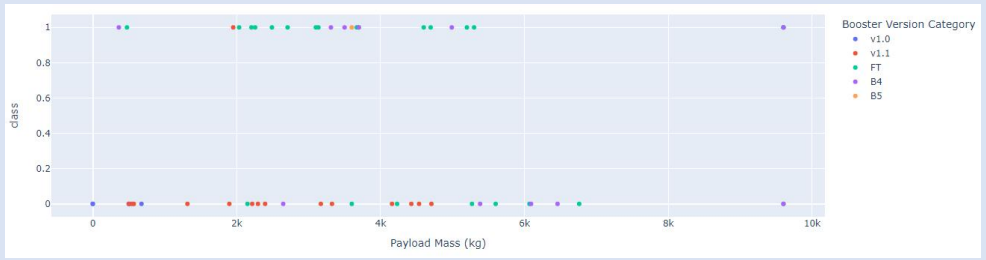
Build a Dashboard with Plotly Dash

- **PIE CHARTS:** To show the total success launches and failures by sites.



- **RANGESLIDER:** To select different payload range.

- **SCATTER PLOT:** To show the relationship between class(outcomes) and payload mass(kg).



https://github.com/PROD-A33A/COURSERA-DATA-SCIENCE-CAPSTONE/blob/main/spacex_dash_app%20(1).py

Predictive Analysis (Classification)

- Loaded the data using numpy and pandas, transformed the data, split the data into training and testing.
- Created a NumPy array from the column Class in data, by applying the method to_numpy() then assign it to the variable Y.
- Created 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.
- Calculated the accuracy on the test data using the method score.
- Created a support vector machine object then create a GridSearchCV object svm_cv with cv - 10.
- Create a k nearest neighbors object then create a GridSearchCV object knn_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.
- Found the method that performs best.

https://github.com/PROD-A33A/COURSERA-DATA-SCIENCE-CAPSTONE/blob/main/Machine%20Learning%20Prediction%20lab.ipynb

