IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Luca Galliani
02-10-2022

# Outline

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

# Executive Summary

- Summary of methodologies split by weeks
  1. Data collection – Webscraping - Data wrangling
  2. SQL
  3. Data visualization
  4. Predictive Analysis using Logistic Regression, SVN, Decision Tree, KNN
- Summary of results
  - All launch sites on a map
  - the success/failed launches for each site on the map
  - Calculate the distances between a launch site to its proximities

# Introduction

- Project background and context
  - SpaceX  Falcon 9 first stage Landing Prediction

- Problems you want to find answers
  - We can evaluate the cost of a mission that land
  - We can evaluate a competitor costs to enter in that business

Section 1

# Methodology

# Methodology

## Executive Summary

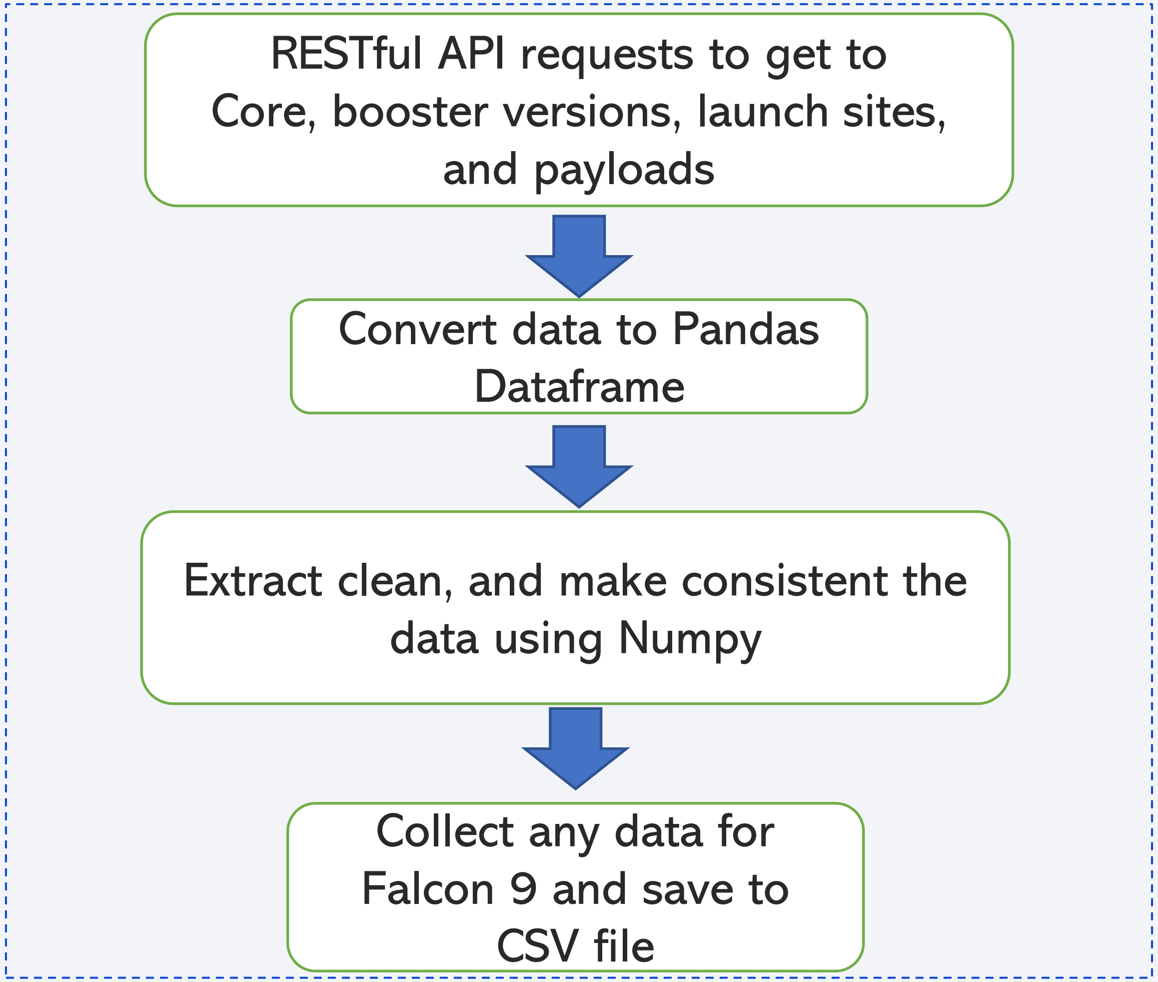
- Data collection methodology:
  - Extract data via SpaceX-API and using webscraping from SpaceX Wikipedia page.
- Perform data wrangling
  - Check integrity of data and replace the missing values using PayloadMass
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Execute SQL queries to answer some questions.
- Perform interactive visual analytics using Folium and Plotly Dash
  - Visual maps and interactive dashboard (Analysis by Site, Payload and booster version)
- Perform predictive analysis using classification models
  - Logistic Regression, SVM, Decision Tree, KNN

# Data Collection

- SpaceX REST API
  - RESTful Interface
  - Core Data
  - Booster Version
  - Launch Site Data
  - Payload Data
- Webscraping of SpaceX Wikipedia
  - Wget
  - BeautifulSoup
- Jupyter Notebooks folders

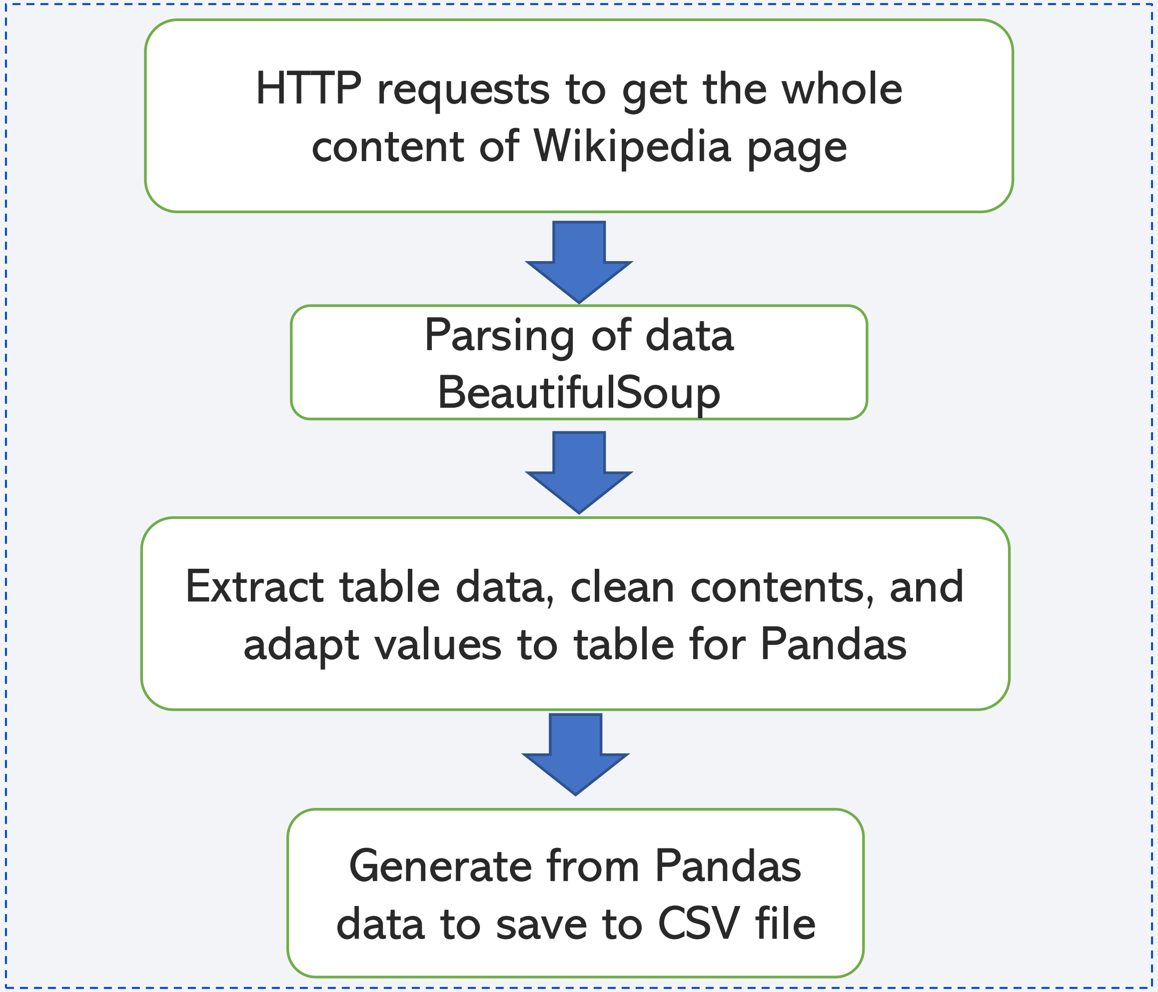
# Data Collection – SpaceX API

- Used SpaceX-API to collect data via RESTful Interface (collecting core data, booster version, launch site data, and payload data)
- Used Python for collect the data and the libraries: Pandas and Numpy
- Here the link to GitHub of my SpaceX API calls Jupyter 1_1 Notebook



