



IBM Developer  
SKILLS NETWORK

# Winning Space Race with Data Science

Isaac A Abiola  
February 05, 2025



# Outline

---

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

# Executive Summary

---

- We used 4 methods to predict the the probability of a success launch of a recock. The methods are the binary logistic regression model, support vector machine (SVM), Decision tree algorithm and k-nearest neighbor method.
- In the data we examined we observed the following:
  - The predictive accuracy as measured by the score, of the binary logistic regression, the support vector machine, and the K-Nearest Neighbor model on the test data are the same. Thus none is preferred over the other in this study.
  - The confusion matrix of the three models are also the same which further reinforces the fact that the models performs similarly.
  - The tree model though easy to interprete and explain is not the preferred choice here because its performance in the test sample is not as good as the other models.
- ...

# Introduction

---

## Background

- **SpaceX** is a private Space Exploration company. It designs, manufactures, and launches advanced rockets and spacecraft into various earth orbits. It competes with other companies in launching rockets/spacecrafts into orbits. The competitive advantage of SpaceX over its competitors is its ability to reuse the first stage of its Falcon 9 booster rockets, As such, based on this, SpaceX is in a position to quote lower launch price than its competitors, if they can reuse the captured booster from the launch.

## Problem Statement

The business problems emanating from the above are as follows:

- First SpaceX is interested in assessing and predicting the probability of capturing the first stage of Falcon 9 booster rocket following a launch into orbit.
- Second, given the four launch sites they can launch from, they are interested in determining/identifying the launch site which is most likely to ensure a successful recapture of the booster rocket.



Section 1

# Methodology

# Methodology

---

## Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

# Data Collection

---

- Describe how data sets were collected.
  - Data was collected from sites on the internet using web scrapping applications
- You need to present your data collection process use key phrases and flowcharts

# Data Collection – SpaceX API

Github Link:

#1:

[https://github.com/iabiola1979/Course-10----Applied-Data-Science-Capstone/blob/main/Course10\\_iaa.py](https://github.com/iabiola1979/Course-10----Applied-Data-Science-Capstone/blob/main/Course10_iaa.py)

#2:

[jupyter-labs-spacex-data-collection-api-v2\\_iaa.ipynb](https://github.com/iabiola1979/Course-10----Applied-Data-Science-Capstone/blob/main/Course10_iaa.ipynb)

Import Python Libraries:

Requests  
Pandas  
numpy

4 Helper Files to get the following:  
BoosterVersion, LaunchSites  
PayloadData; and CoreData

## TASK #1

- Using request library
  - → Get SpaceX past launch data from url
  - → Response is Jason data
- The Jason data is turned to pandas dataframe by applying `pd.json_normalize()`
- Apply the 4 helper functions to extract BoosterVersion, LaunchSite, PayloadMass, FlightNumber, Date, oRbit, Longitude,latitude, and etc.

## TASK #2

- Filter the data above by removing recods withFalcon1 Booster Version
- The flightNumber is then reset into their location in the list +1
- The last step test for records with missing values

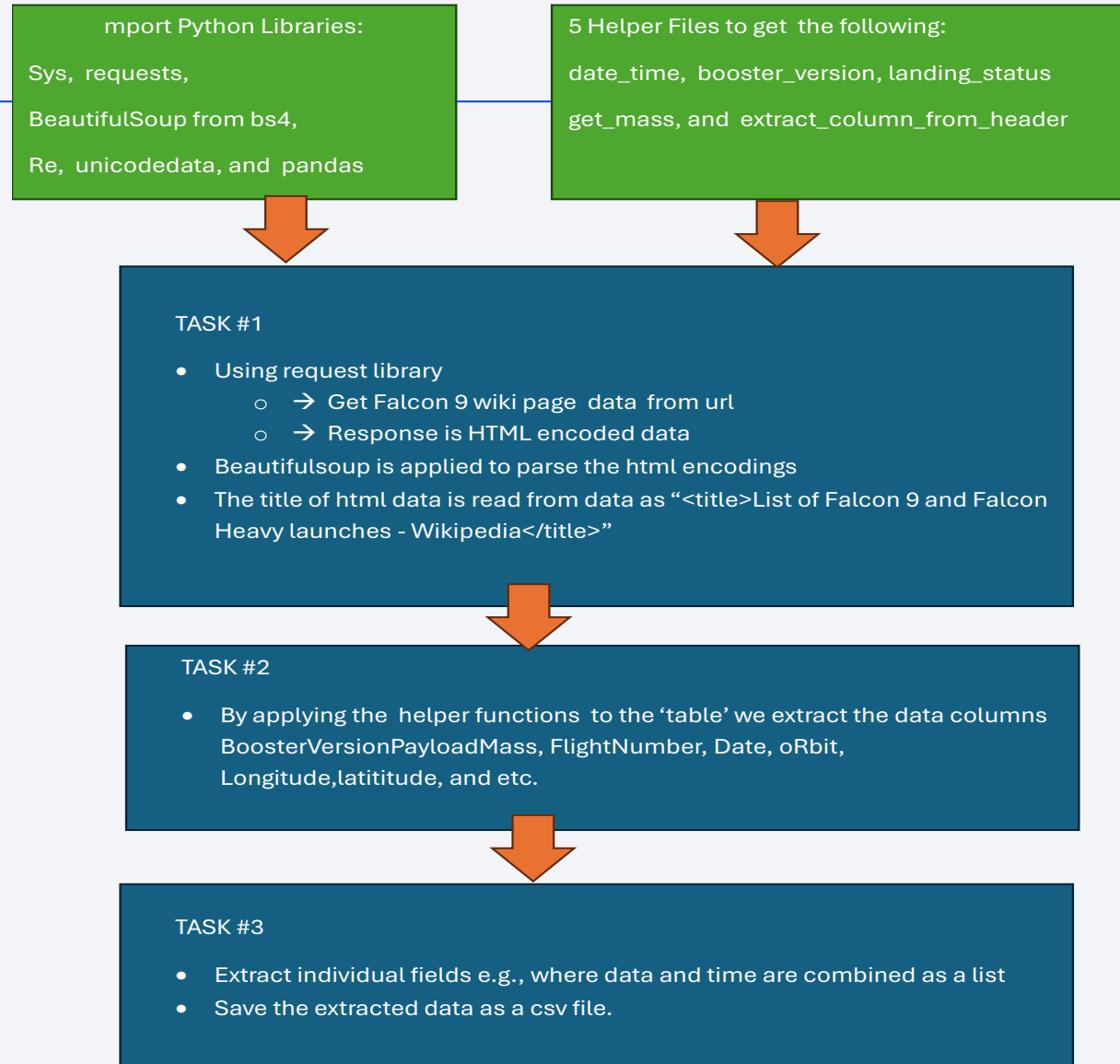
## TASK #3

- Compute the mean value of payload mass from dataframe from Task #2
- Then replace missing value of payload mass with calculated mean value.
- The dataframe from this task is then saved to a “dataset\_part1.csv” file.