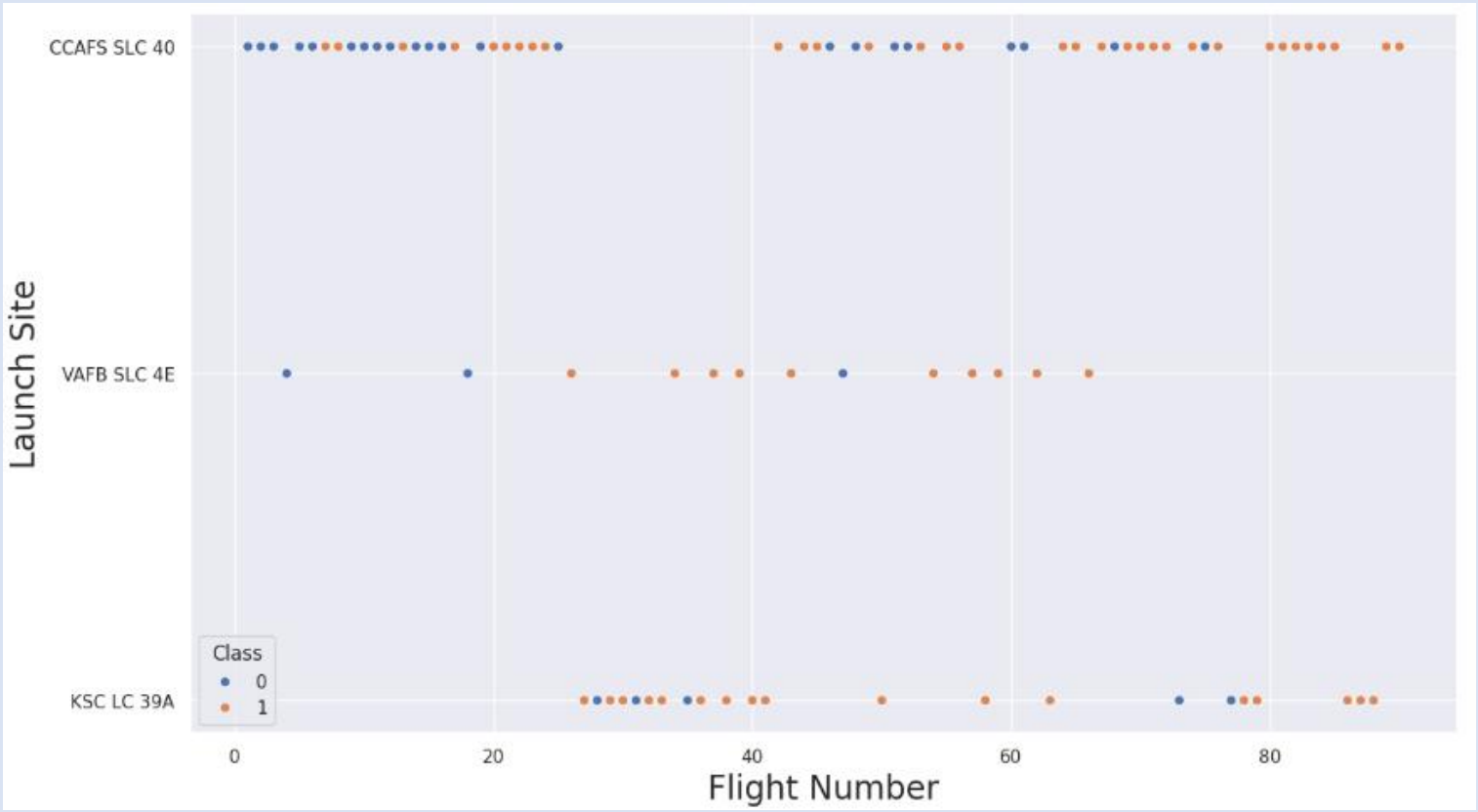
Results

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

Section 2

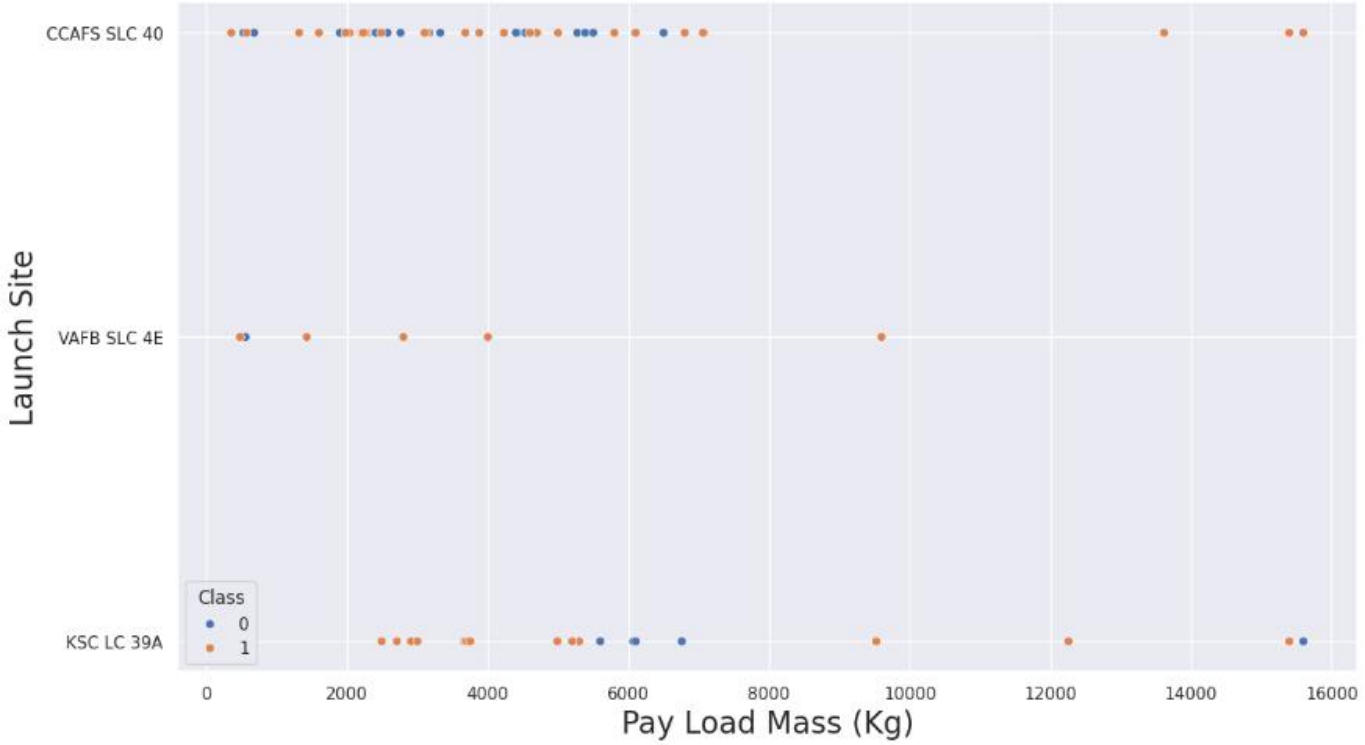
Insights drawn from EDA

Flight Number vs. Launch Site



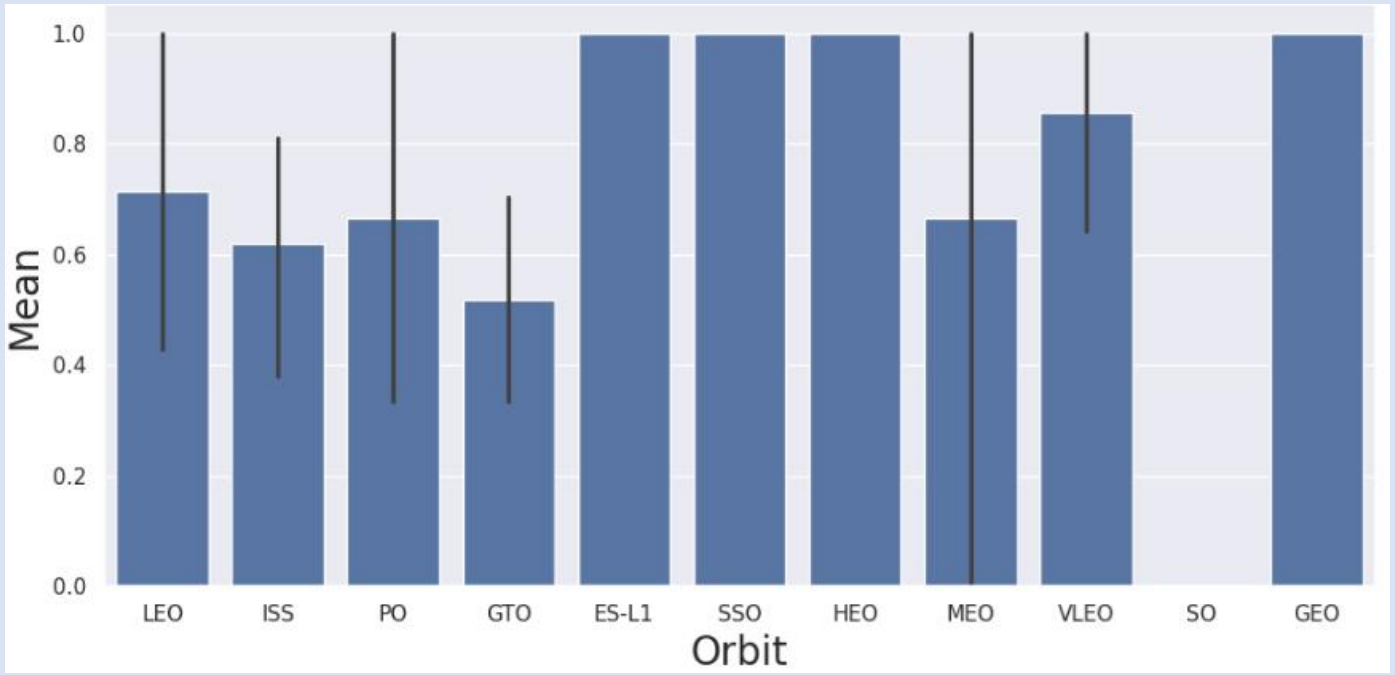
- From the plot we can conclude that the size of the flight amount is directly proportional to the succes of the launch sites i.e the larger the flight amount at a launch site, the greater the success rate at a launch site.