# Data Collection - Scraping

- Get data from Wikipedia website (http request), using Python to handle data and export to CSV
- Used Python for collect the data and the libraries: Pandas and BeautifulSoup
- [Here the GitHub URL of the webscraping data from Wikipedia]



# Data Wrangling

- Collected CSV files in the first notebook and analysis of missing data in the collection.
- Missed data in payload mass values
- Calculate the mean value to replace the missing data.
- Used Python for handle, calculate the mean value from the existing data and used the libraries: Pandas and Numpy

- Here the GitHub URL of data wrangling

# EDA with Data Visualization

- Used Python for handle the data in the CSV file, used the libraries: Pandas, Numpy, Matplotlib, and Seaborn
- Use table for create charts related to:
  - FlightNumber vs. PayloadMass
  - FlightNumber vs LaunchSite
  - Payload and Launch Site
  - relationship between success rate of each orbit type
  - FlightNumber and Orbit type
  - Payload and Orbit type
  - launch success yearly trend
- Here the GitHub URL of EDA with data visualization notebook

# EDA with SQL

- Using dataset to generate queries:
  - Tasks 1-2: launch sites name and filter for names that begin with “CAA”
  - Tasks 3-4: Calculate the total and the average of PAYLOAD_MASS from Nasa and F9
  - Task 5: Find the oldest date stored
  - Task 6: Show name of boosters filtered by specific values
  - Task 7: Show total of success and failure missions
  - Task 8: Find the names of the booster_versions which carried the maximum payload mass
  - Task 9-10: Show landing outcomesin drone ship using filters from failures and date
- Here the GitHub URL of EDA with data visualization notebook

# Build an Interactive Map with Folium

- Map object used here:
  - folium.Circle (for highline the saunch sites)
  - folium.Marker (for apply labels in the maps)
  - Extend folium.Marker to marker_cluster (for handle visual effects e.g. green-red color launches)
  - PolyLine (for mesure some distances between specific sites in the map)
- Used Python for handle data to generate the maps using the libraries: Pandas and Folium (and related Folium plugins)
- Here the GitHub URL of interactive map with Folium map

# Build a Dashboard with Plotly Dash

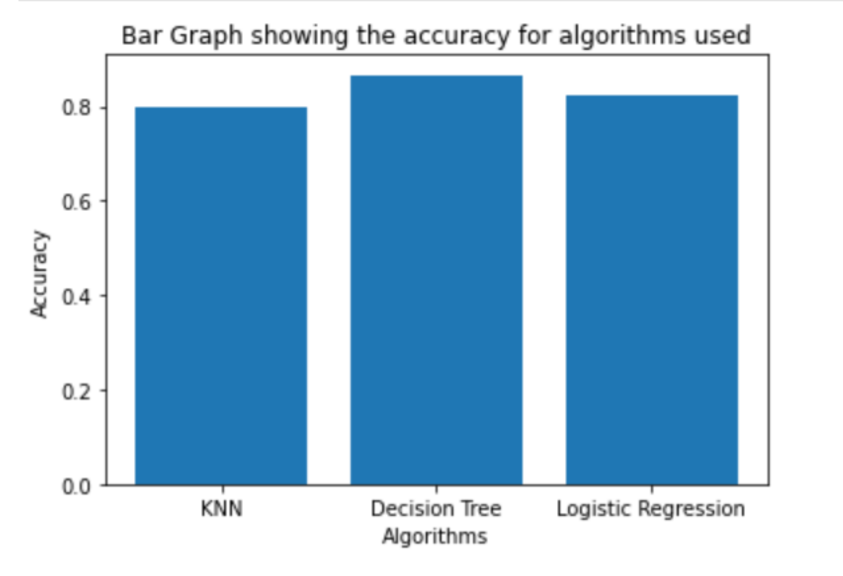
- The app contains two selectors: A dropdown list and a bar slider, the first one is for determinate the site, the second one the mass
- A chart as Pie contains the statistics of success/ the number of successful landing outcome
- Graph scatter show success and failure related to the payload selected from bar slider
- Here the GitHub URL of my Plotly Dash lab

# Predictive Analysis (Classification)

- Handling data for preparation:
  - Encoding
  - Split data from train and test data and scaled using StandardScaler().fit_transform(X)
- Model Building Methods (using Grindsearch):
  - Logistic regression
  - SVM
  - Decision Tree
  - K-Nearest Neighbor
- Use score to compare the better classification method used
- Used Python for use the predictive analysis using the libraries: Pandas, Numpy, matplotlib, seaborn, and sklearn
- [Here the GitHub URL of predictive analysis lab]

# Results

- Exploratory data analysis results
  - Success rate is related to overtime and higher orbits
- Predictive analysis results
  - The better algorithm studied here is
    Decision Tree the score is 0.8666666666666668