# Data Collection - Scrapping

- #1
- <https://github.com/iabiola1979/Course-10----Applied-Data-Science-Capstone/blob/main/Course10-Webscrapping.py>
- #2
- [https://github.com/iabiola1979/Course-10----Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping\\_vscode.ipynb](https://github.com/iabiola1979/Course-10----Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping_vscode.ipynb)



# Data Wrangling

---

## Import Python Libraries:

1. Numpy
2. Pandas

- <https://github.com/iabiola1979/Course-10----Applied-Data-Science-Capstone/blob/main/labs-jupyter-space-launches-data-wrangling-v2.ipynb>

### Step #1

- Using Pandas, we load the data from data from previous step.
- To ensure we loaded the appropriate data we printed out the first five observations. Following , we check each field for percent of missing observations. All fields had 0% missing except "landind pad with 28% missing. Next, we check each column ns. Most fields are numeric float64, others such as 'BoosterVersion', "LaunchSite', 'Outcome' are of object and there were three Boolean type fields as well ('reused', 'legs' and 'gridfins')

### Step #2

- We further analyzed the counts by launchsites, the number of orbit types
- We analyzed the outcome of launches from this we made the determination whether it failed (0) or it was successful (1) . The last step was used to determine Y, the dependent variable. The success rate was 66.67%

# EDA with Data Visualization

---

- Summarize what charts were plotted and why you used those charts
- Visualize the relationship between the following variables in the dataset:  
between Flight Number and Launch Site; between Payload and Launch Site;  
between success rate of each orbit type; between FlightNumber and Orbit  
type; between Payload and Orbit type; and rate the launch success yearly  
trend
- <https://github.com/iabiola1979/Course-10----Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-dataviz-v2.ipynb>

# EDA with SQL

---

- Display the names of the unique launch sites in the space mission
  - CCAFS LC-40 26
  - CCAFS SLC-40 34
  - KSC LC-39A 25
  - VAFB SLC-4E 16
- Display the total payload mass carried by boosters launched by NASA (CRS)
  - (619967,)
- Display average payload mass carried by booster version F9 v1.1
  - 6138.29
- List the date when the first succesful landing outcome in ground pad was acheived.
  - '2015-12-22'
- [https://github.com/iabiola1979/Course-10----Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera\\_sqlite.ipynb](https://github.com/iabiola1979/Course-10----Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqlite.ipynb)

# Build an Interactive Map with Folium

---

- Mark all launch sites on the MAP of USA– Florida and California
- Mark location with success/failure flags
- Mark the launch site by closeness to the coastline, nearest railine and the closest city/town
- <https://github.com/iabiola1979/Course-10----Applied-Data-Science-Capstone/blob/main/lab-jupyter-launch-site-location-v2.ipynb>



# Build a Dashboard with Plotly Dash

---

- Plotted a pie chart that shows the distribution of launches by launch sites
- The second graph plots the relationship between the success rate and payload by the booster version

The pie chart is helpful in showing that the KSC LC 39A launch site produces the most successful launches.

- <https://github.com/iabiola1979/Course-10----Applied-Data-Science-Capstone/blob/main/Module10-DashboardExam.py>

# Predictive Analysis (Classification)

- <https://github.com/iabiola1979/Course-10--Applied-Data-Science-Capstone/blob/main/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb>

Import Python Libraries:

- 1, Numpy
2. Pandas
- 3.seaborn and
- 4 scikit-learn

Helper function leveraged

Plot\_confusion\_Matric()

## Step #1

- Using Pandas, we load the data from data from previous step.
- To ensure we loaded the appropriate data we printed out the first five observations. Following , we check each field for percent of missing observations.
- We then assign the fields into X, the independent variables ; and Y the dependent variable with 1/0 value. The X variable was scaled using Standard Scaler to normalize the input variables. Following, we then apply sklearn.train a-test\_split to assign records into 80% training group and 20% test group

## Step #2

- We further analyzed the above data by running binary logistic regression , Support Vector classifier (SVC) model, decision tree model, and K-Nearest Neighbor model on the data. We used 10-fold cross to evaluate each of this model by evaluating their accuracy and confusion metrics.
- Logistic REgression perform best with 10-fold cross valition score 87.5%

# Results

---

## Exploratory data analysis results

- The plots indicate a significant relationship between payload mass and launch success with the most success from KSC LC -39A launchsite.

## Interactive analytics demo in screenshots

## Predictive analysis results

- The binary logistic regression model is the most predictive. The cross validation score on test sample is 87.5%. Next are the Support vector classification and KNN models each with 83.3% and the last is the tree model with score of 72.2% in test sample.
- The confusion matrix however shows that the binary logistic, SVM and KNN model have the same performance based on the confusion matrix.