Payload vs. Launch Site



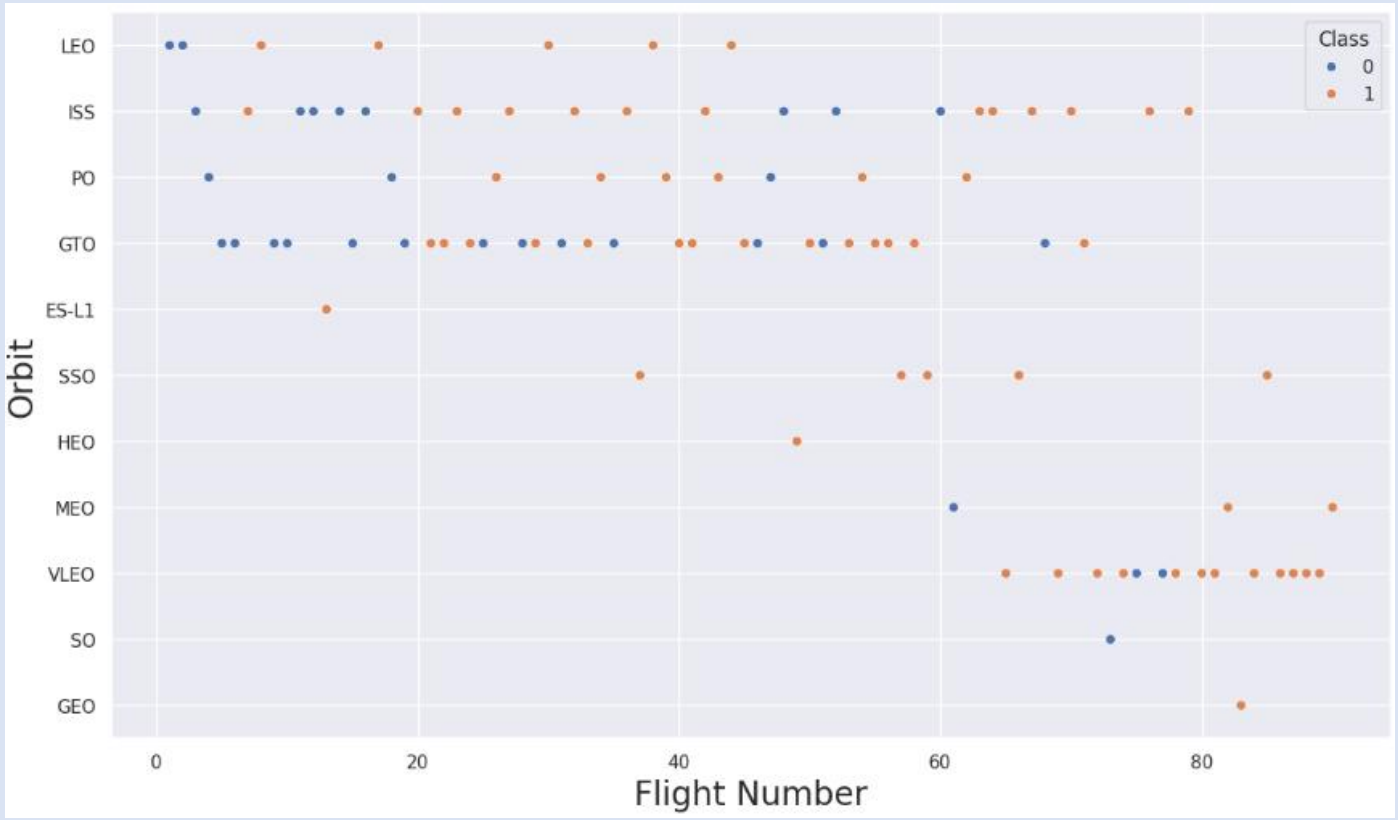
- The majority of Pay Loads with lower Mass have been launched from CCAFS SLC 40.

Success Rate vs. Orbit Type



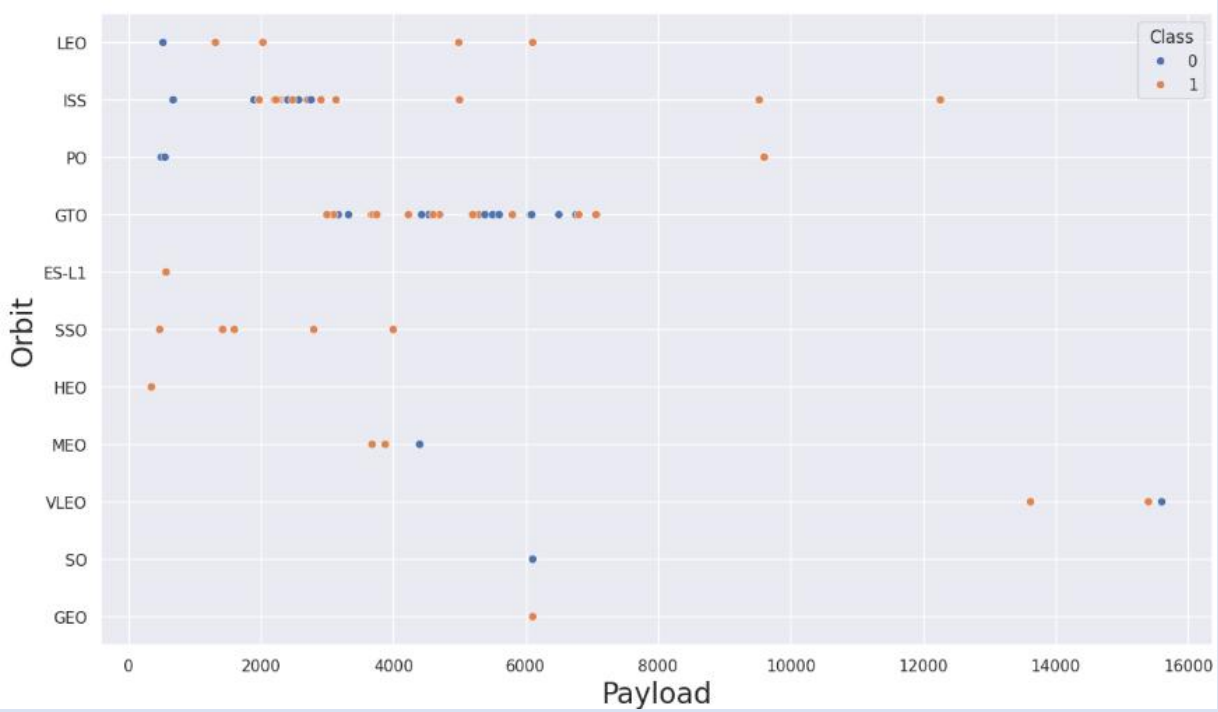
- **The orbit types of ES-L1, GEO, HEO, SSO are among the highest success rate.**

