IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Mampolokeng Marumo
14/11/2023

Outline

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

Executive Summary

Summary of methodologies

- Data Collection
- Data Wrangling
- Exploratory Data Analysis
- Interactive Visual Analytics
- Predictive Analysis

Summary of Results

- Exploratory Data Analysis (EDA) results
- Geospatial analytics
- Interactive dashboard
- Predictive analysis of classification models

Introduction

- Making space travel commercial
- The affordability of space travel
- The most successful company in space travel is SpaceX.
  - SpaceX is so much more affordable because it can reuse its first stage.
  - Sometimes the first stage will crash.

Objectives of the project:

- Determine the price of each launch
- Determine if SpaceX will reuse its first stage
- Train a machine learning model to predict if SpaceX will reuse its first stage

Section 1

Methodology

Methodology

1. Data collection methodology:
   - Get Requests to the SpaceX REST API
   - Web Scraping
2. Perform data wrangling:
   - Removing NaN values using .fillna()
   - Landing outcomes from 0 and 1 for unsuccessful and successful landings respectively
   - Using .value_counts to count several launches, mission outcomes and occurrences of different orbits.
3. Perform exploratory data analysis (EDA) using visualization and SQL:
   - SQL queries to manipulate data
   - Employed Pandas, and Matplotlib to visualize relationships
4. Perform interactive visual analytics using Folium and Plotly Dash
   - Folum for geospatial analytics
   - Plotly dash to create an interactive dashboard
5. Perform predictive analysis using classification models
   - Employed Scikit-Learn to train and classify models
   - Assess the accuracy of models

Data Collection – SPACEX REST API

1. GET request to the SpaceX REST API
2. Conversion to a .json file then to a Pandas DataFrame
3. Cleaning data
4. Calling functions to retrieve data and fill lists
5. Create a Pandas DataFrame from a constructed dictionary dataset
6. Filtered the DataFrame to only include Falcon 9 launches
7. Reset the FlightNumner column
8. Replaced missing values with mean values

https://github.com/marumom1/DS_Capstone/blob/main/jupyter-labs-spacex-data-collection-api%20(1).ipynb

Data Collection – SpaceX API

1. Create a GET response to the SpaceX REST API
2. Convert the response to a .json file then -> Pandas DataFrame
3. Clean data
4. Define lists for data to be stored in and call custom functions to retrieve data
5. Create a Pandas DataFrame from the dictionary dataset
6. Filter the DataFrame for only Falcon 9 launches
7. Reset the FlightNumber column
8. Replace missing values of PayloadMass with the mean PayloadMass value

- https://github.com/marumom1/DS_Capstone/blob/main/jupyter-labs-spacex-data-collection-api%20(1).ipynb

Data Collection – Web Scraping

1. Request the HTML from the URL
2. Assign a response to an object
3. BeautifulSoup from the HTML response object
4. Find tables within the HTML page
5. Collect all column header names from tables
6. Assigned column names as keys in a dictionary
7. Used functional and logic to parse tables to fill values in the dictionary
8. Converted dictionary to pandas DataFrame ready for export
9. https://github.com/marumom1/DS_Capstone/blob/main/jupyter-labs-webscraping.ipynb

Place your flowchart of web scraping here

Data Wrangling

To determine whether a booster will successfully land, it is best to have a binary column, i.e., where the value is 1 or 0, representing the success of the landing:

1. Defining a set of unsuccessful (bad) outcomes, `bad_outcome`
2. Create a list, `landing_class`, where the element is 0 if the corresponding row in `Outcome` is in the set `bad_outcome`, or else it's 1.
3. Create a `Class` column that contains the values from the list `landing_class`
4. Export the DataFrame as a .csv file

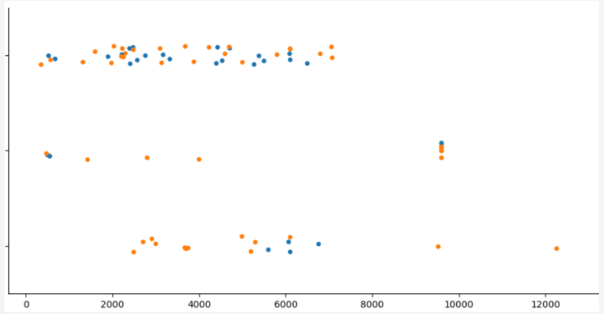
- https://github.com/marumom1/DS_Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