Section 2

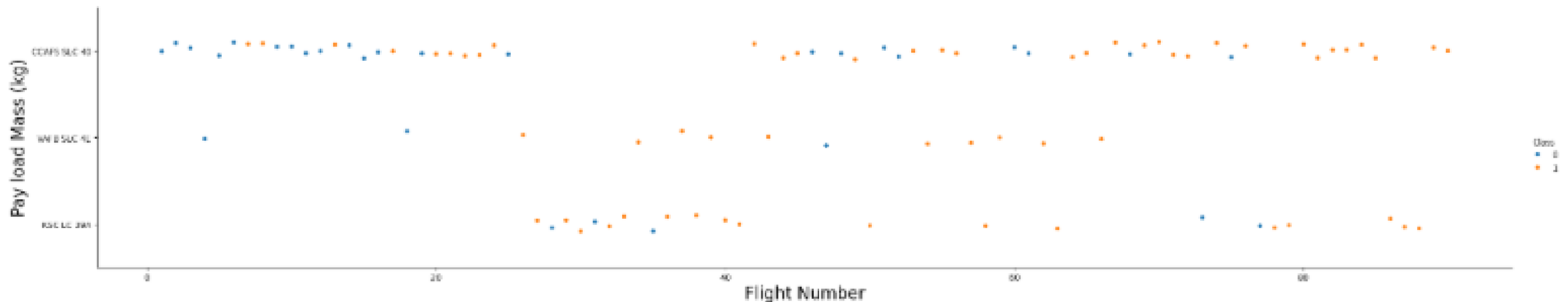
# Insights drawn from EDA



# Flight Number vs. Launch Site

[6]:

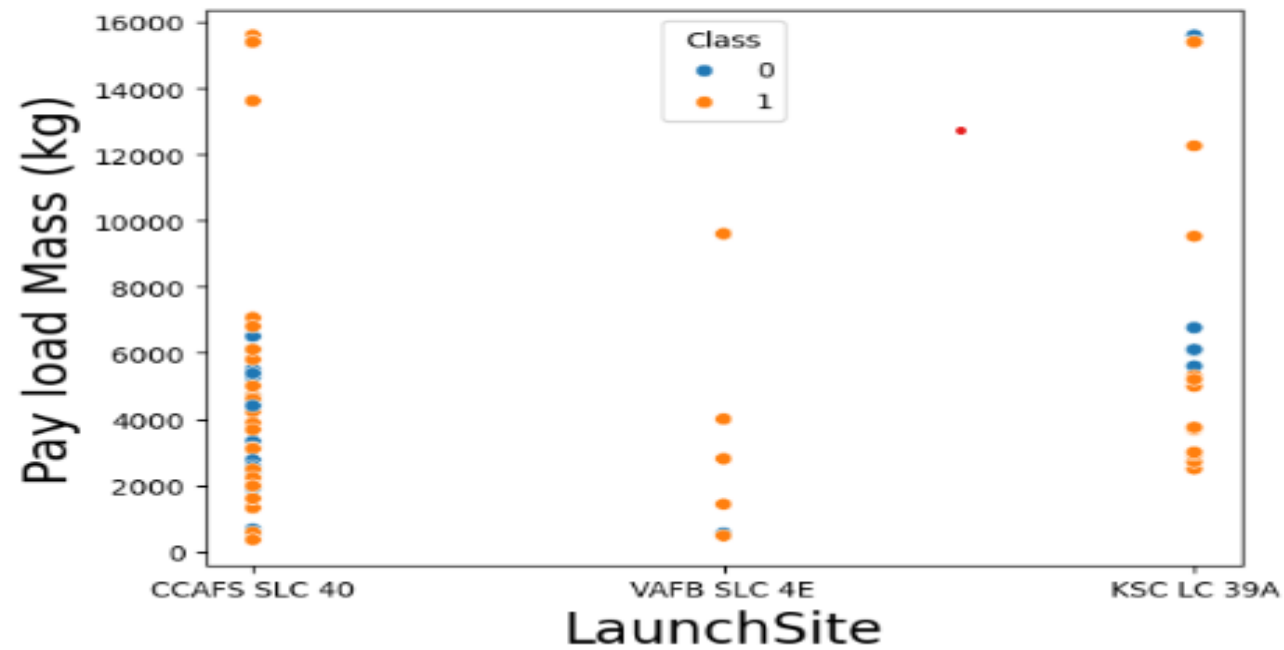
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the Launch site, and hue to be the class variable
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```





# Payload vs. Launch Site

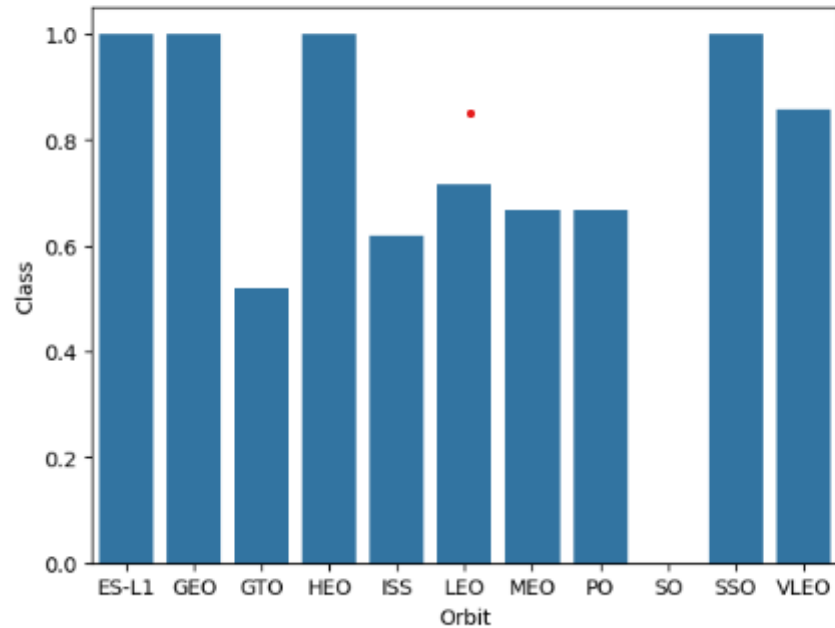
```
4]: # Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the Launch site, and
sns.scatterplot(y="PayloadMass", x="LaunchSite", hue="Class", data=df)
plt.xlabel("LaunchSite", fontsize=20)
plt.ylabel("Pay load Mass (kg)", fontsize=20)
plt.show()
```



# Success Rate vs. Orbit Type

```
[15]: # HINT use groupby method on Orbit column and get the mean of Class column  
aa = df.groupby('Orbit')['Class'].mean().reset_index()  
sns.barplot(data=aa, x='Orbit', y='Class')
```