Flight Number vs. Orbit Type



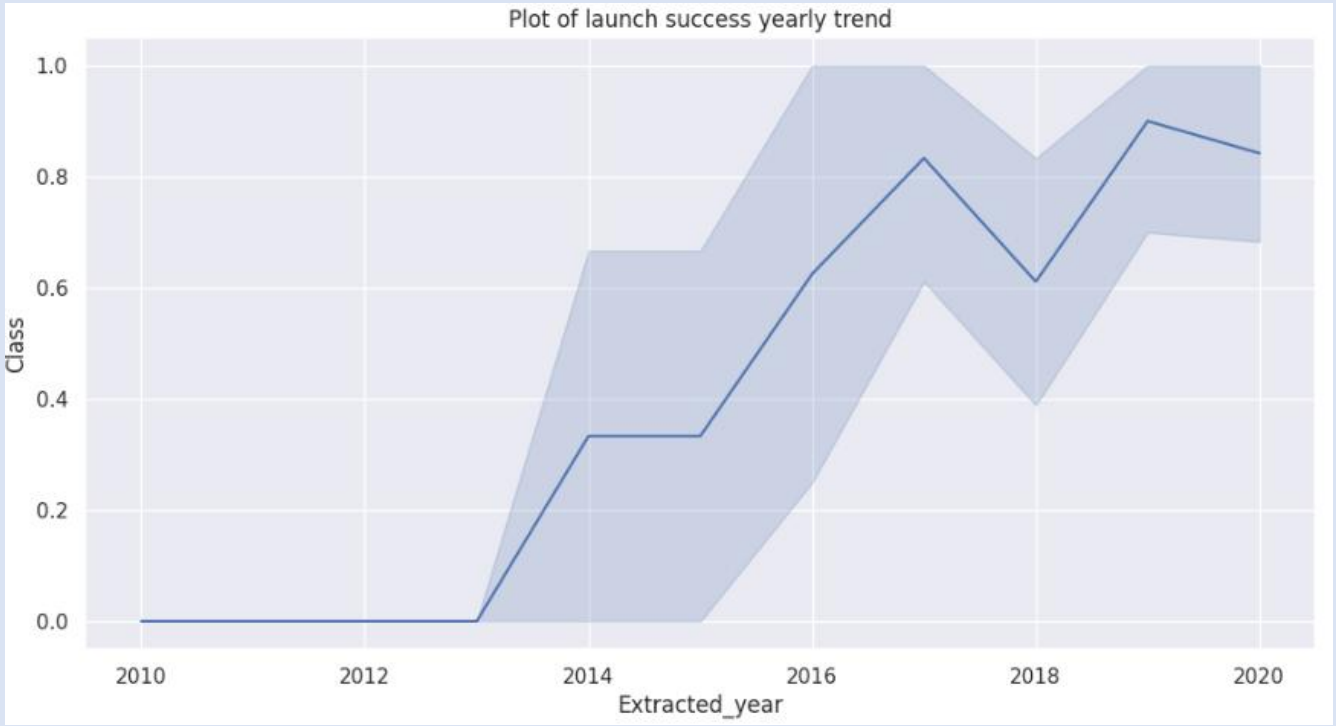
- A trend can be observed of shifting to VLEO launches in recent years.

Payload vs. Orbit Type



- **Launches with heavy payloads are more successful for ISS, PO and LEO.**

Launch Success Yearly Trend



- Launch success rate has increased significantly since 2013 and has stablised since 2019, potentially due to advance in technology and lessons learned.

All Launch Site Names

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

- 1. Executed a SQL query using the DISTINCT keyword to retrieve the unique launch site names from the SpaceX data.
- 2. Query: `%sql SELECT DISTINCT "LAUNCH_SITE" FROM SPACEXTABLE`

Launch Site Names Begin with 'CCA'

Task 2

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

[33]:
```
%sql SELECT * FROM SPACEXTABLE WHERE "LAUNCH_SITE" LIKE '%CCA%' LIMIT 5
```

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

[33]:

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

Used the *“%sql SELECT * FROM SPACEXTABLE WHERE "LAUNCH_SITE" LIKE '%CCA%' LIMIT 5”* query to display 5 records where launch sites begin with `CCA’ .

Total Payload Mass

Task 3

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

```
%sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "CUSTOMER" = 'NASA (CRS)'
```

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

SUM("PAYLOAD_MASS__KG_")
45596

- **Calculated the total payload carried by boosters from NASA as 45596 using the query *"%sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "CUSTOMER" = 'NASA (CRS)' ",* which gave a sum of 45596.**

Average Payload Mass by F9 v1.1

Task 4

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

```
%sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "BOOSTER_VERSION" LIKE '%F9 v1.1%'
```

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

AVG("PAYLOAD_MASS__KG_")
2534.6666666666665

- **Calculated the average payload mass carried by booster version F9 v1.1 using thee query *“%sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "BOOSTER_VERSION" LIKE '%F9 v1.1%' “ , which gave a result of 2534.67***

First Successful Ground Landing Date

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_date FROM SPACEXTBL WHERE Landing_Outcome LIKE 'Success (ground pad)'
```

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

First_Successful_landing_date
2015-12-22

- Used the query (***%sql SELECT MIN("DATE") FROM SPACEXTABLE WHERE "Landing _Outcome" LIKE '%Success%'***) to show the first successful landing which gave a date of 22nd December, 2015.

Successful Drone Ship Landing with Payload between 4000 and 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_ > 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

- Used the ***WHERE*** and the ***And*** condition ***(%sql SELECT BOOSTER_VERSION FROM SPACEXTABLE WHERE LANDING_OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000)*** to show the successful landing with payload between 4000 and 6000.

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTABLE WHERE "MISSION_OUTCOME" LIKE '%Success%') AS SUCCESS, \
(SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS FAILURE
```

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

SUCCESS	FAILURE
100	1

Used the query (***%sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTABLE WHERE "MISSION_OUTCOME" LIKE '%Success%') AS SUCCESS, \ (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS FAILURE***) to count the number of mission outcome and group by mission outcome, which gave a result of 100 successful missions and 1 failed mission.

Boosters Carried Maximum Payload

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

```
%sql SELECT DISTINCT "BOOSTER_VERSION" FROM SPACEXTABLE \
WHERE "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") FROM SPACEXTBL)
```

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

- Used the query ***(%sql SELECT DISTINCT "BOOSTER_VERSION" FROM SPACEXTABLE \WHERE "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") FROM SPACEXTBL))*** to determine the booster that have carried the maximum payload .

2015 Launch Records

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 Booster_Version, Launch_Site, LANDING_OUTCOME FROM SPACEXTABLE WHERE strftime('%Y', Date) = '2015' AND LANDING_OUTCOME = 'Failure (drone ship)
```

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

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

- Used the query ***(%sql SELECT Booster_Version, Launch_Site, LANDING_OUTCOME FROM SPACEXTABLE WHERE strftime('%Y', Date) = '2015' AND LANDING_OUTCOME = 'Failure (drone ship)';)*** to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015.

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

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(LANDING_OUTCOME) FROM SPACEXTABLE WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'GROUP BY LANDING_OUTCOME ORDER BY COU
```

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

Landing_Outcome	COUNT(LANDING_OUTCOME)
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

- **QUERY :** %sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) FROM SPACEXTABLE WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'GROUP BY LANDING_OUTCOME ORDER BY COUNT(LANDING_OUTCOME) DESC
- ‘strftime('%Y', Date) = '2015': This condition extracts the year from the 'Date' column using the strftime function and filters the results to include only rows where the year is '2015.'
- LANDING_OUTCOME = 'Failure (drone ship)': This condition filters the results to include only rows where the landing outcome is specifically 'Failure (drone ship).'