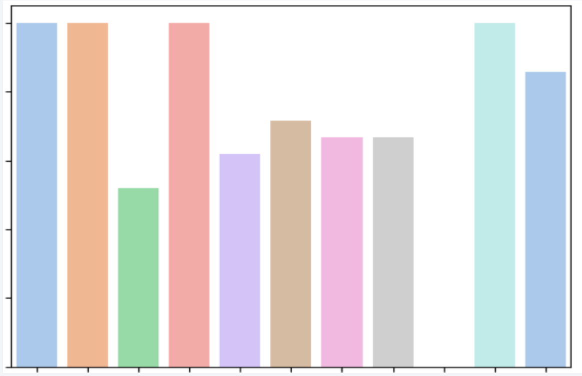
EDA with Data Visualization

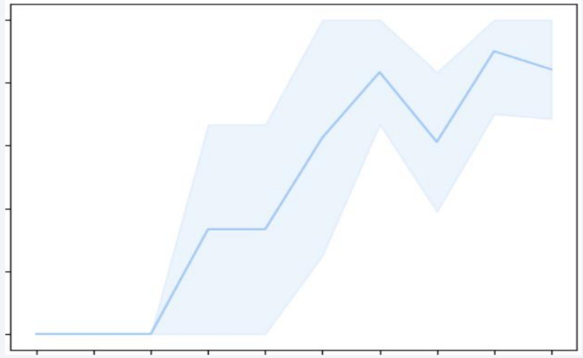
- Summarize what charts were plotted and why you used those charts

1. Scatter Charts –they are useful to observe relationships, or correlations, between two numeric variables
2. Bar Charts –they are used to compare a numerical value to a categorical variable.
3. Line Charts –contain numerical values on both axes and are generally used to show the change of a variable over time.

- https://github.com/marumom1/DS_Capstone/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb







EDA with SQL

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string ‘CCA’
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display the average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome on a ground pad was achieved
6. List the names of the boosters which had success on a drone ship and a payload mass between 4000 and 6000 kg
7. List the total number of successful and failed mission outcomes
8. List the names of the booster versions that have carried the maximum payload mass
9. List the failed landing outcomes on drone ships, their booster versions, and launch site names for 2015
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

- https://github.com/marumom1/DS_Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Folium.Circle –to add a highlighted circle area with a text label on a specific coordinate.
- Folium.Marker –to mark a specific location with an icon as a text label.
- Folium.PolyLine –to create lines between two coordinate points/locations and their distance

- https://github.com/marumom1/DS_Capstone/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Pie chart (px.pie())
- Scatter graphs (px.scatter())
- Bar chart (px.bar())

https://github.com/marumom1/DS_Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- To prepare the dataset for model development:
  - Load dataset
  - Data transformations (standardise and pre-process)
  - Split into training and test data sets: `train_test_split()`
  - Find the best type of machine learning algorithms
- For the algorithms:
  - `GridSearchCV` object and a dictionary of parameters
  - Fit the object to the parameters
  - Use the training data set to train the model

- For each chosen algorithm:
  - Using the output GridSearchCV object:
    - Check the tuned hyperparameters (`best_params_`)
    - Check the accuracy (`score` and `best_score_`)
  - Plot and examine the Confusion Matrix

- Review the accuracy scores for all chosen algorithms
- The model with the highest accuracy score is determined as the best-performing model