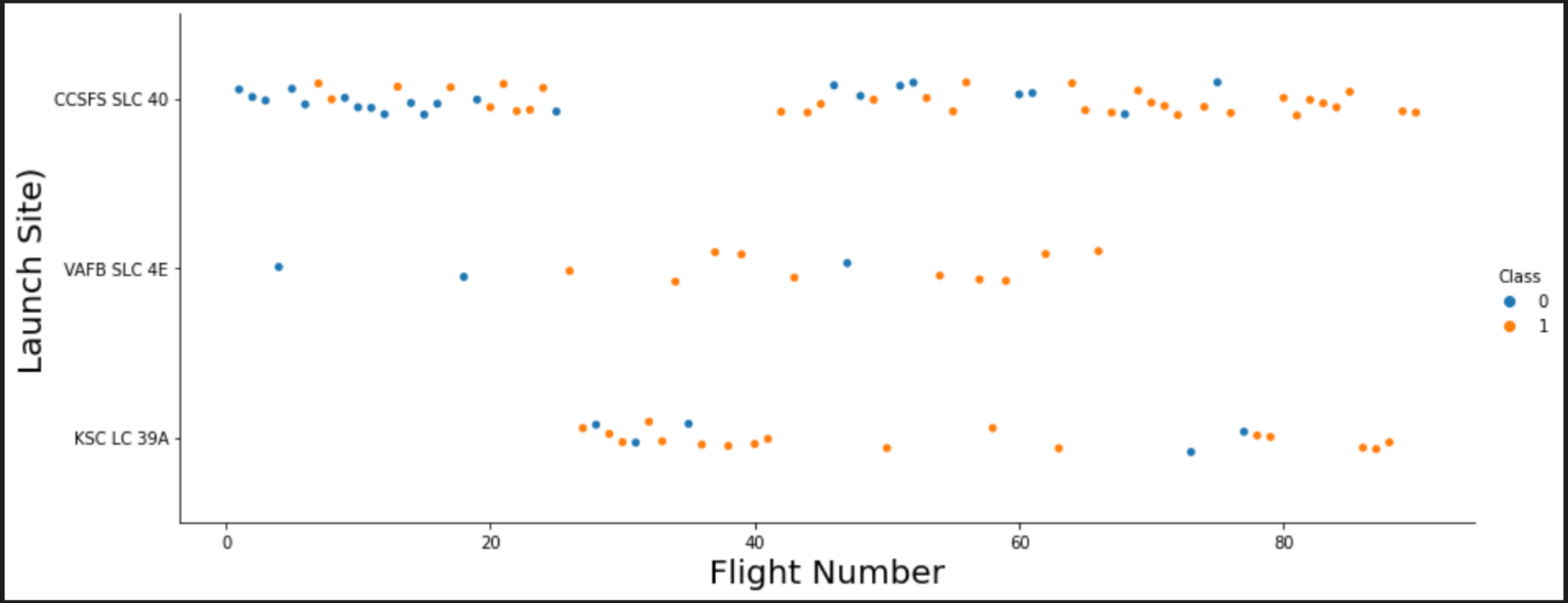


Section 2

# Insights drawn from EDA

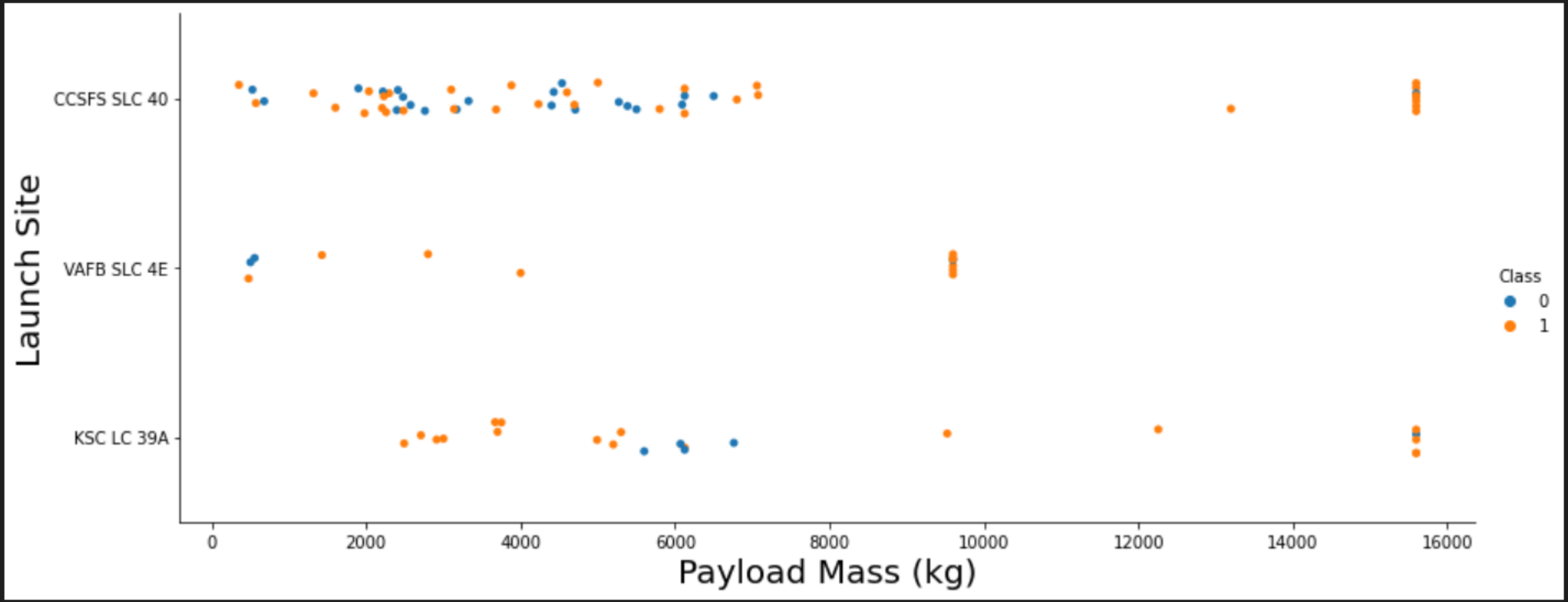
# Flight Number vs. Launch Site

```python
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect=2.5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site)", fontsize=20)
plt.show()
```



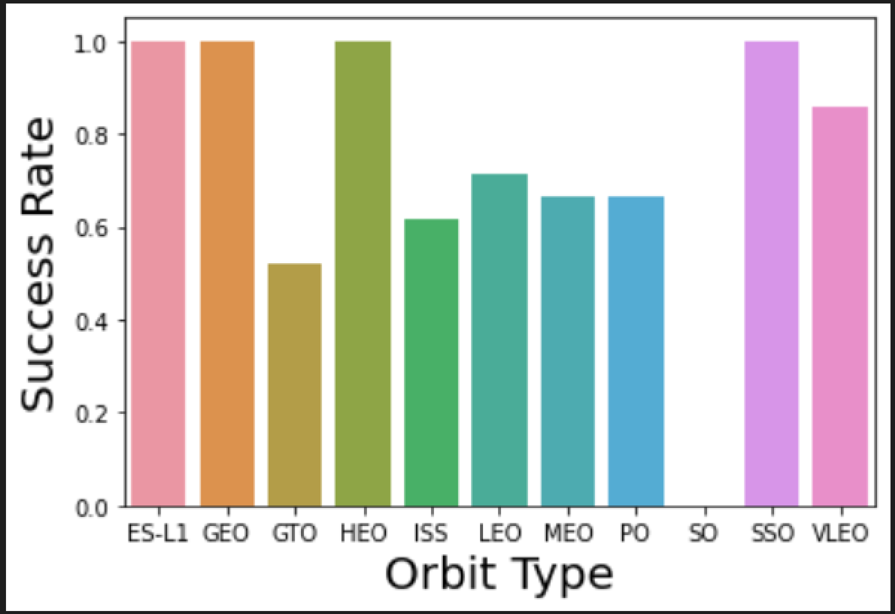
# Payload vs. Launch Site

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect=2.5)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```