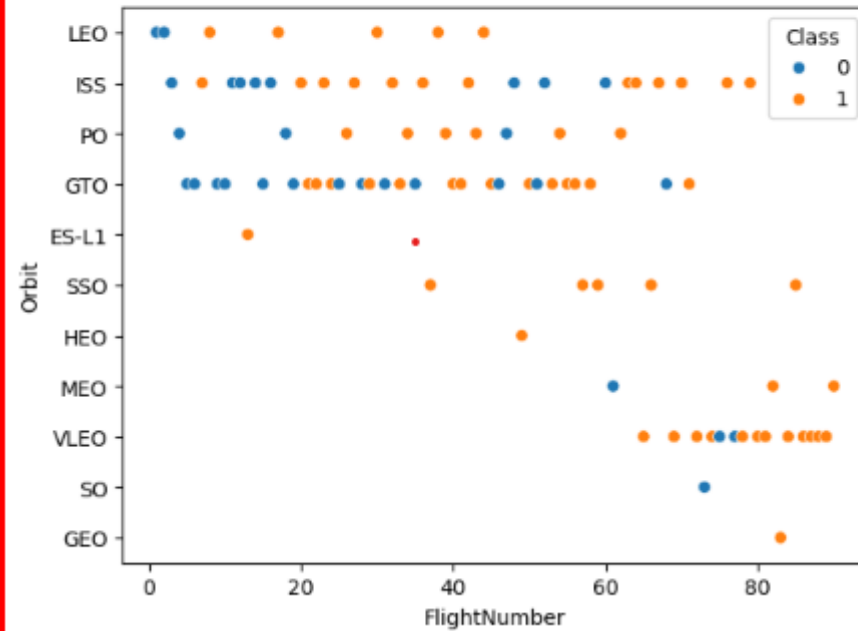
```
[15]: <Axes: xlabel='Orbit', ylabel='Class'>
```



# Flight Number vs. Orbit Type

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value  
sns.scatterplot(data=df, x='FlightNumber', y='Orbit', hue='Class')
```

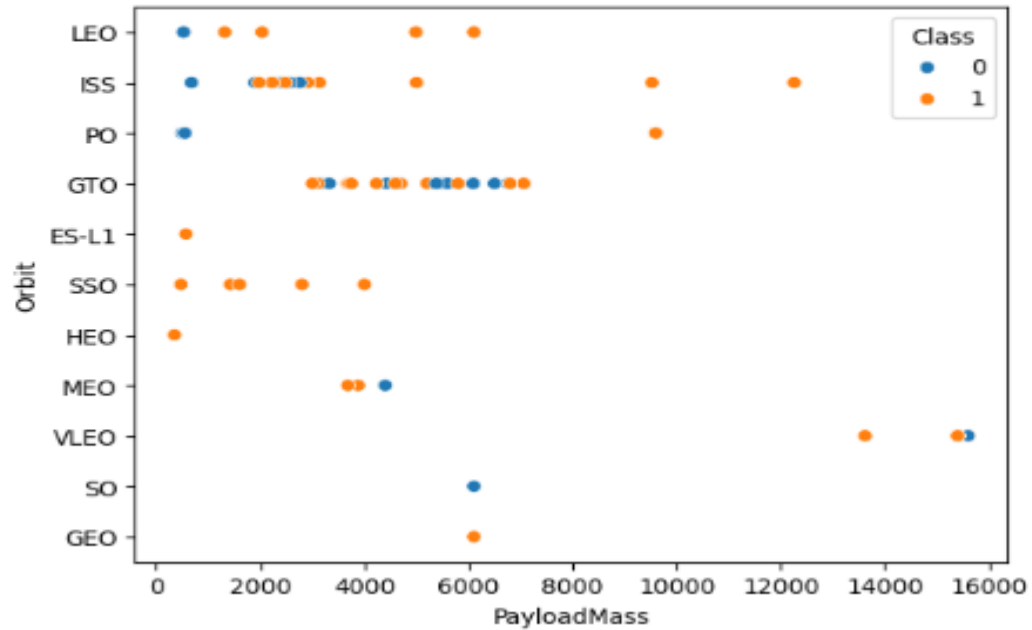
<Axes: xlabel='FlightNumber', ylabel='Orbit'>



# Payload vs. Orbit Type

```
0]: # 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.scatterplot(data=df, x='PayloadMass', y='Orbit', hue='Class')

0]: <Axes: xlabel='PayloadMass', ylabel='Orbit'>
```

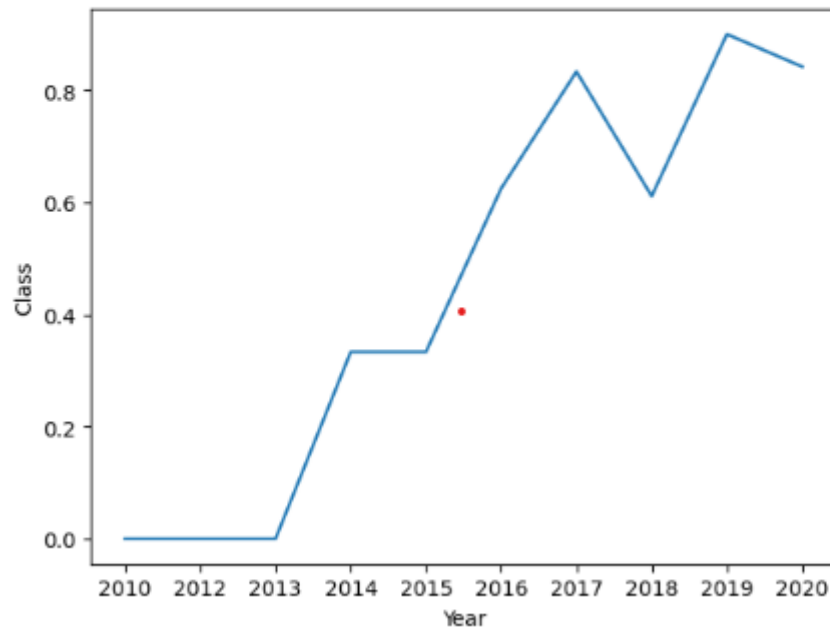


# Launch Success Yearly Trend

```
: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
xx = list(df.Date)
df['Year'] = [str(xx[i])[0:4] for i in range(len(xx))]

xdf = df.groupby('Year')['Class'].mean().reset_index()
xdf
sns.lineplot(data=xdf, x='Year', y='Class', errorbar= ('ci', 95))
```

<Axes: xlabel='Year', ylabel='Class'>





# All Launch Site Names

---

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