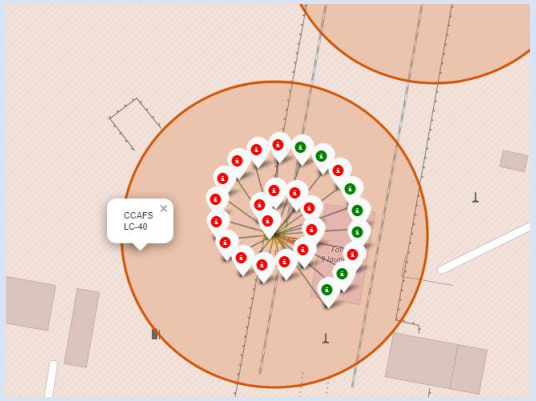

Section 3
Launch Sites
Proximities Analysis

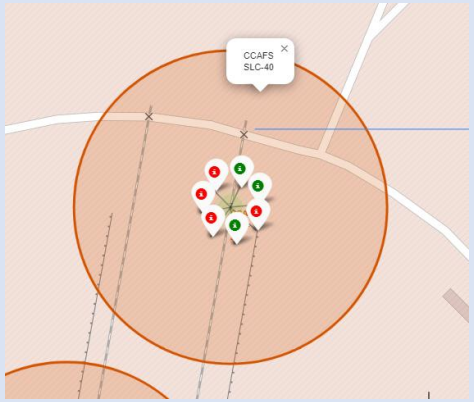
Launch sites marked on a map

- **SpaceX has 2 launch sites which are situated in USA. This is enhance operational Flexibility and to reduce Launch Congestion.**

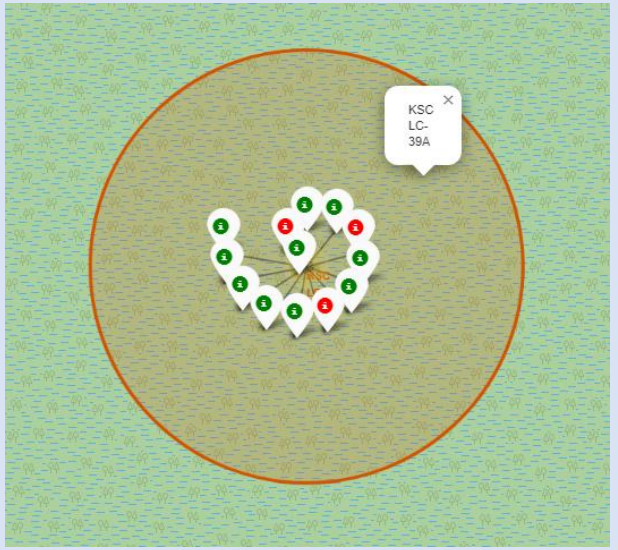
Launch Outcomes on Color-Labeled Folium Map

- CCAFS LC-40





- CCAFS SLC-40



- KSC LC-39A

- **FLORIDA LAUNCH SITES**

- VAFB SLC-4E

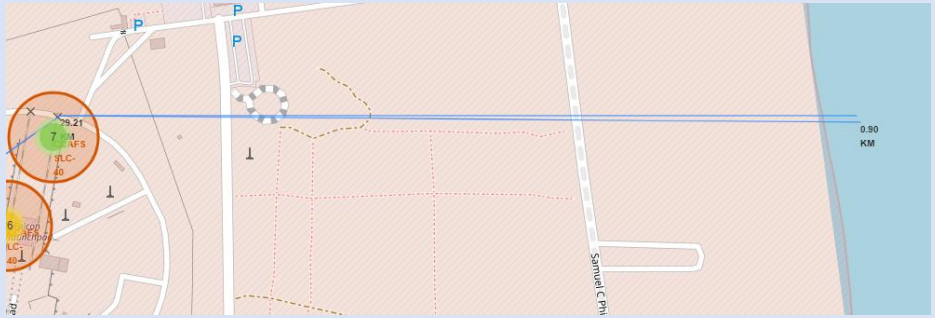
- **CALIFORNIA LAUNCH SITE**

GREEN MARKERS = SUCCESSFUL LAUNCHES
RED MARKERS = FAILED LAUNCHES

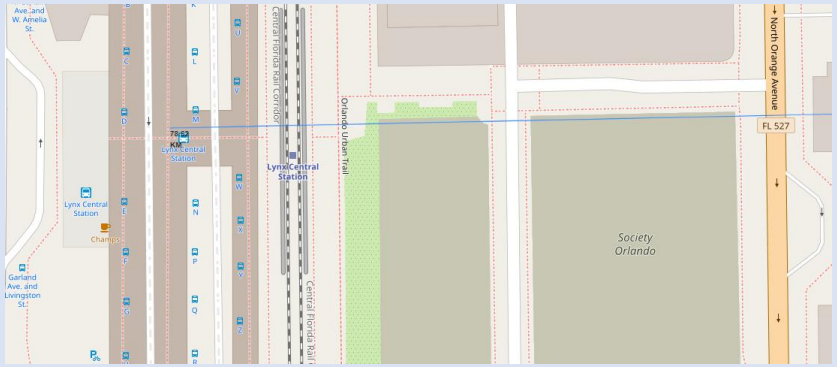
Launch Sites Proximities

- These maps show the distance between launch sites and its proximities such as railway, highway, coastline, with distance calculated and displayed. The distance from railway happens to be the farthest and the distance from coastline is the closest.