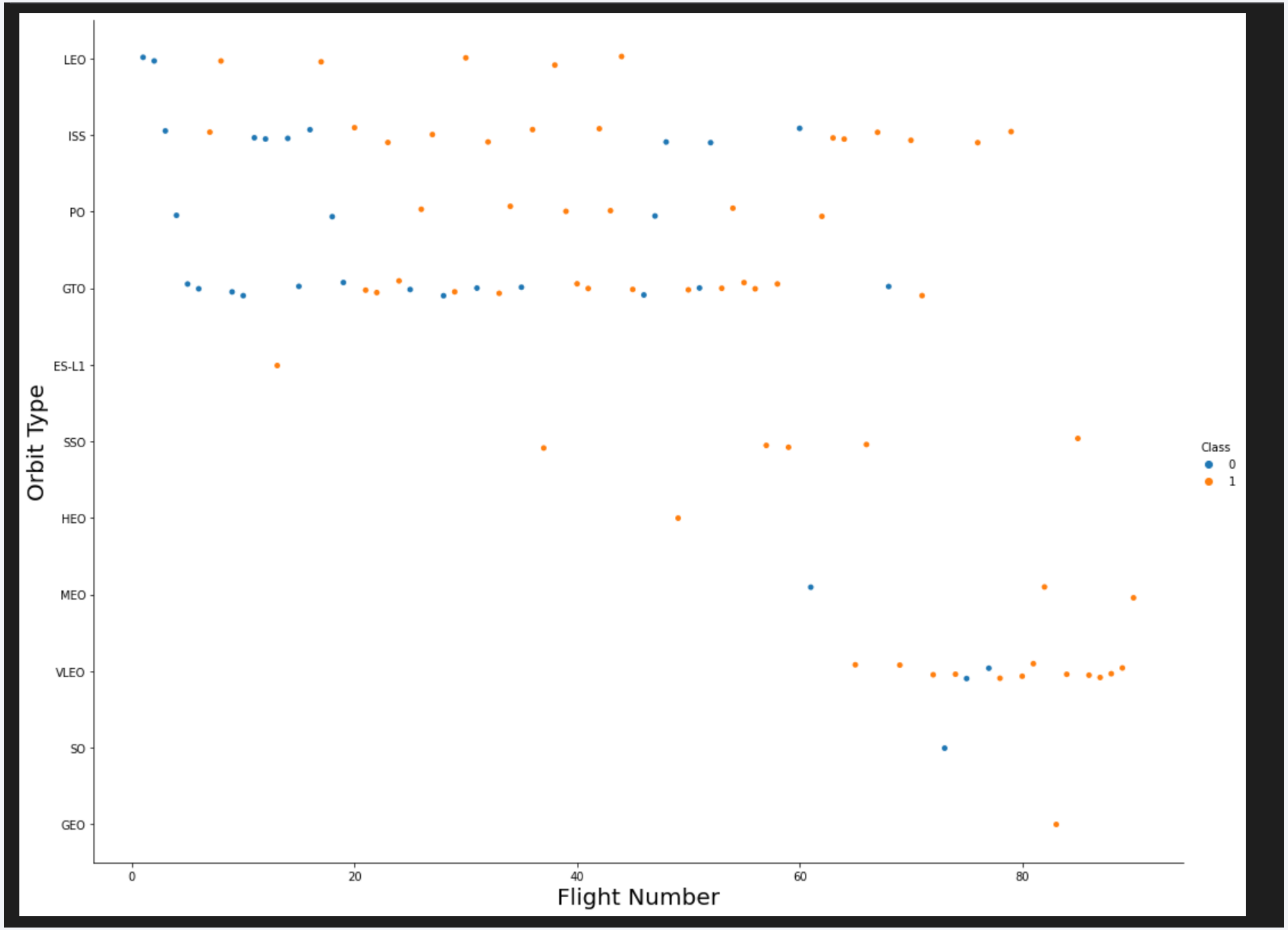


# Success Rate vs. Orbit Type

```
# HINT use groupby method on Orbit column and get the mean of Class column
df_orbit = df.groupby(df['Orbit'], as_index=False).agg({"Class": "mean"})
#df_orbit
sns.barplot(y="Class", x="Orbit", data=df_orbit)
plt.xlabel("Orbit Type", fontsize=20)
plt.ylabel("Success Rate", fontsize=20)
plt.show()
```

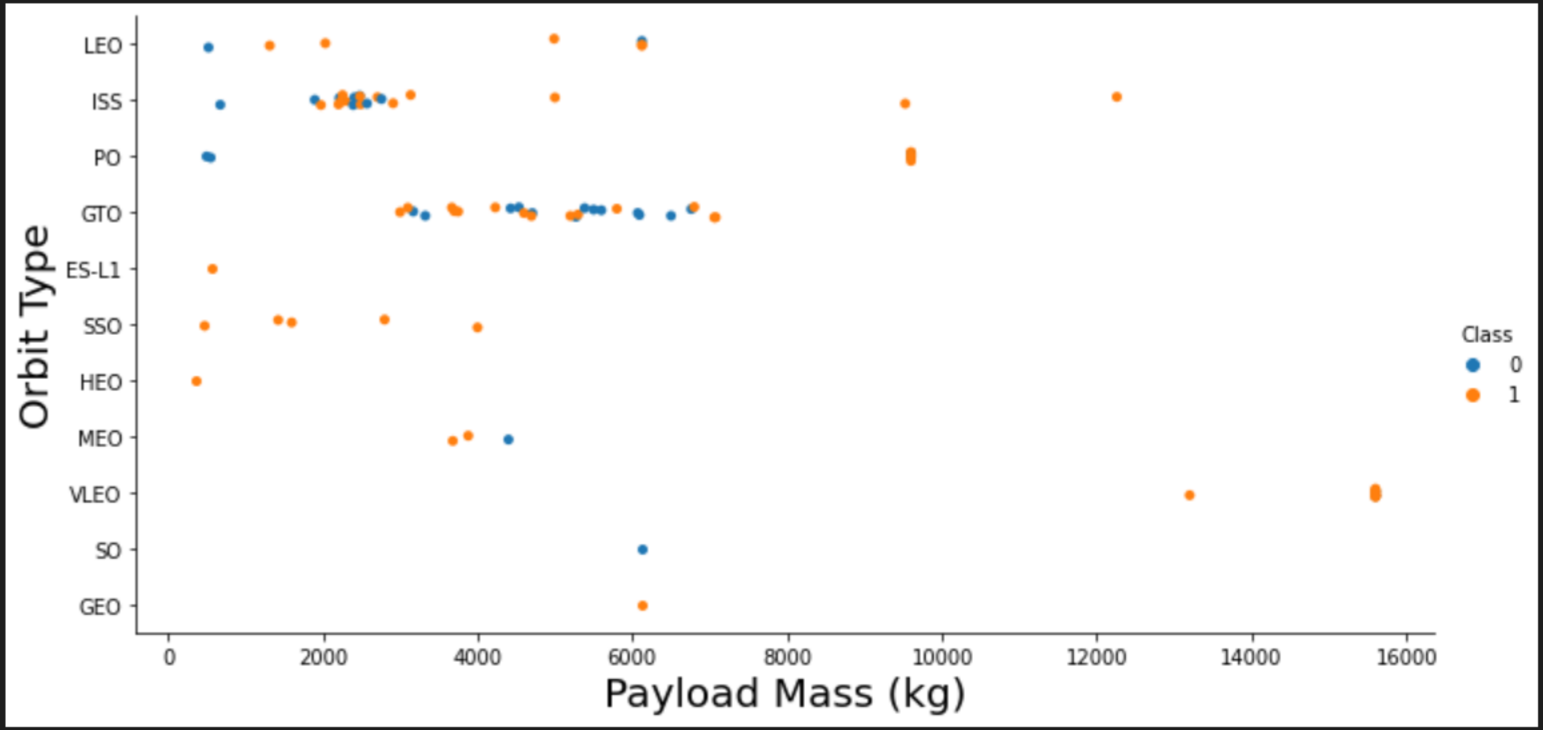


# Flight Number vs. Orbit Type



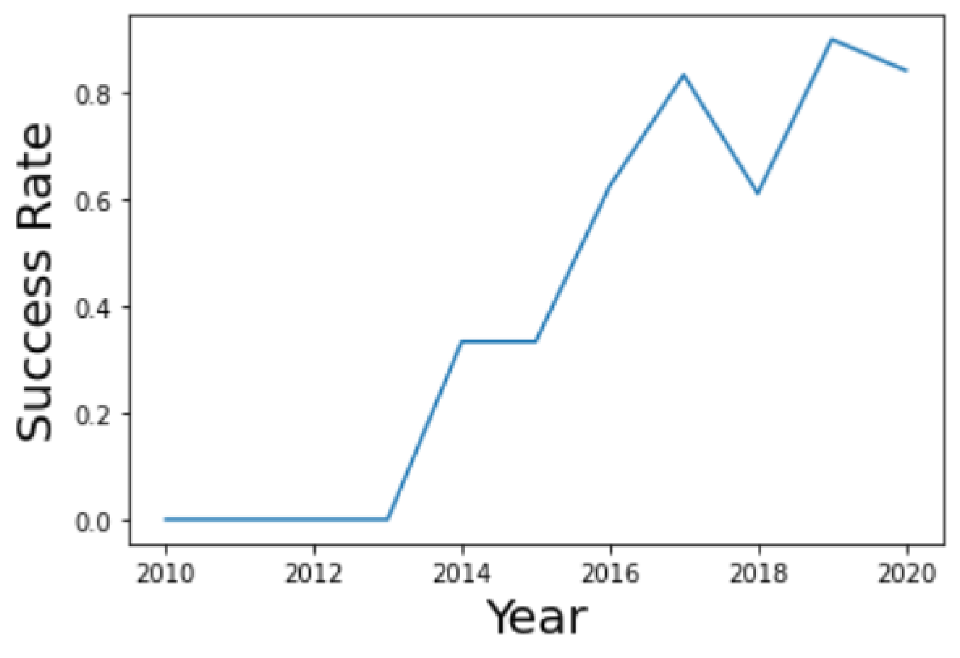
# Payload vs. Orbit Type

```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect=2)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Orbit Type", fontsize=20)
plt.show()
```



# Launch Success Yearly Trend

```python
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
sns.lineplot(y="Class", x="Year", data=df_year)
plt.xlabel("Year", fontsize=20)
plt.ylabel("Success Rate", fontsize=20)
plt.show()
```



# All Launch Site Names

[5]

```
%sql select distinct Launch_Site from  SPACEX_DATASET
```

```
 * ibm_db_sa://ppk46614:***@ea286ace-86c7-4d5b-8580-3fbfa4
Done.
```

| launch_site |
|---|
| CCAFS LC-40 |
| CCAFS SLC-40 |
| KSC LC-39A |
| VAFB SLC-4E |