```
cursor = conn.cursor()
cursor.execute("SELECT Launch_Site, COUNT(Launch_Site) as LS FROM spacex_table GROUP BY Launch_Site" )
result = cursor.fetchall()

for k, v in result:
    print(k, v)
```

```
CCAFS LC-40 26
CCAFS SLC-40 34
KSC LC-39A 25
VAFB SLC-4E 16
```

# Launch Site Names Begin with 'CCA'

---

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

```
cursor = conn.cursor()
cursor.execute("SELECT * FROM spacex_table WHERE Launch_Site LIKE ? LIMIT 5", ('%CCA%',) )
result = cursor.fetchall()

for k in result:
    print(k)
```

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

# Total Payload Mass

---

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

```
:  
cursor = conn.cursor()  
cursor.execute("SELECT sum(PAYLOAD_MASS__KG_) FROM spacex_table" )  
result = cursor.fetchall()  
  
for k in result:  
    print(k)
```

(619967,)

# Average Payload Mass by F9 v1.1

---

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

```
cursor = conn.cursor()
cursor.execute("SELECT avg(PAYLOAD_MASS_KG_) FROM spacex_table" )
result = cursor.fetchall()

for k in result:
    print(k)
```

```
(6138.287128712871,)
```

# First Successful Ground Landing Date

---

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

*Hint: Use min function*

```
4]: cursor = conn.cursor()
cursor.execute("SELECT MIN(DATE) FROM spacex_table Where (Mission_Outcome == 'Success') & (Landing_Outcome == 'Success (ground pad)')")
result = cursor.fetchall()

for k in result:
    print(k)

('2015-12-22',)
```



## Successful Drone Ship Landing with Payload between 4000 and 6000

---

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
cursor = conn.cursor()
cursor.execute("SELECT Booster_Version FROM spacex_table Where ((4000 < PAYLOAD_MASS_KG_) and (PAYLOAD_MASS_KG_ < 6000) ) & (Landing_Outcome=='Success'")
result = cursor.fetchall()

for k in result:
    print(k)
```

```
('F9 FT B1022',)
('F9 FT B1026',)
('F9 FT B1021.2',)
('F9 FT B1031.2',)
```

# Total Number of Successful and Failure Mission Outcomes

---

List the total number of successful and failure mission outcomes

```
cursor = conn.cursor()
cursor.execute("SELECT Mission_Outcome, count(Mission_Outcome) FROM spacex_table GROUP BY Mission_Outcome" )
result = cursor.fetchall()

for k, v in result:
    print(k, v)
```

```
Failure (in flight) 1
Success 98
Success 1
Success (payload status unclear) 1
```

# Boosters Carried Maximum Payload

---

List the names of the booster\_versions which have carried the maximum payload mass. Use a subquery

```
cursor = conn.cursor()
cursor.execute("SELECT Booster_Version FROM spacex_table Where PAYLOAD_MASS_KG_ = (SELECT Max(PAYLOAD_MASS_KG_) FROM spacex_table)" )
result = cursor.fetchall()

for k in result:
    print(k)
```

```
('F9 B5 B1048.4',) •••
```

# 2015 Launch Records

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:** SQLite 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.

```
xmonth = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
monthdict={}

for i in range(len(xmonth)):
    if i+1 < 10:
        monthdict['0'+str(i+1)] = xmonth[i]
    else:
        monthdict[str(i+1)] = xmonth[i]

cursor = conn.cursor()
cursor.execute("SELECT substr(Date, 6,2), substr(Date,0,5), Booster_Version, Launch_Site, Landing_Outcome FROM spacex_table Where (Landing_Outcome = 'Failure (drone ship)')")
result = cursor.fetchall()

print('Month','Year','Booster_Version', 'Launch_Site', 'Landing_Outcome')
for i, j, k, l, m in result:
    print(monthdict[i],j,k,l,m)
```

```
Month Year Booster_Version Launch_Site Landing_Outcome
Jan 2015 F9 v1.1 B1012 CCAFS LC-40 Failure (drone ship)
Apr 2015 F9 v1.1 B1015 CCAFS LC-40 Failure (drone ship)
```

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

---

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
:
cursor = conn.cursor()
cursor.execute("SELECT Landing_Outcome, count(Landing_Outcome) as counter FROM spacex_table \
               where ('2010-06-04' <= Date) & (Date <='2017-03-20') \
               Group by Landing_Outcome" )
result = cursor.fetchall()