- Distance from Coastline: 0.90km



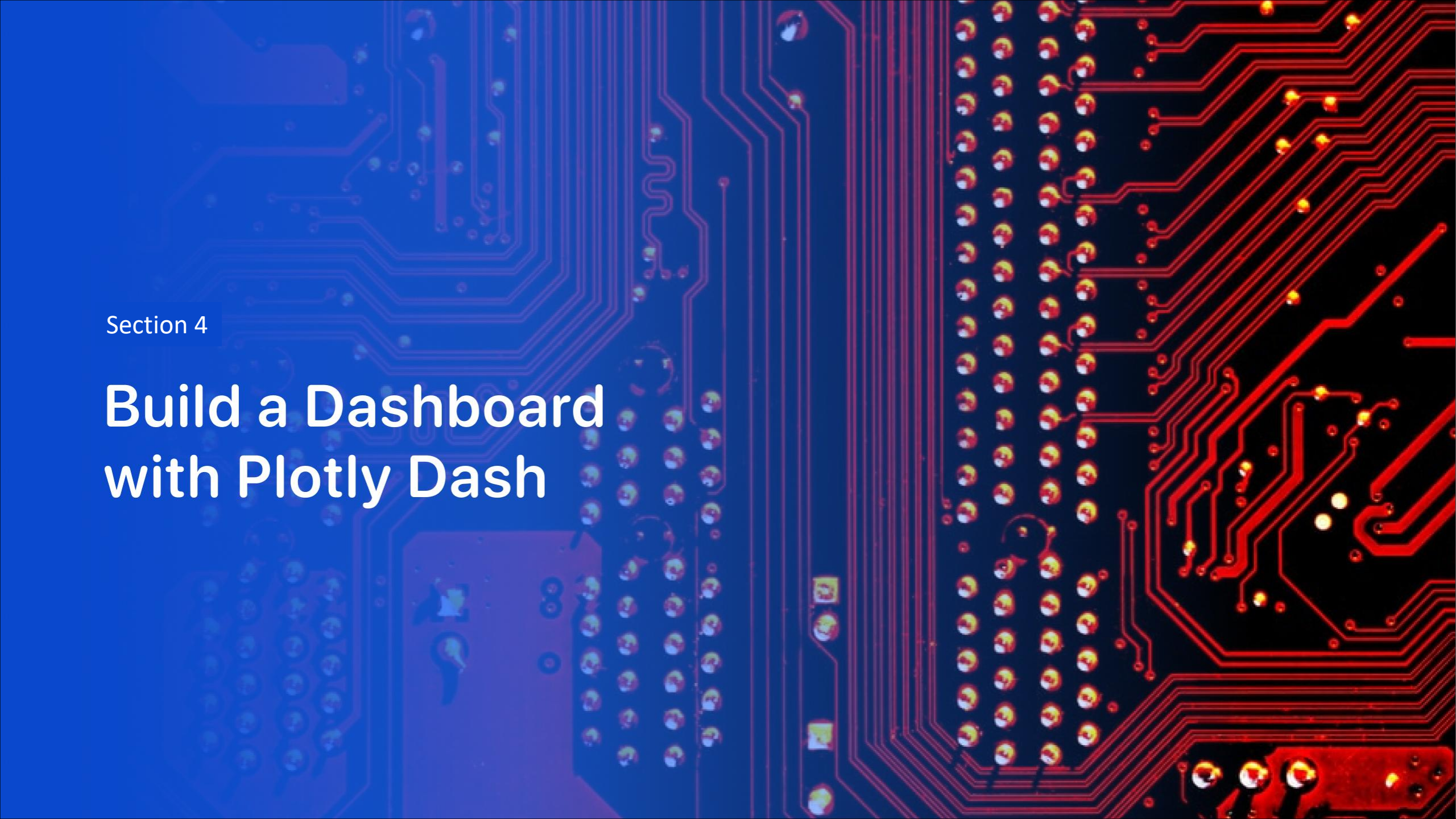
- Distance from Railway: 78.62km



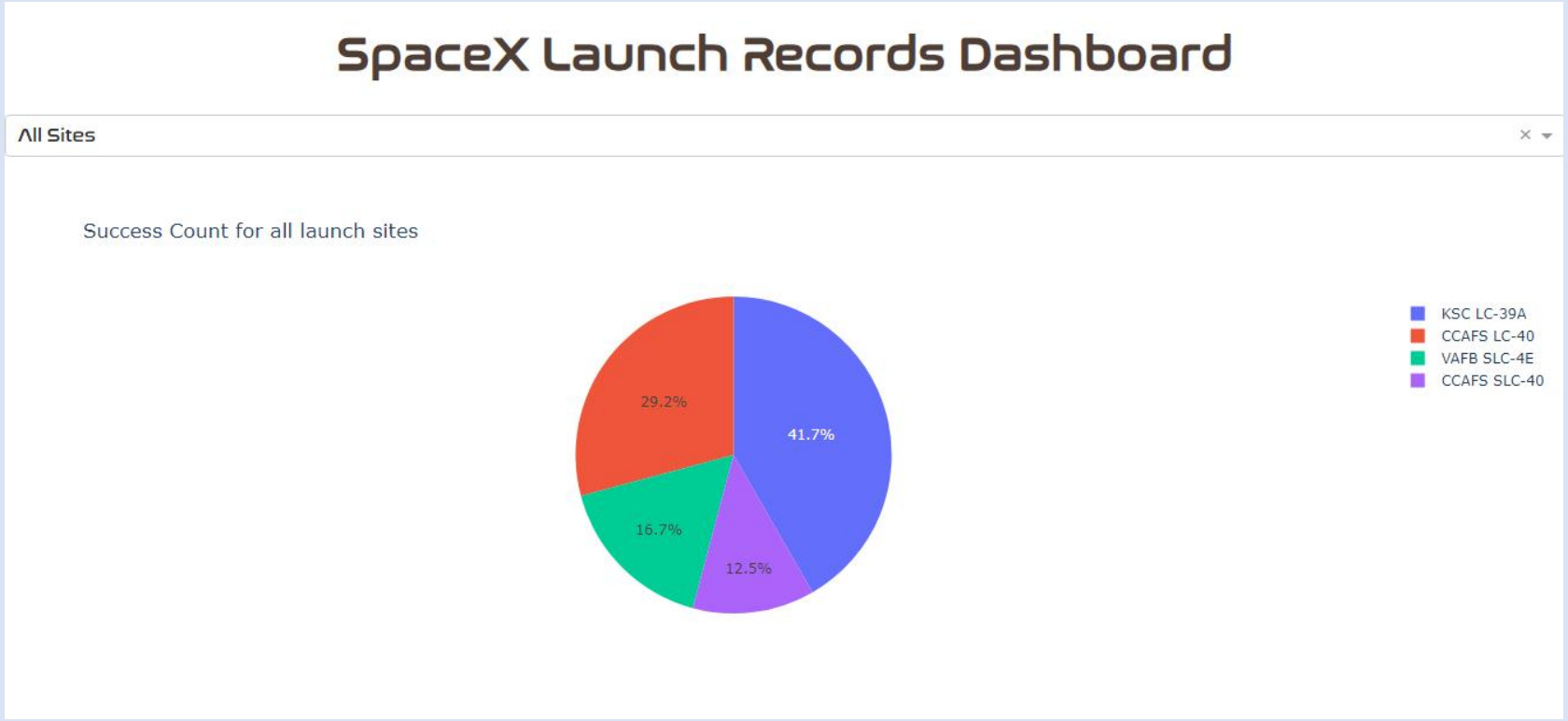
- Distance from Highway: 29.21km

Section 4

Build a Dashboard with Plotly Dash

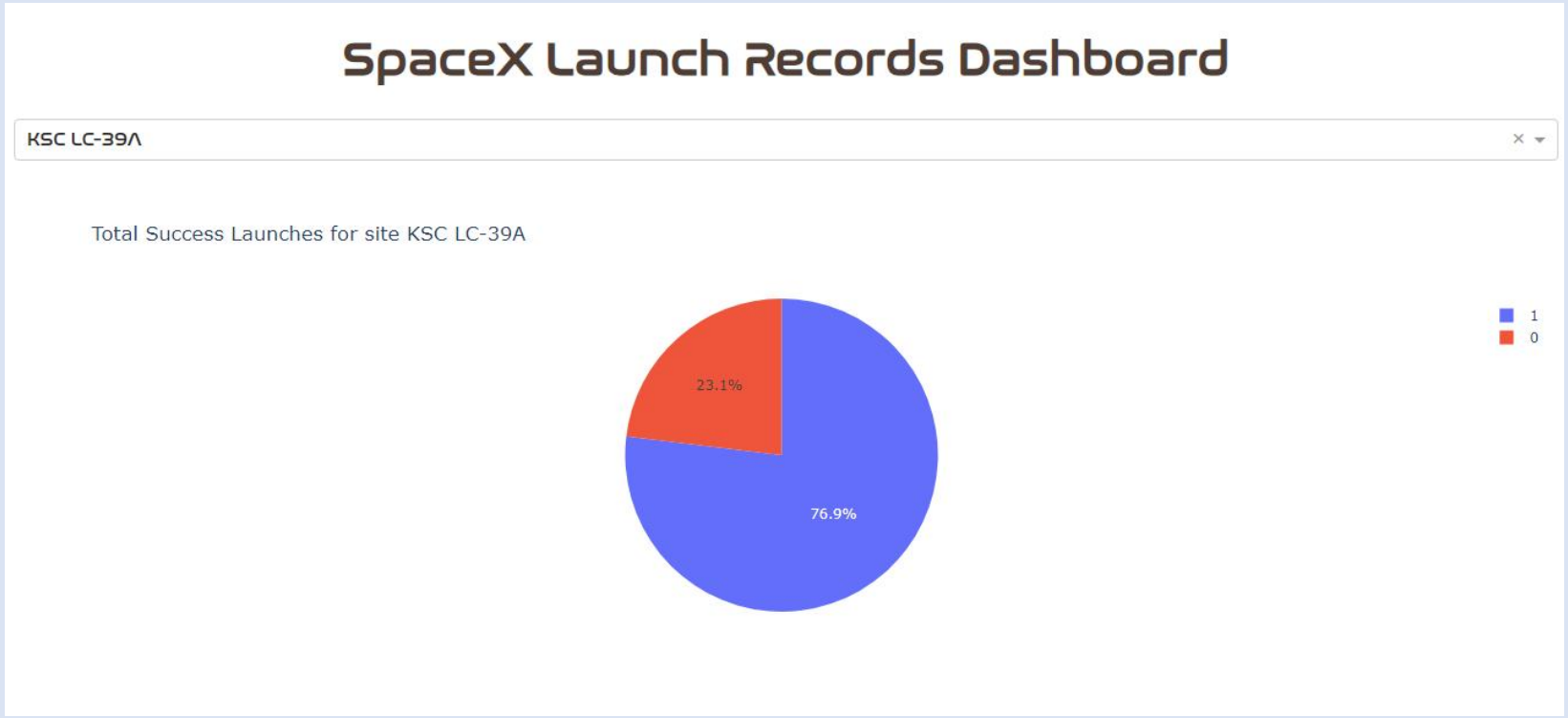


Launch Success Count for all Sites



- **This is a dashboard with a pie chart showing the success percentage achieved by each launch site with KSC LC-39A in florida having the most successful launches and VAFB SLC-4E in california having the least amount of successful launches.**