Results

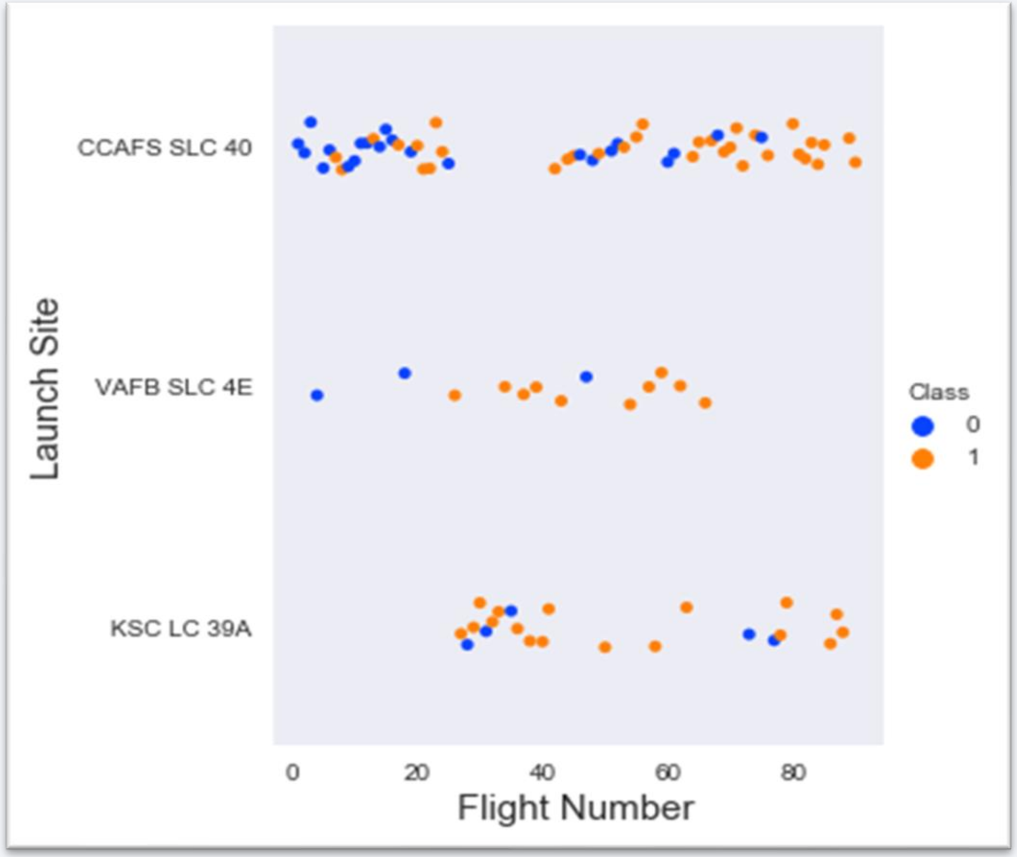
SUMMARY:

1. Exploratory data analysis results
2. Interactive analytics demo in screenshots
3. Predictive analysis results

Section 2

Insights drawn from EDA

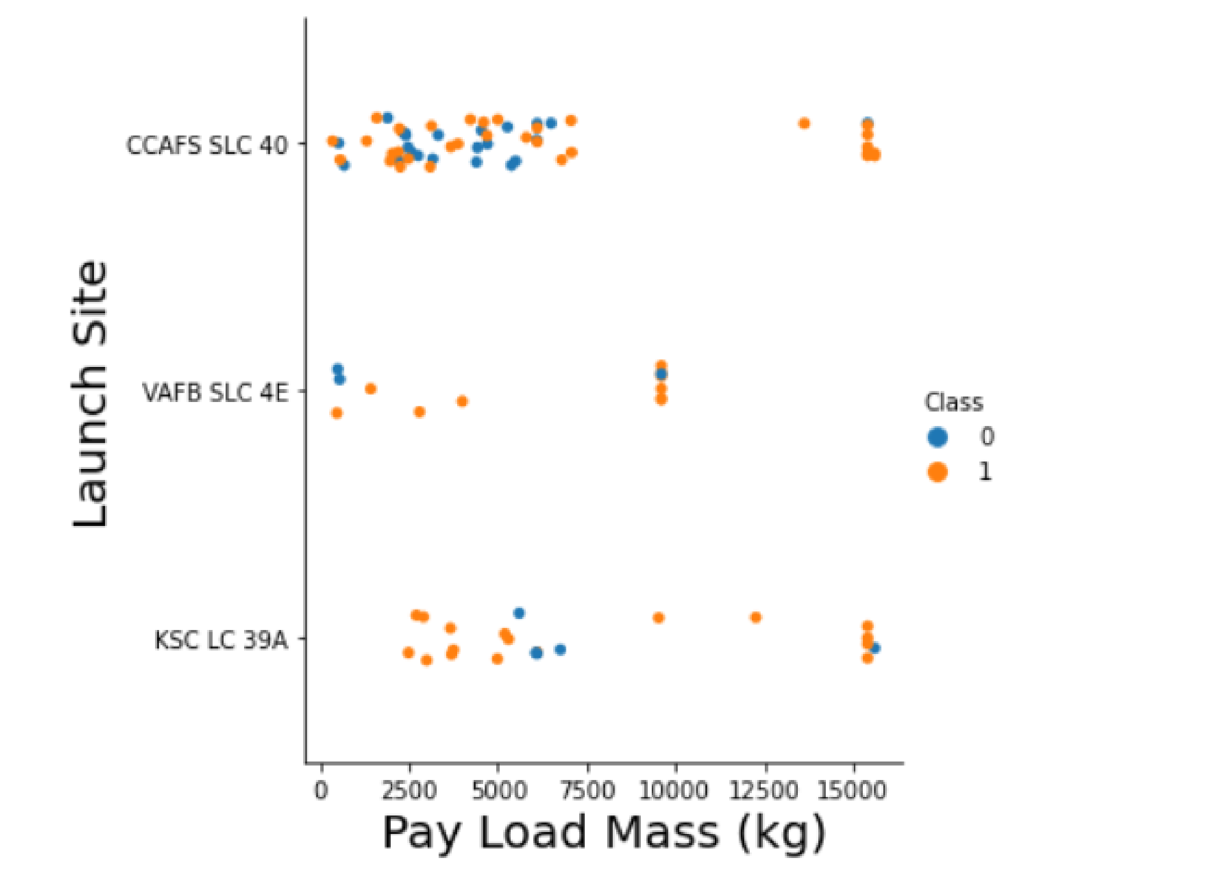
Flight Number vs. Launch Site



Flight Number vs. Launch Site

- As number of flights increases, the rate of success at a launch site increases.
- The early flights were launched from CCAFS SLC 40 are mostly unsuccessful.
- No flights launched from KSC LC 39A.

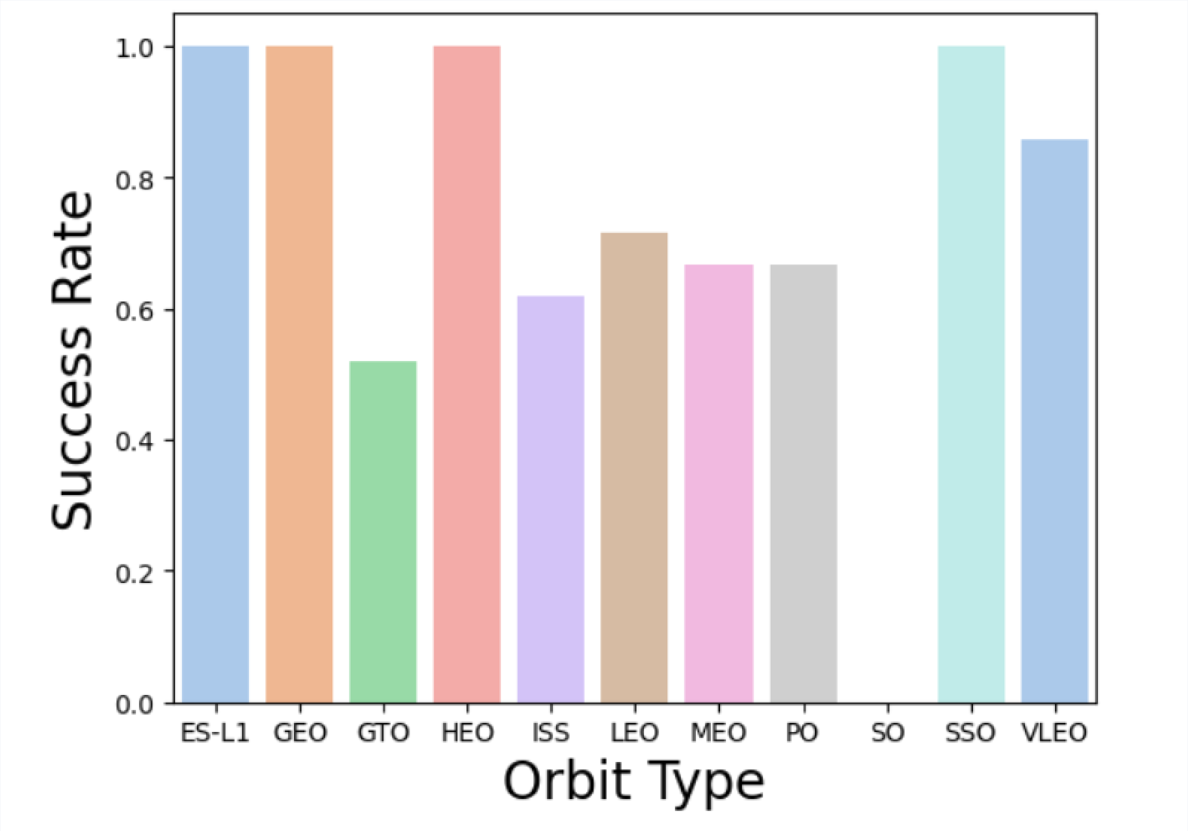
Payload vs. Launch Site



Payload vs. Launch Site

- No correlation between payload mass and success rate for a given launch site.
- Above 7000 kg, there are few unsuccessful landings.