# Launch Site Names Begin with 'CCA'

```
%sql select * from  SPACEX_DATASET where Launch_Site like 'CCA%' limit 5
```

```
 * ibm_db_sa://ppk46614:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
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 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 00:35:00 | 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

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

```
%sql select sum(PAYLOAD_MASS__KG_) as SUM_OF_PAYLOAD_MASS from SPACEX_DATASET WHERE Customer like 'NASA (CRS)'
```

```
 * ibm_db_sa://ppk46614:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:315
Done.
```

| sum_of_payload_mass |
|---|
| 45596 |

# Average Payload Mass by F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) as AVERAGE_OF_PAYLOAD_MASS from SPACEX_DATASET WHERE Booster_Version like 'F9 v1.1'
```

```
 * ibm_db_sa://ppk46614:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/blud
Done.
```

| average_of_payload_mass |
|---|
| 2928 |

# First Successful Ground Landing Date

```
%sql select min(Date) as oldest_date from SPACEX_DATASET WHERE Landing__Outcome like 'Success (ground pad)'
```

```
 * ibm_db_sa://ppk46614:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:
Done.
```

| oldest_date |
|---|
| 2015-12-22 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select distinct Booster_Version FROM SPACEX_DATASET WHERE Landing__Outcome like 'Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000
```

```
 * ibm_db_sa://ppk46614:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

| booster_version |
|---|
| F9 FT B1021.2 |
| F9 FT B1031.2 |
| F9 FT B1022 |
| F9 FT B1026 |

# Total Number of Successful and Failure Mission Outcomes

```
%sql select substr(Mission_Outcome,1,7) as Mission_Outcome , count(*) as Nr_Of_missions FROM SPACEX_DATASET group by substr(Mission_Outcome,1,7) order by substr(Mission_Outcome,1,7) desc
```

```
 * ibm_db_sa://ppk46614:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

| mission_outcome | nr_of_missions |
|---|---|
| Success | 100 |
| Failure | 1 |

# Boosters Carried Maximum Payload

```
%sql select distinct Booster_Version FROM SPACEX_DATASET where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) FROM SPACEX_DATASET)
```

```
 * ibm_db_sa://ppk46614:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

| booster_version |
|---|
| F9 B5 B1048.4 |
| F9 B5 B1048.5 |
| F9 B5 B1049.4 |
| F9 B5 B1049.5 |
| F9 B5 B1049.7 |
| F9 B5 B1051.3 |
| F9 B5 B1051.4 |
| F9 B5 B1051.6 |
| F9 B5 B1056.4 |
| F9 B5 B1058.3 |
| F9 B5 B1060.2 |
| F9 B5 B1060.3 |

# 2015 Launch Records

```
%sql select distinct Landing__Outcome, Booster_Version, Launch_Site FROM SPACEX_DATASET where Landing__Outcome='Failure (drone ship)'
```

```
 * ibm_db_sa://ppk46614:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

| landing__outcome | booster_version | launch_site |
|---|---|---|
| Failure (drone ship) | F9 FT B1020 | CCAFS LC-40 |
| Failure (drone ship) | F9 FT B1024 | CCAFS LC-40 |
| Failure (drone ship) | F9 v1.1 B1012 | CCAFS LC-40 |
| Failure (drone ship) | F9 v1.1 B1015 | CCAFS LC-40 |
| Failure (drone ship) | F9 v1.1 B1017 | VAFB SLC-4E |

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

```
%sql select Landing__Outcome, count(*) as Nr_of FROM SPACEX_DATASET where Date between '2011-06-04' and '2017-03-20' group by Landing__Outcome order by 2
```

```
 * ibm_db_sa://ppk46614:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

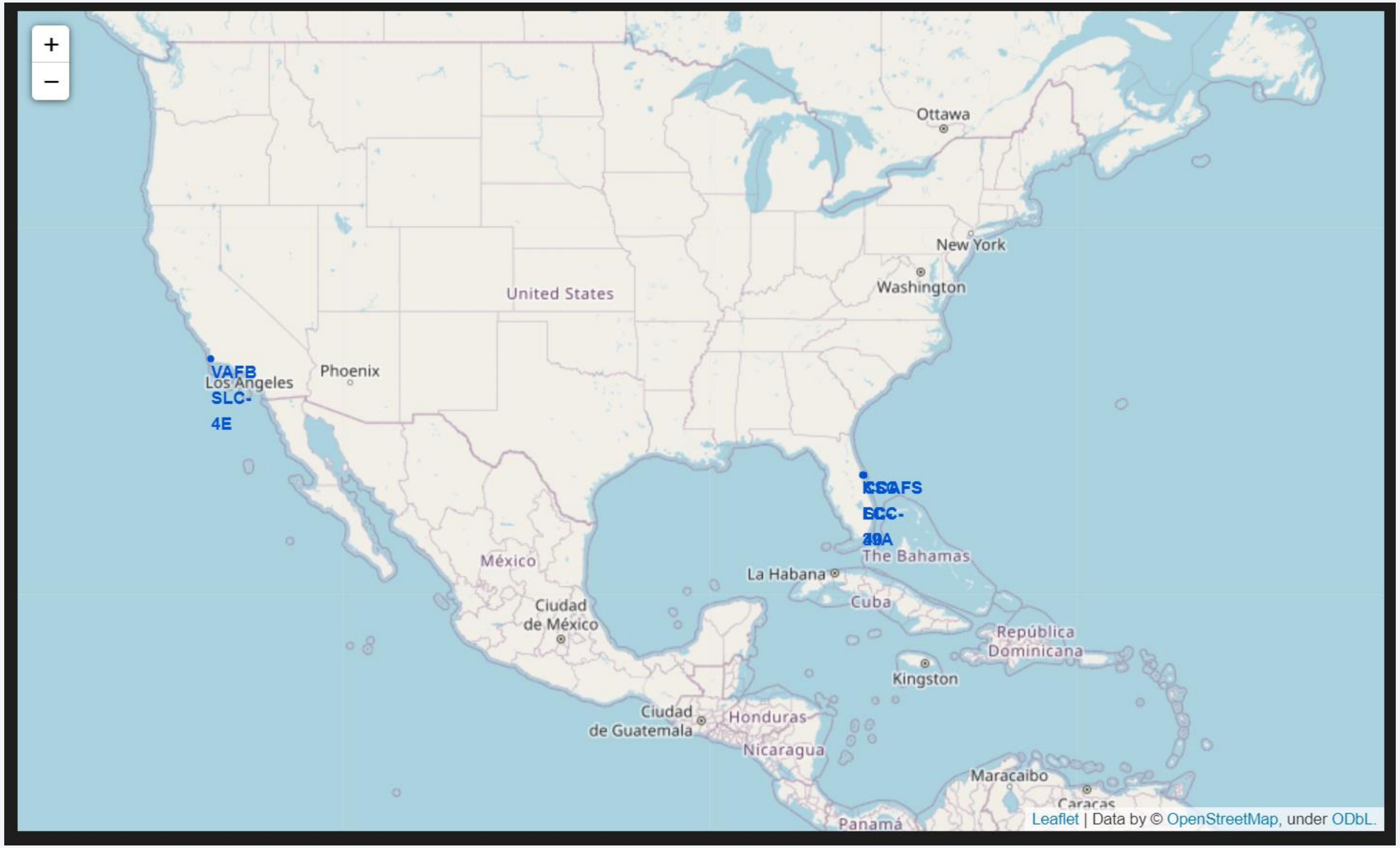
| landing__outcome | nr_of |
|---|---|
| No attempt | 10 |
| Failure (drone ship) | 5 |
| Success (drone ship) | 5 |
| Controlled (ocean) | 3 |
| Success (ground pad) | 3 |
| Uncontrolled (ocean) | 2 |
| Precluded (drone ship) | 1 |