print('Landing_Outcome', 'Counter')
for i,j in result:
    print(i,j)
```

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

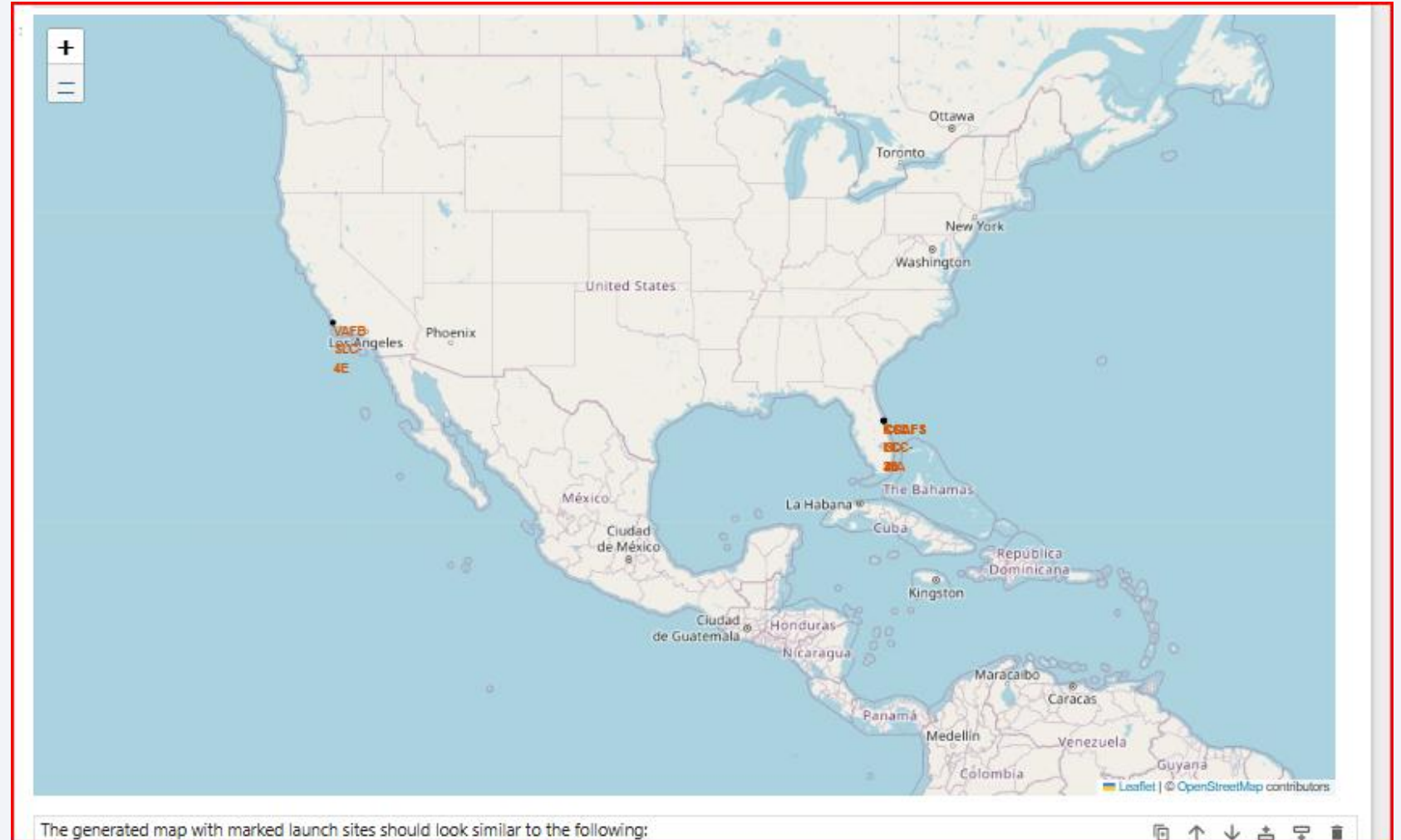
A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

Section 3

# Launch Sites Proximities Analysis

# Falcon 9 Rockets Launch Sites

- The four locations where Falcon 9 rockets are launched in California and Florida





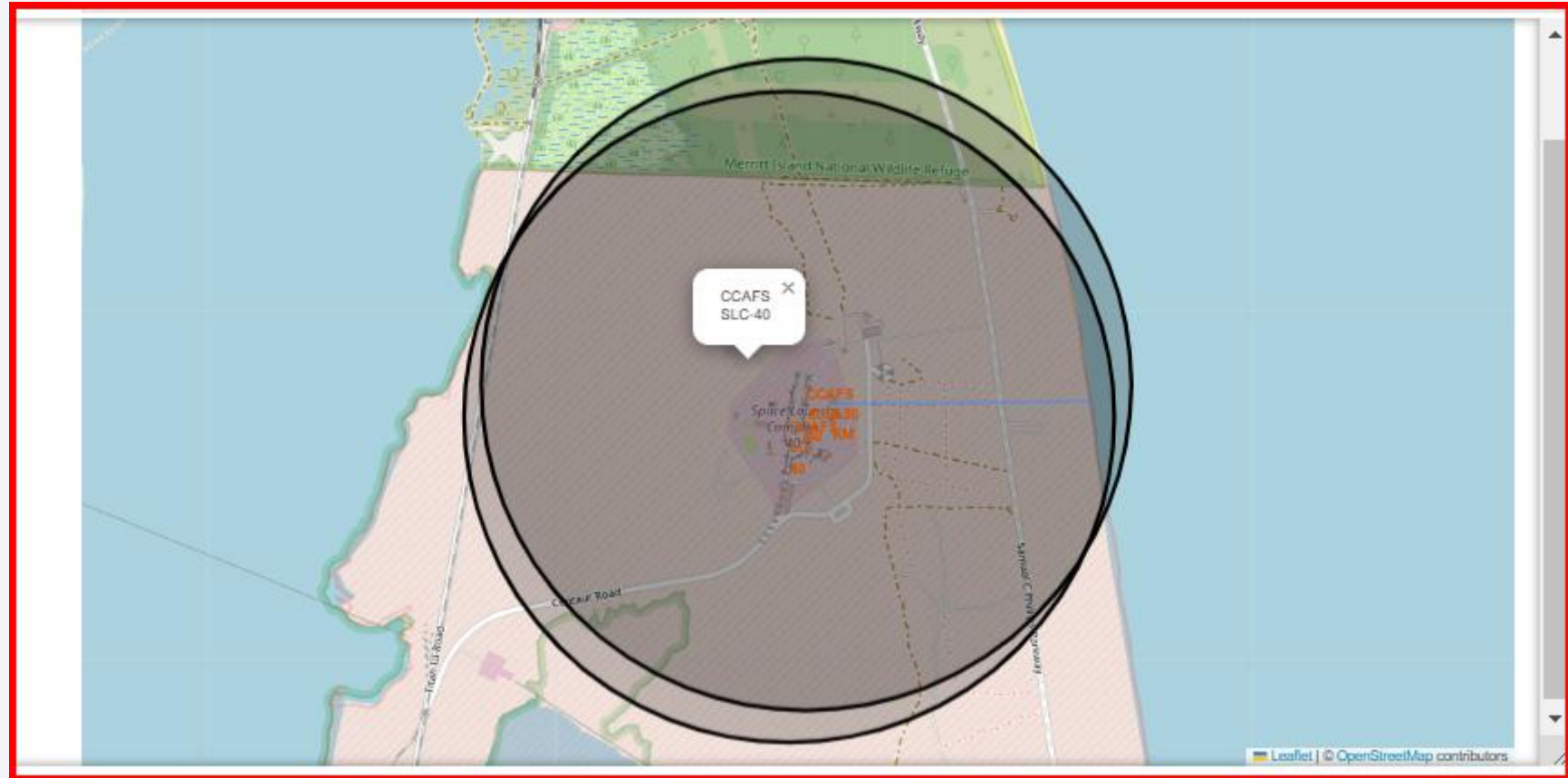
# Launch Sites Marked with Success/Failures





# Launch Site Closeness to Facilities—Beach, Rail and Cities

- Launch site proximities to railway, highway, coastline, with distance calculated and displayed
- Launch site is about 9 miles to the beach and there is no city nearby.



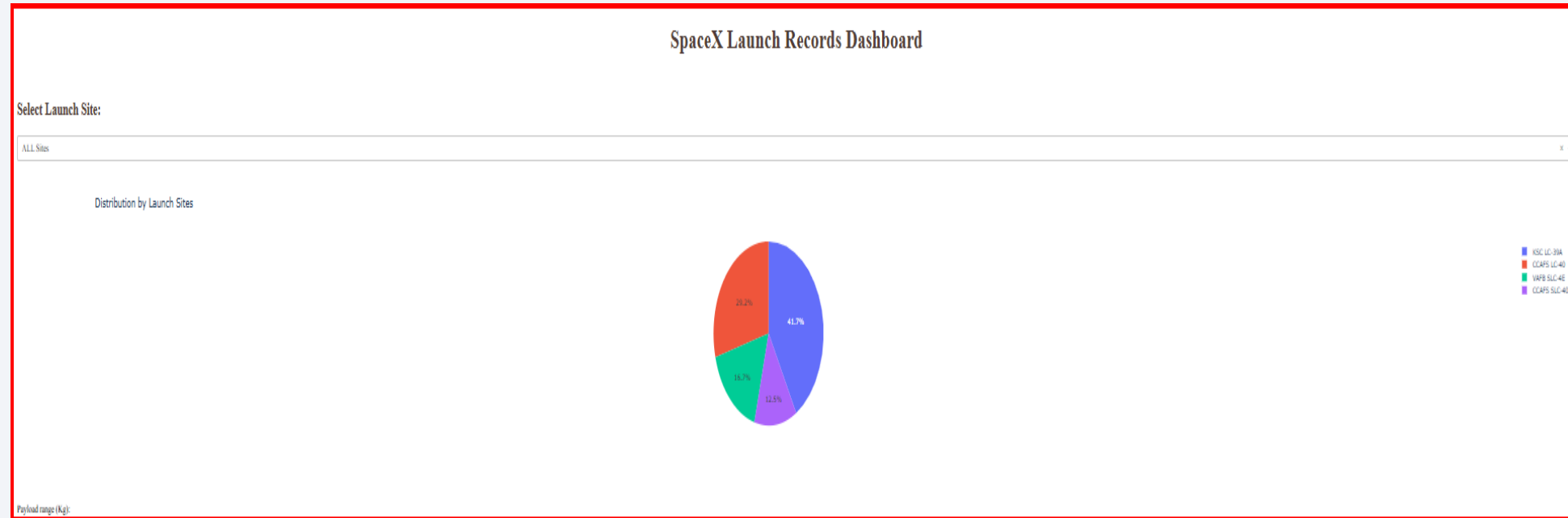


Section 4

# Build a Dashboard with Plotly Dash

# Launch Site Success Rate Distrinution

- Show the screenshot of launch success count for all sites, in a piechart