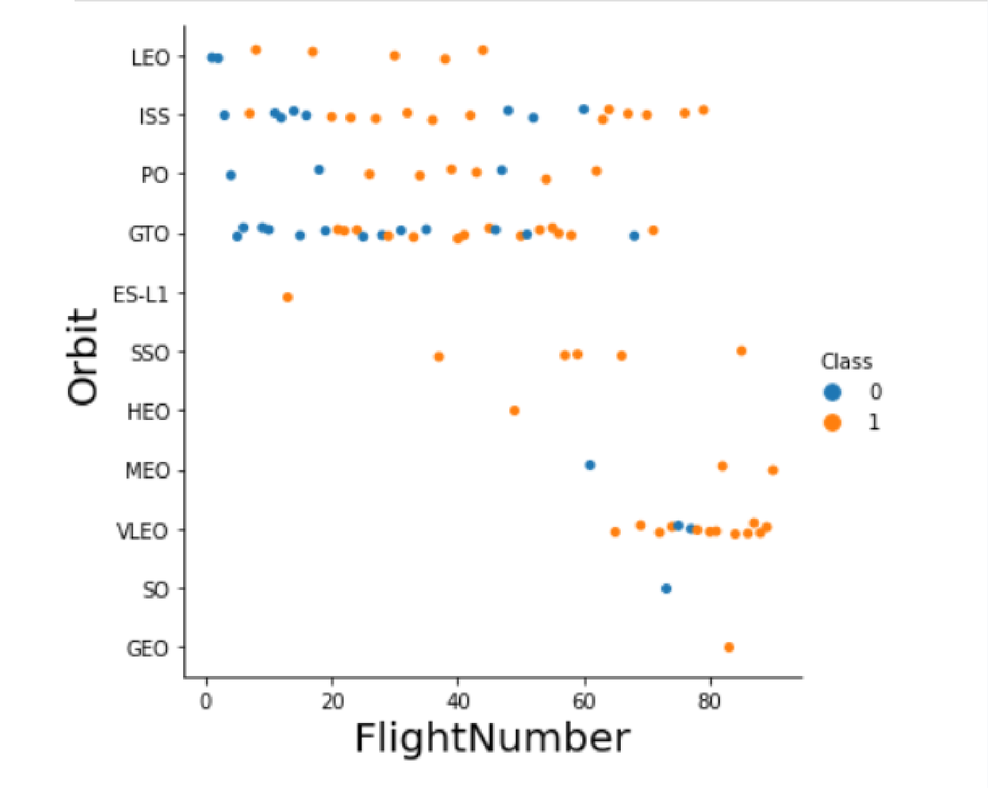
Success Rate vs. Orbit Type



Bar chart for the success rate of each orbit type

- Lowest success rate is SO orbit type
- 4 Orbit types with the highest success rates:
  - ES-L1
  - GEO
  - HEO
  - SSO