Launch Site with Highest Launch Success Ratio



- **KSC LC-39A is the launch site with the highest success ratio with 76.9% success rate and a failure rate of 23.1%.**

Payload vs. Launch Outcome scatter plot for all sites

- **Payload between 0kg - 5000kg**

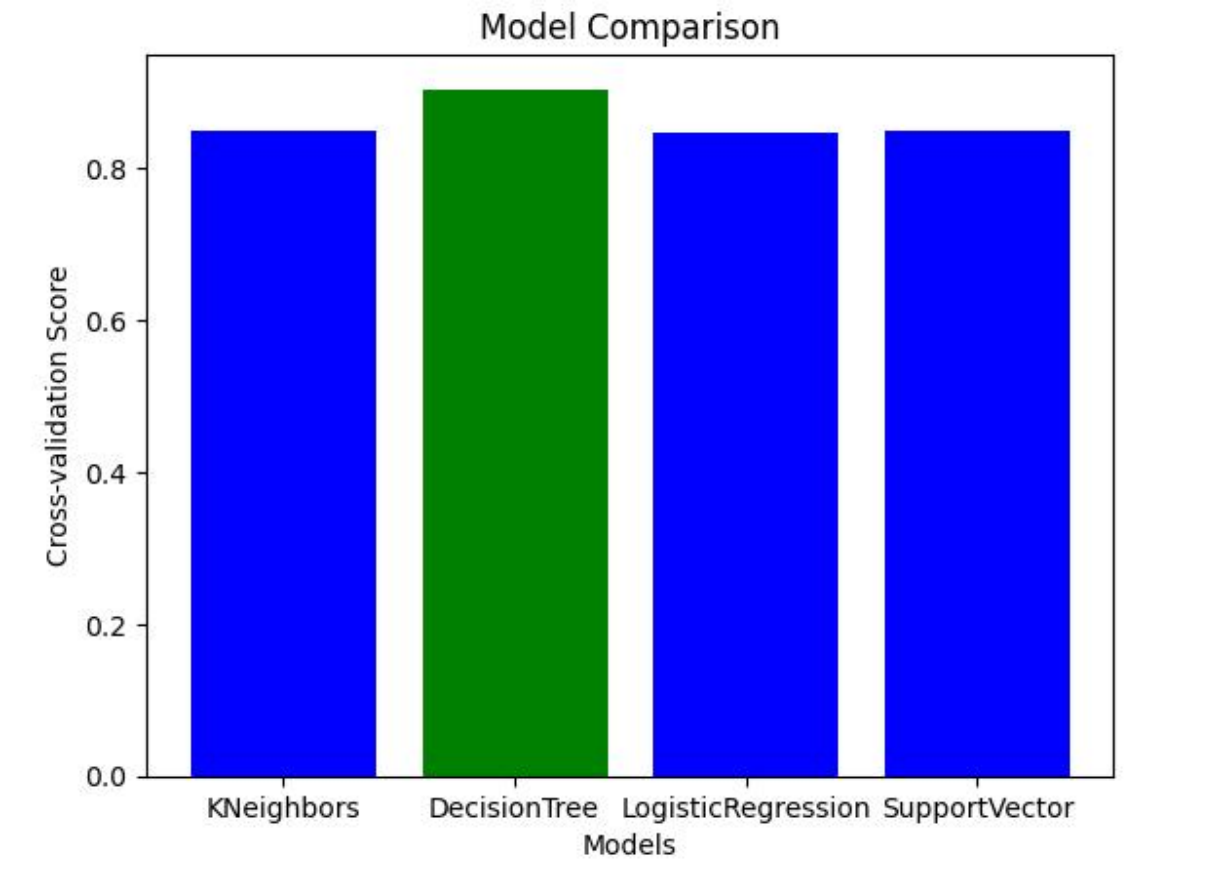
- **Payload between 5000kg - 10000kg**

- **Success rate for payloads between 0 - 5000kg are higher than that of payloads between 5000kg - 10000kg**
- **Booster version ‘FT’ had the highest success rate with most of its payload between 0kg - 5000kg**