- KSC LC 39A is the best site for successful launch of all the available launch sites.



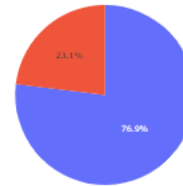
# KSC LC -39A Launch Site Success Rate

## SpaceX Launch Records Dashboard

Select Launch Site:

KSC LC-39A

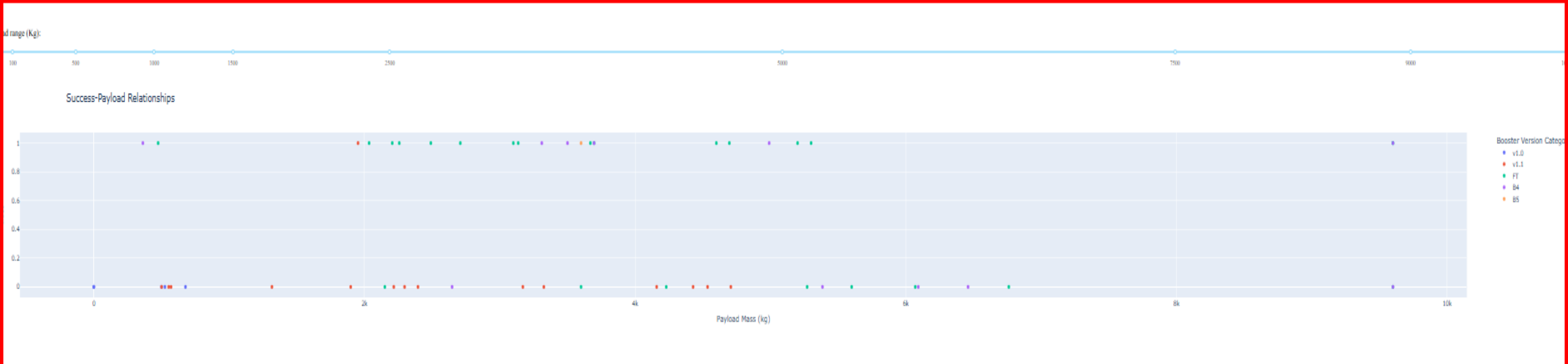
Total Success Launches for site KSC LC-39A



Payload range (Kg):

- Show the screenshot of the piechart for the launch site with highest launch success ratio
- The site have a successful launch rate of 76.9%

# Successful Luanch – Payload Relationship by Rocket Type



- The FT booster is the most successful given that most of the launches in reach orbit

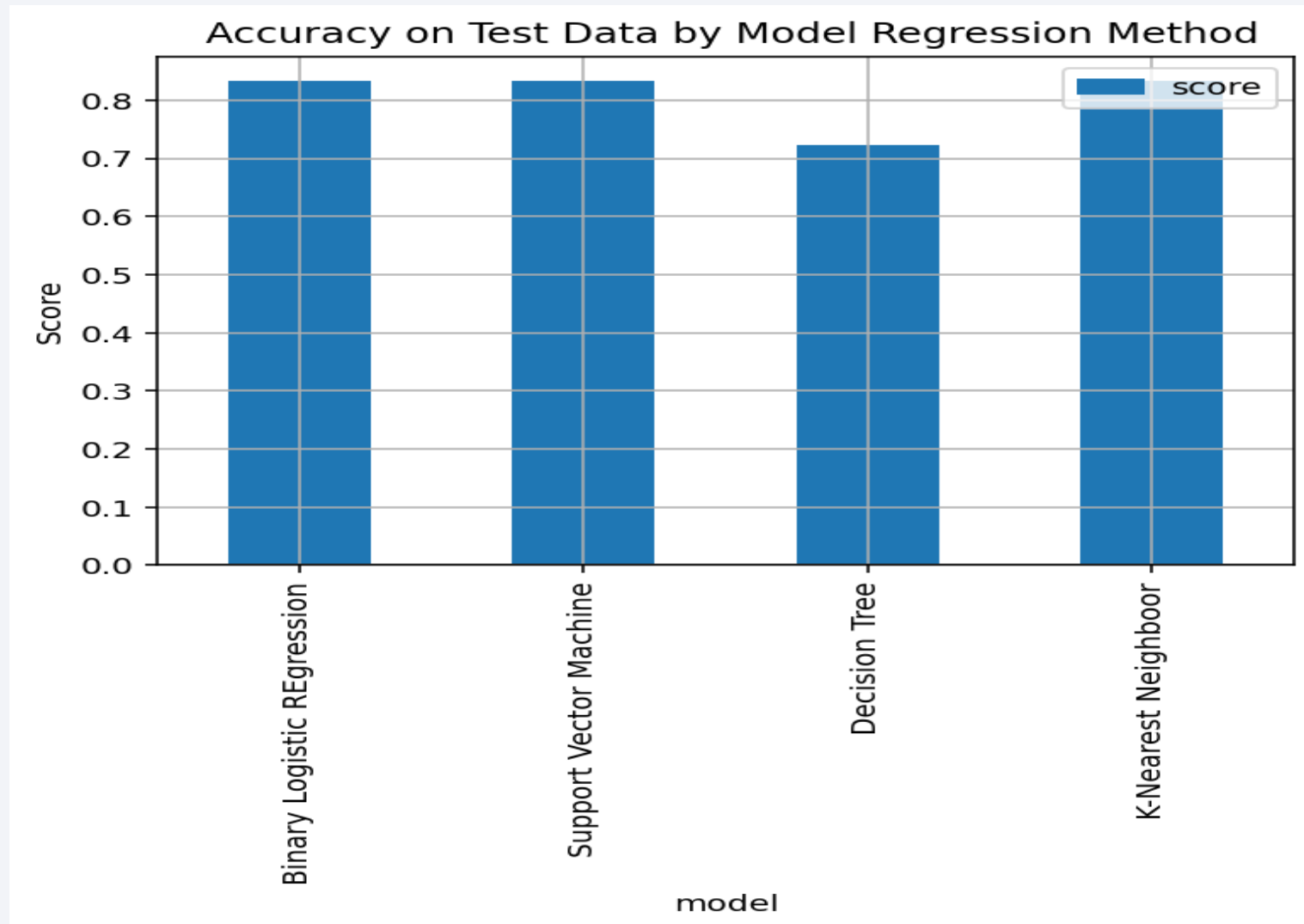


Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

- Binary Logistic, Support Vector Machines and KNN Regression have the best accuracy. They are the tallest and the score value is the same at 83.3%.



# Confusion Matrix

- The confusion matrix of logistic regression , SVM and KNN are similar. They also have the same score on the test sample.