Section 3

# Launch Sites Proximities Analysis

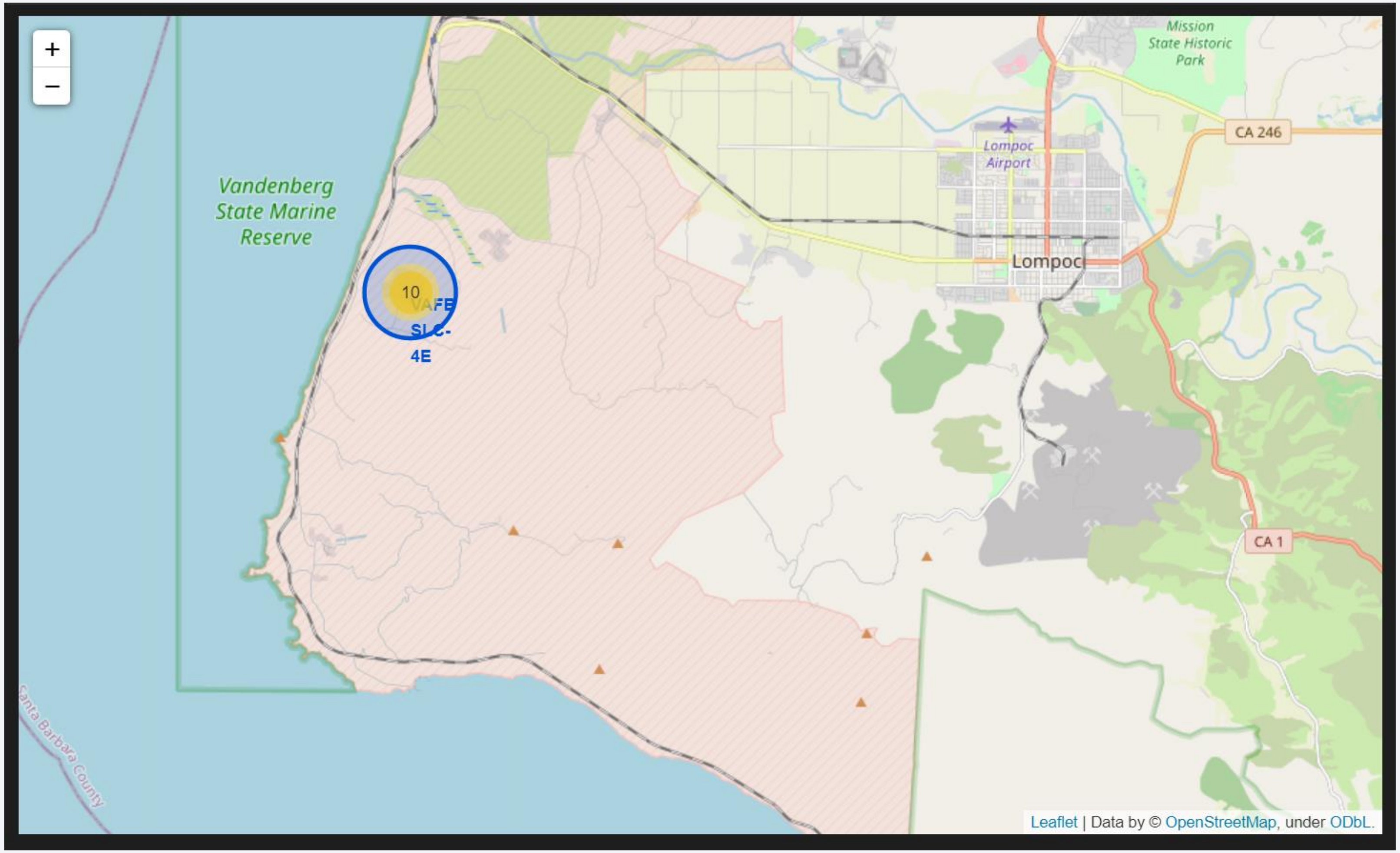
# Folium Map Screenshot 1: Launch sites

- In blue are East and West coast of USA



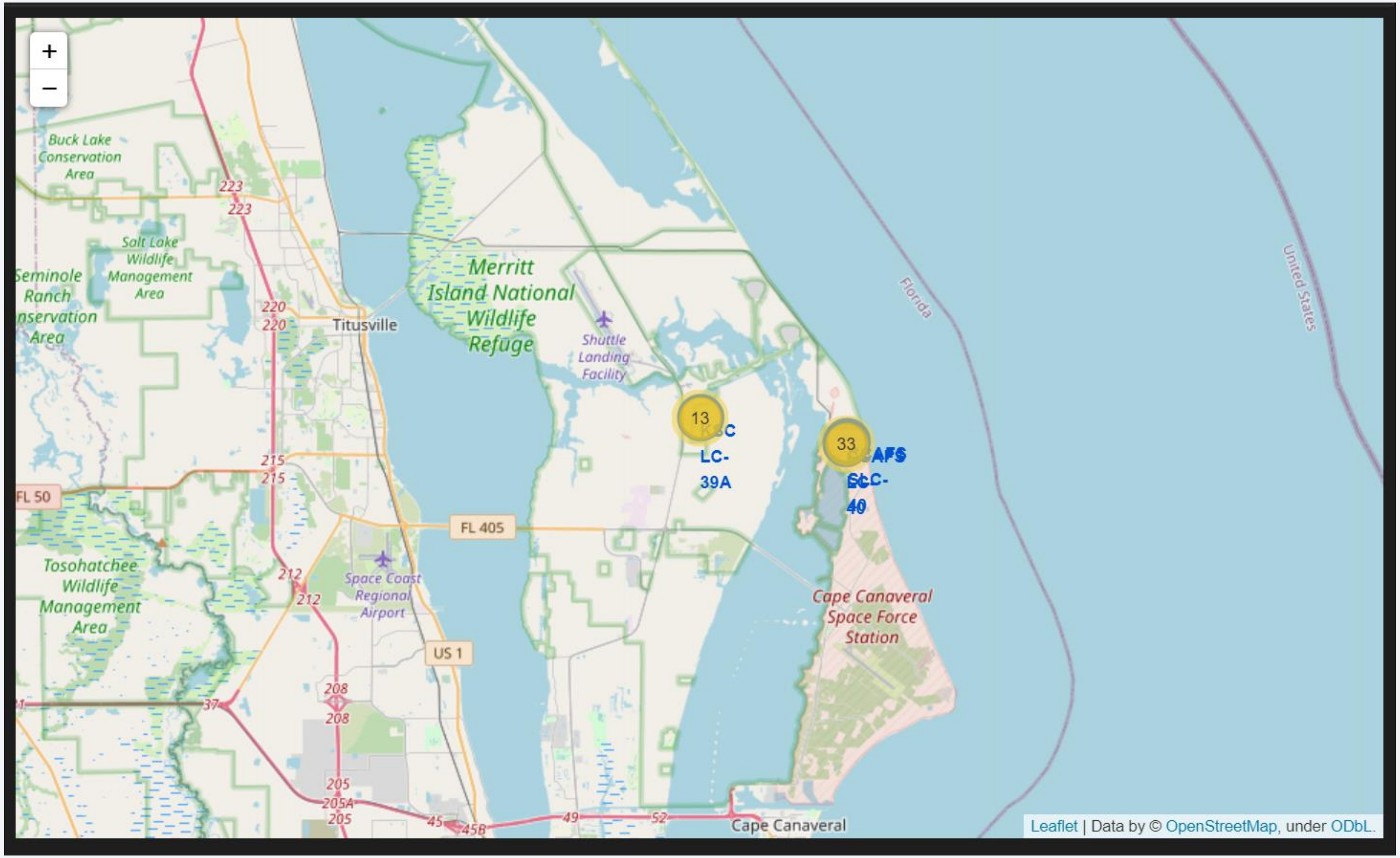
# Folium Map Screenshot 2: West Coast

- Zoom detail for West Coast close the base of Vandenberg



# Folium Map Screenshot 3: East Coast

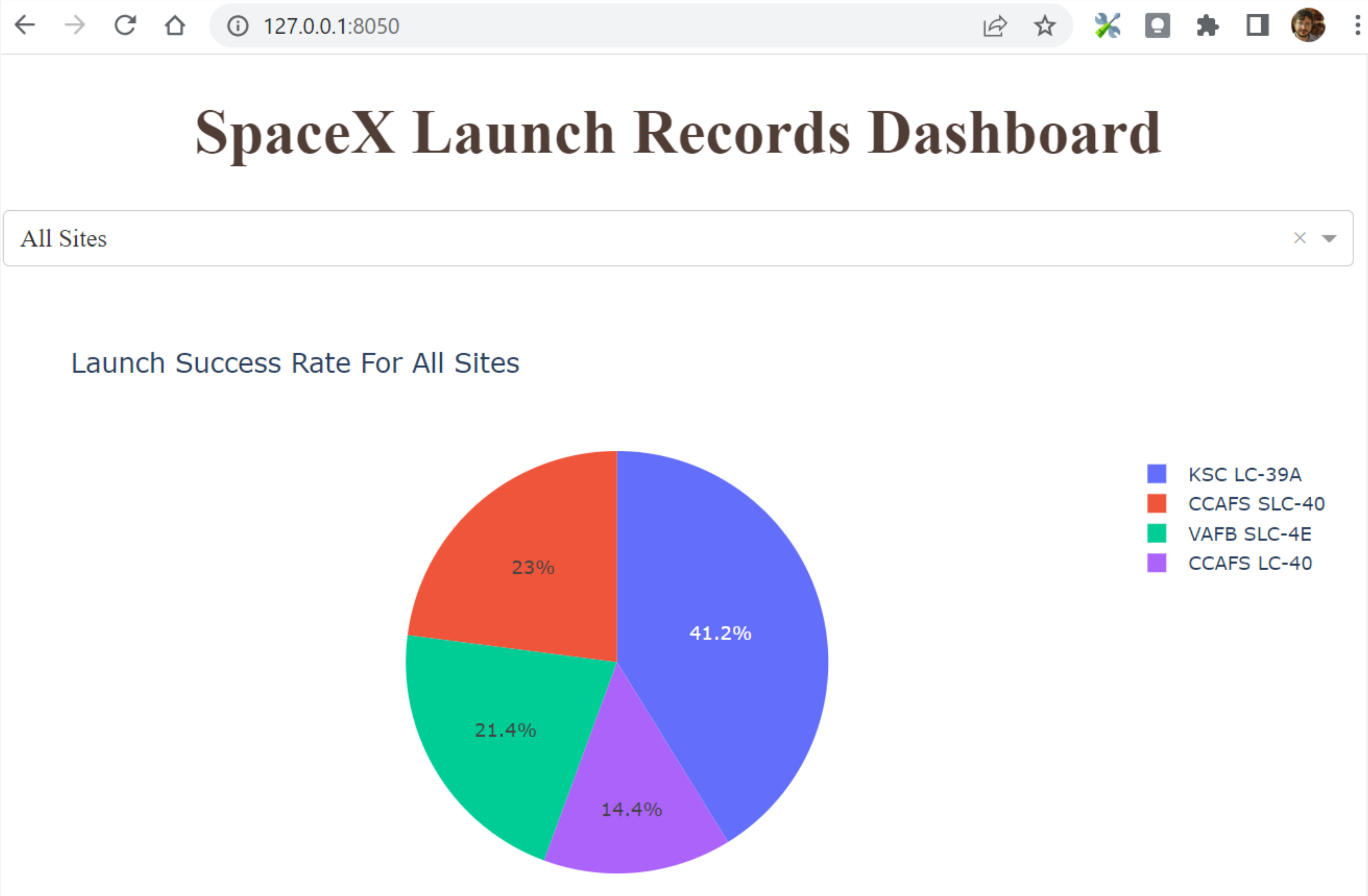
- Zoom detail from East Coast close to Cape Canaveral