Section 5

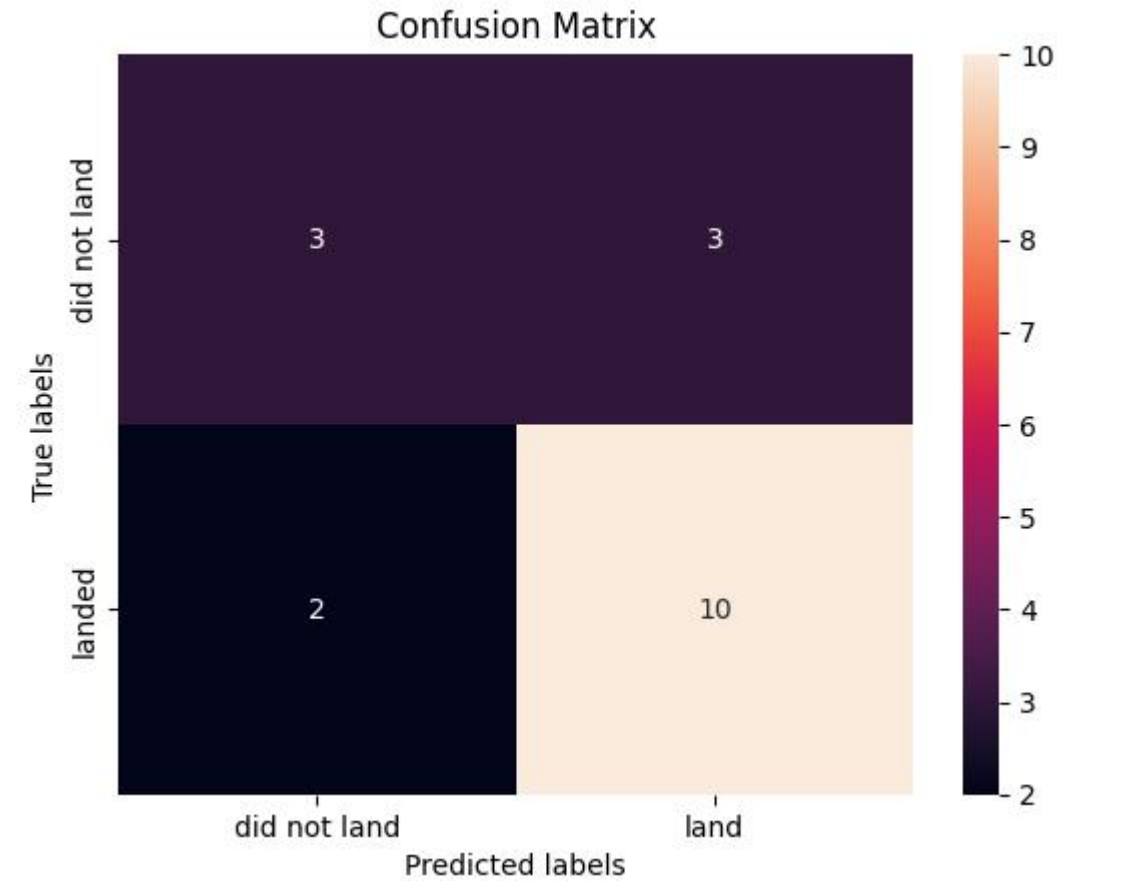
Predictive Analysis (Classification)

Classification Accuracy



- **The decision tree model has the highest classification accuracy with a score of 0.9035714285714287.**

Confusion Matrix



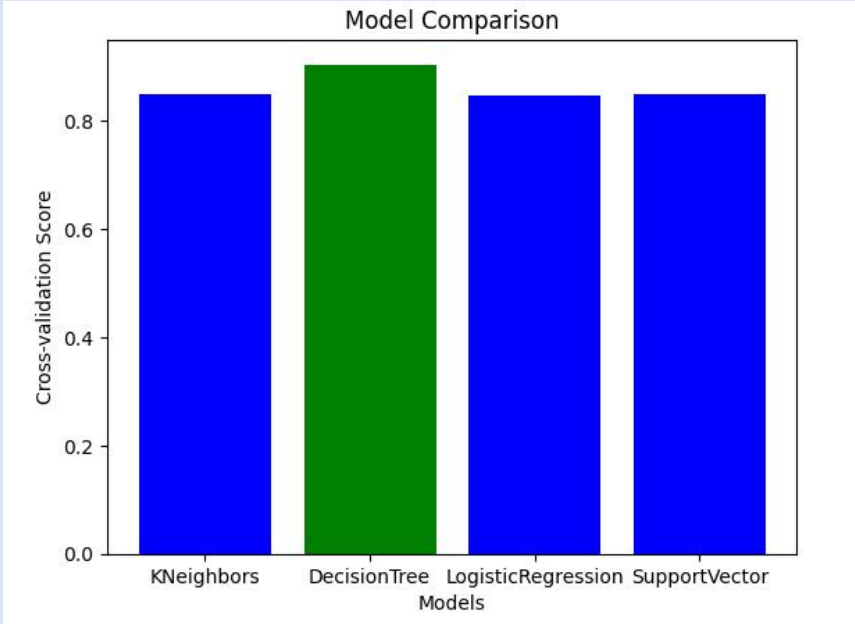
- **Accuracy:** (TP + TN) / (TP + TN + FP + FN)≈ ***0.7222***
- **Precision (Positive Predictive Value):** TP / (TP + FP) = ***0.6***
- **Recall (Sensitivity, True Positive Rate):** TP / (TP + FN) = ***0.5***
- **Specificity (True Negative Rate):** TN / (TN + FP) ≈ ***0.8333***
- **F1 Score:** 2 * (Precision * Recall) / (Precision + Recall) ***≈ 0.5455***

Conclusions

- **The decision tree model has the highest classification accuracy with a score of 0.9035714285714287**
- **Success rate for payloads between 0 - 5000kg are higher than that of payloads between 5000kg - 10000kg**
- **KSC LC-39A is the launch site with the highest success ratio with 76.9% success rate and a failure rate of 23.1%.**
- **The first successfull landing was on 22nd December, 2015.**
- **Launch success rate has increased significantly since 2013 and has stablised since 2019, potentially due to advance in technology and lessons learned.**

Appendix

```
# Your existing code for model scores
models = {'KNeighbors': knn_cv.best_score_,
          'DecisionTree': tree_cv.best_score_,
          'LogisticRegression': logreg_cv.best_score_,
          'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm, 'with a score of', models[bestalgorithm])

# Your existing code for printing best hyperparameters
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
elif bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
elif bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
elif bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)

# Plotting the bar chart
plt.bar(models.keys(), models.values(), color=['blue' if model != bestalgorithm else 'green' for model in models.keys()])
plt.xlabel('Models')
plt.ylabel('Cross-validation Score')
plt.title('Model Comparison')
plt.show()
```



- **Code used to visualize a bar chart of all the models and their classification accuracy.**

- **Results**

Thank you!