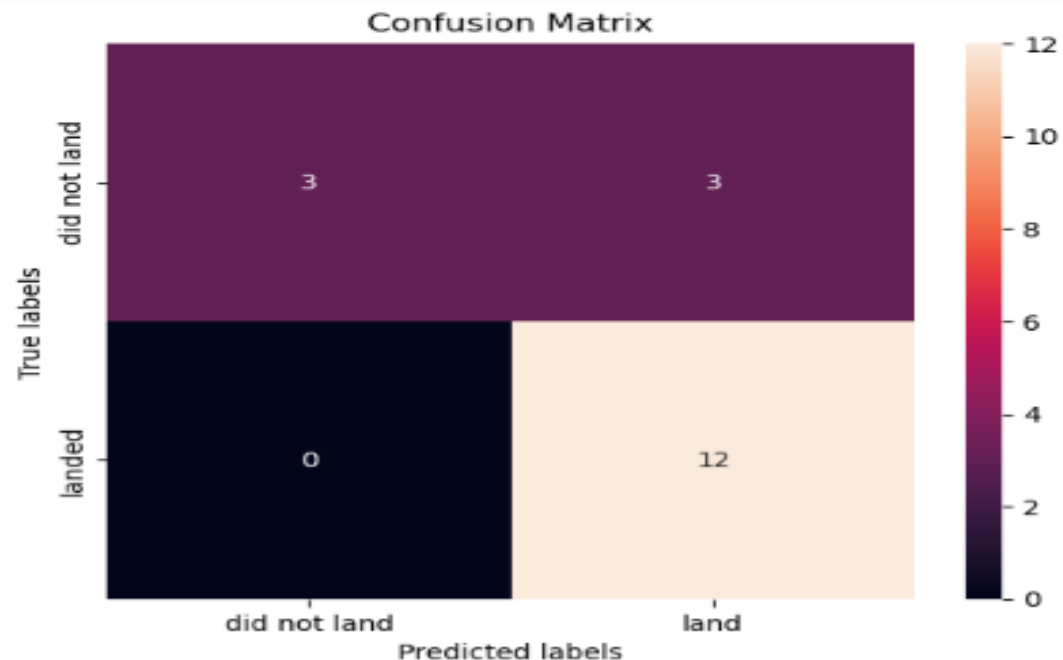
Calculate the accuracy on the test data using the method `score` :

```
16]: score = logreg_cv.score(X_test, Y_test)
      score
```

```
16]: 0.8333333333333334
```

Lets look at the confusion matrix:

```
17]: yhat=logreg_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```





# Conclusions

---

- The predictive accuracy as measured by the score, of the binary logistic regression, the support vector machine, and the K-Nearest Neighbor model on the test data are the same. Thus none is preferred over the other in this study.
- The confusion matrix of the three models are also the same which further reinforces the fact that the models performs similarly.
- The tree model though easy to interpret and explain is not the preferred choice here because its performance in the test sample is not as good as the other models.
- ...

# Appendix

---

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!