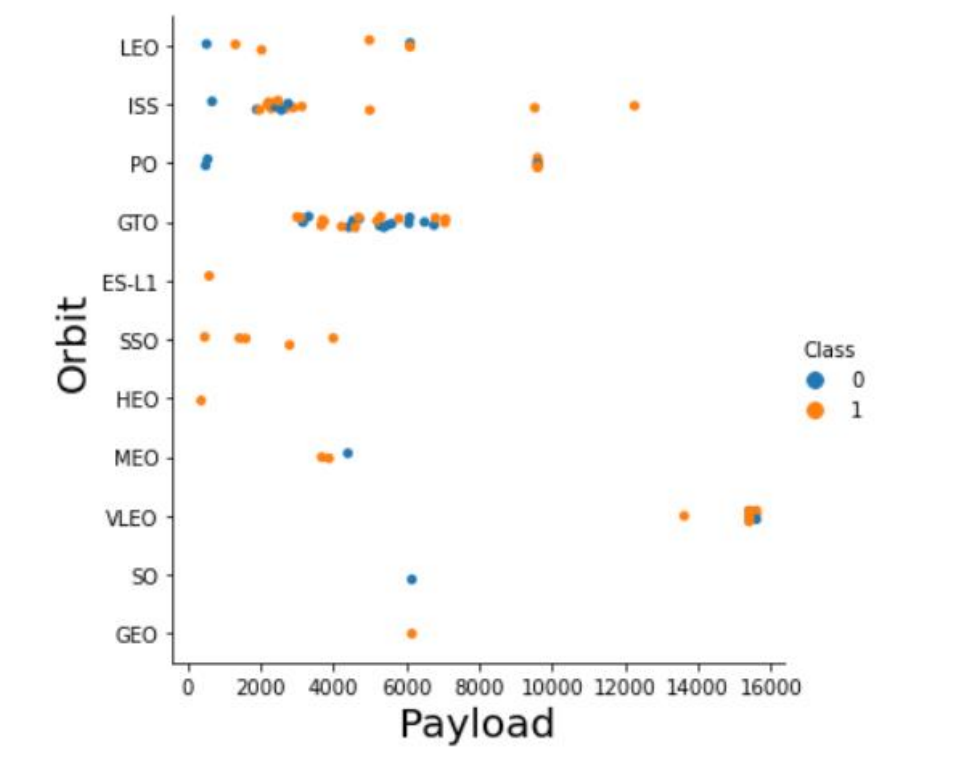
Flight Number vs. Orbit Type



Flight number vs. Orbit type

- There is a weak relationship between flight number and success rate

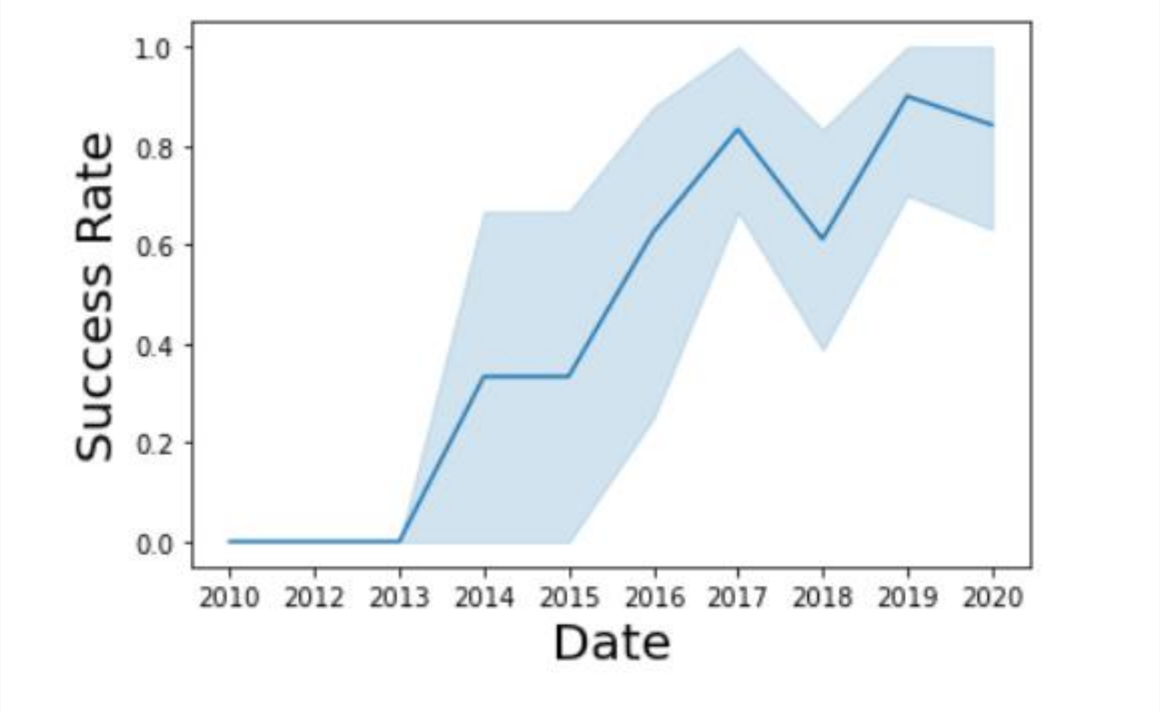
Payload vs. Orbit Type



Payload vs. orbit type

- Very low earth orbit launches are associated with heavier payloads.

Launch Success Yearly Trend



Line chart of yearly average success rate

- All landings unsuccessful between 2010 and 2013
- Increase in success rate after 2013
- A dip in success rate in 2018

All Launch Site Names

- ‘DISTINCT’ returns unique values from the launch site column in the table

```
[69]: %sql select Distinct "Launch_Site" from SPACEXTABLE;

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

[69]:

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

Launch Site Names Begin with 'CCA'

- ‘LIMIT 5’ returns only the first 5 records
- ‘LIKE’ is used with a wildcard to retrieve string values beginning with specified letters

```
[70]: %sql select * from SPACEXTABLE WHERE "Launch_Site" LIKE "CCA%" LIMIT 5;

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

[70]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- ‘SUM’ calculates the total
- Filters used to get results from specified string