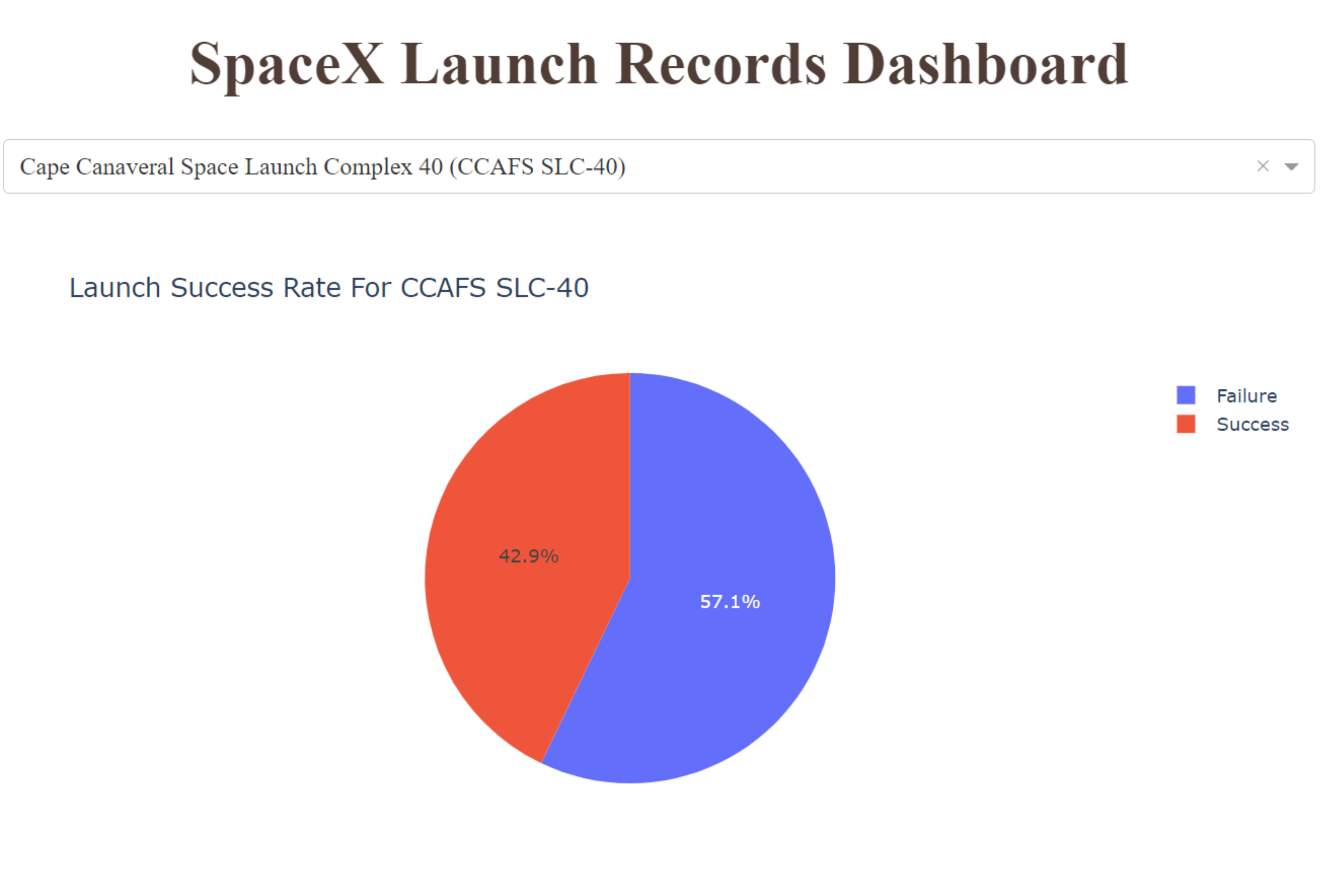
Section 4

# Build a Dashboard with Plotly Dash

# Dashboard Screenshot 1: Success rate All sites

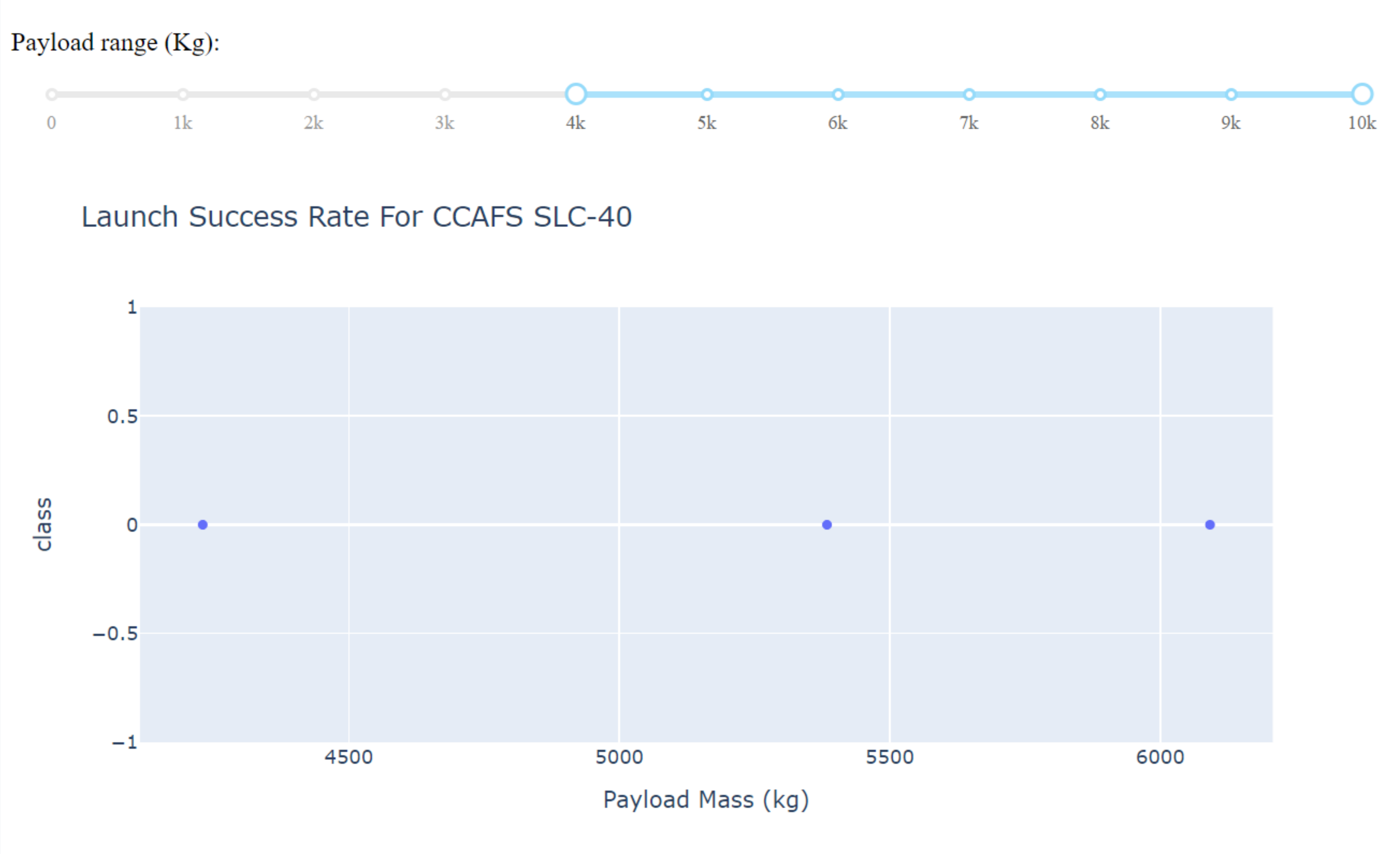


# Dashboard Screenshot 2: Success rate CCAFS SLC-40



# Dashboard Screenshot 3: Success rate for payload rage

- Successful payload range kg: 4000



Section 5

# Predictive Analysis (Classification)

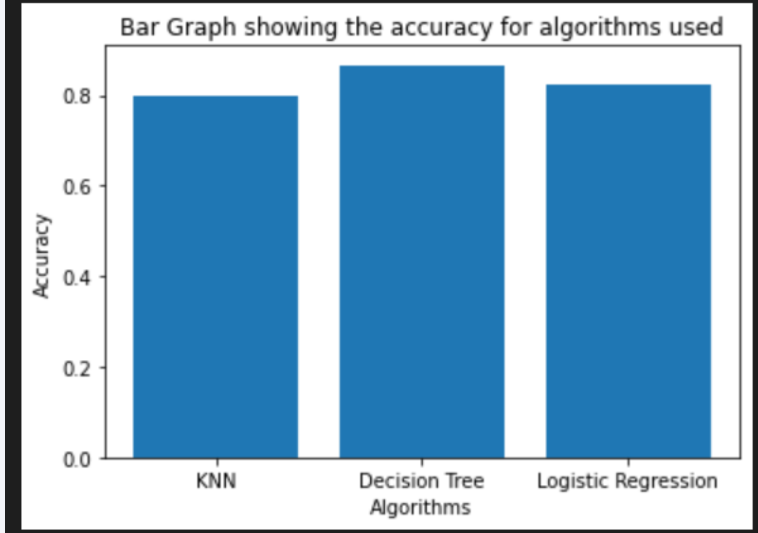
# Classification Accuracy

The better algorithm studied here is Decision Tree the score is 0.8666

```
The better algorithm studied here is Decision Tree the score is 0.8666666666666668
Best Params is : {'criterion': 'gini', 'max_depth': 2, 'max_features': 'auto', 'min_sa

{'KNN': 0.8,
 'Decision Tree': 0.8666666666666668,
 'Logistic Regression': 0.8222222222222222}
```

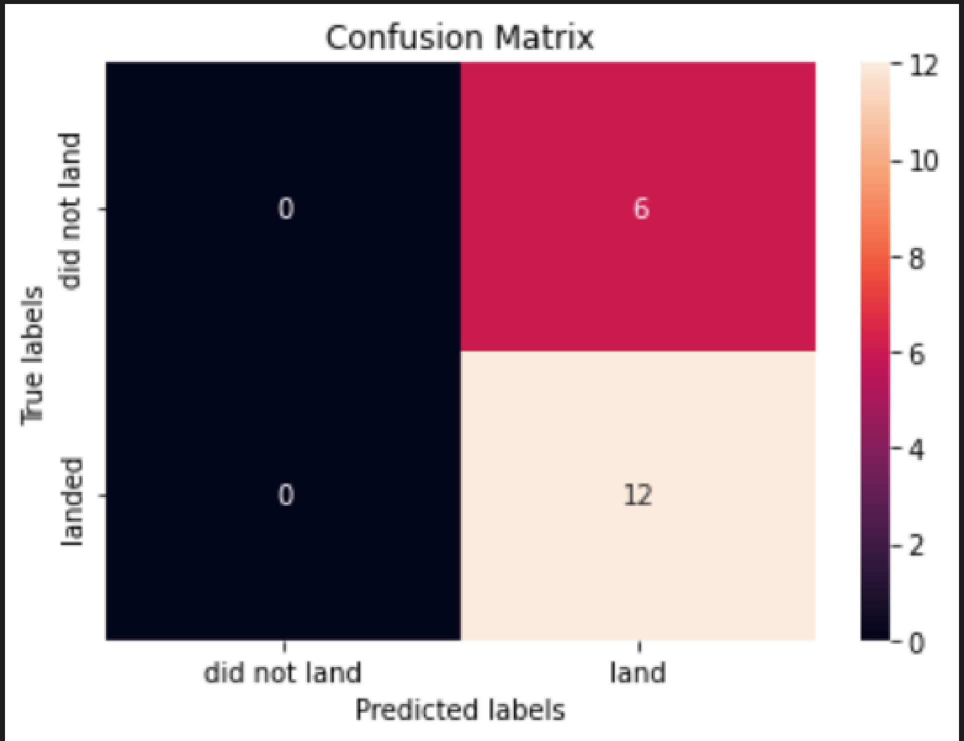
```python
plt.bar(matches.keys(), matches.values())
plt.title("Bar Graph showing the accuracy for algorithms used")
plt.ylabel("Accuracy")
plt.xlabel("Algorithms")
plt.show()
```



# Confusion Matrix

- True positives: 12
- True Negatives: 0
- False Positives: 6
- False Negatives: 0

```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



# Conclusions

- Prediction with Decision Tree is the better used for that analysis
- All models used are close to the same accuracy
- In any models are not presents only true positives
- In any models we have never false negative
- All models shown 12 true positives

# Appendix

- All Python modules used for that presentations are specified in the related slides.
- I used as support Visual Studio Code for the code, for use the Jupyter Notebooks, and also for the interactive Plotly dashboard app.
- All documents used for that presentation are stored in my public GitHub repository at the following link:
  https://github.com/tarrasque/Applied-Data-Science/tree/main/Final%20exam/Jupyter_notebooks
- Thank you for your interest on my presentation!

Thank you!