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

```
[77]: %sql SELECT sum ("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE CUSTOMER = 'NASA (CRS)' ;

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

[77]:

sum ("PAYLOAD_MASS__KG_")
45596

Average Payload Mass by F9 v1.1

- ‘AVG’ calculates the average or mean
- ‘WHERE’ filters the results

```
[103]: %sql SELECT avg ("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE Booster_Version = "F9 V1.1" ;

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

[103]:

avg ("PAYLOAD_MASS__KG_")
None

First Successful Ground Landing Date

- 'MIN' finds the minimum of the specified column

```
[99]: %sql select min("Date") from SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)' ;

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

[99]:

min("Date")
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- 'BETWEEN' keyword allows for results in between a specified range

```
[104]: %sql SELECT DISTINCT Booster_Version from SPACEXTABLE WHERE Landing_Outcome ='Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 600
```

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

[104]:

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

Total Number of Successful and Failure Mission Outcomes

- ‘COUNT’ counts the number of specified occurance
- ‘GROUP BY’ groups results

```
[105]: %sql select substr(Mission_Outcome, 1, 7) as Mission_Outcome, count(*) from SPACEXTABLE GROUP BY 1 ;

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

[105]:

Mission_Outcome	count(*)
Failure	1
Success	100

Boosters Carried Maximum Payload

- Used ‘DISTINCT’ to retrieve only uniques booster versions

```
[107]: %sql select distinct Booster_Version from SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);

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

[107]:

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

2015 Launch Records

- ‘WHERE’ is used to filter the results for only failed landing outcomes
- ‘AND’ for only 2015

```
[108]: %sql select substr(Date, 6,2) as month, DATE, BOOSTER_VERSION, LAUNCH_SITE, [Landing_Outcome] from SPACEXTABLE where [Landing_Outcome] = 'Failu
```

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

[108]:

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

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

- ‘GROUP BY’ –groups the results
- ‘ORDER BY’ –orders the results
- ‘DECS’ –descending order

```
[109]: %sql SELECT [Landing_Outcome], count(*) as count_outcomes from SPACEXTABLE WHERE DATE between '04-06-2010' and '2017-03-20' GROUP BY [Landing_(
```

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

[109]:

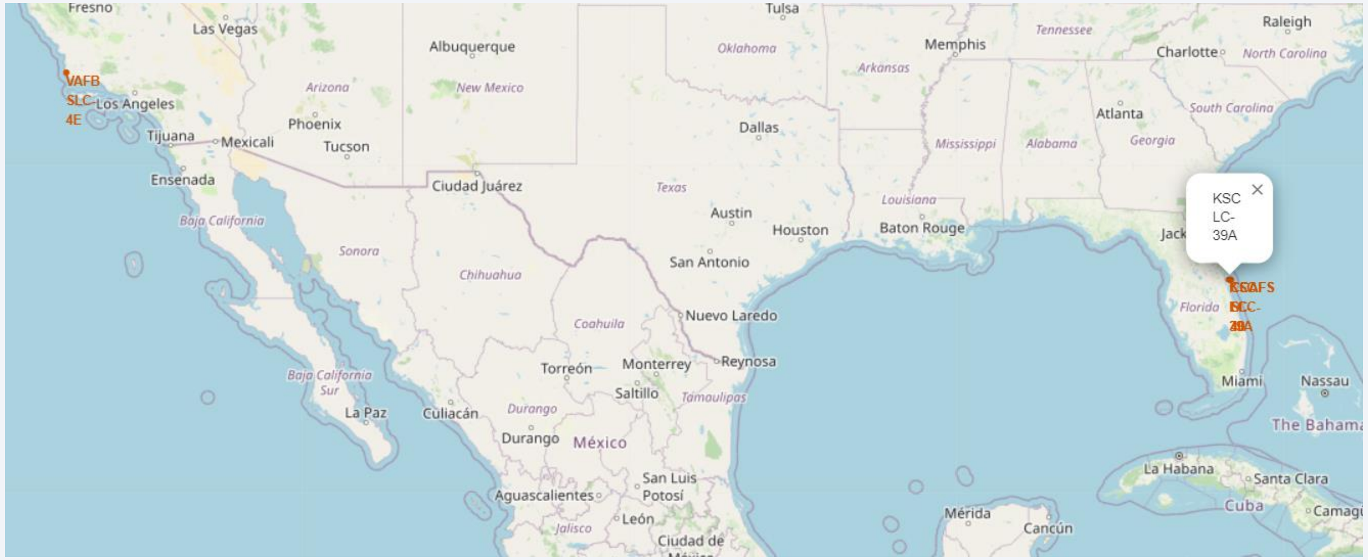
Landing_Outcome	count_outcomes
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

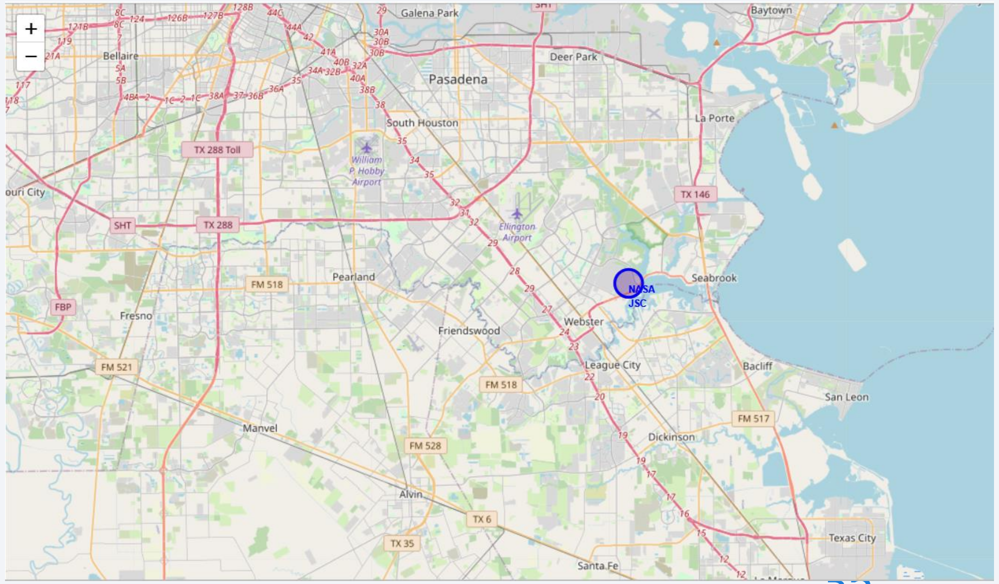
Section 3

Launch Sites Proximities Analysis

All launch sites

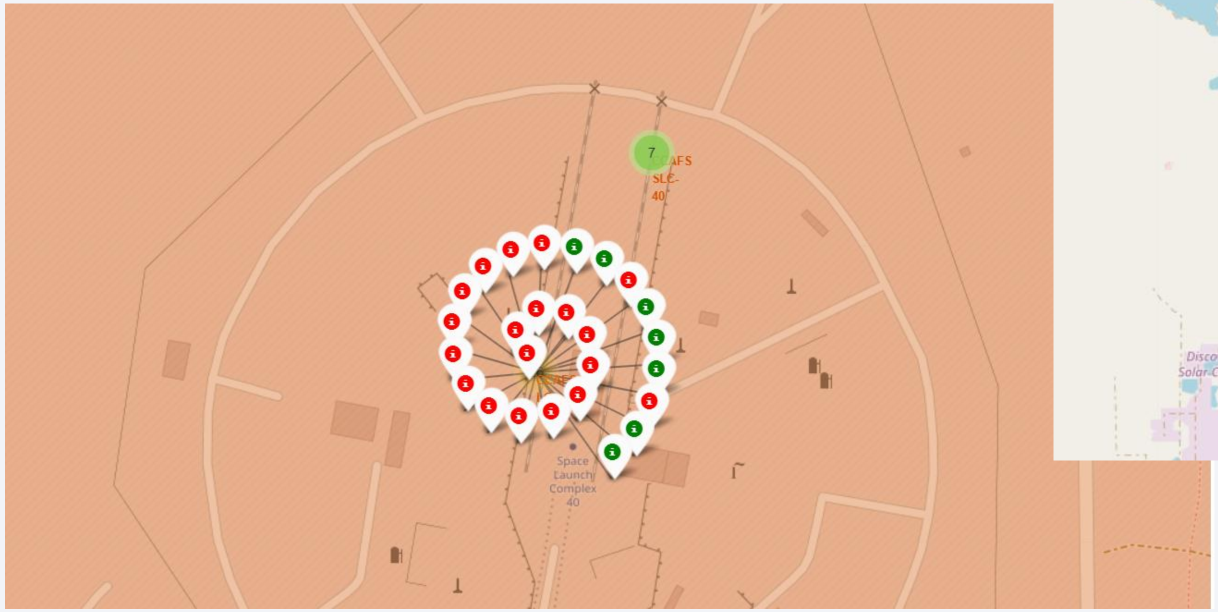
- All SpaceX launch sites in the United States in Florida and California

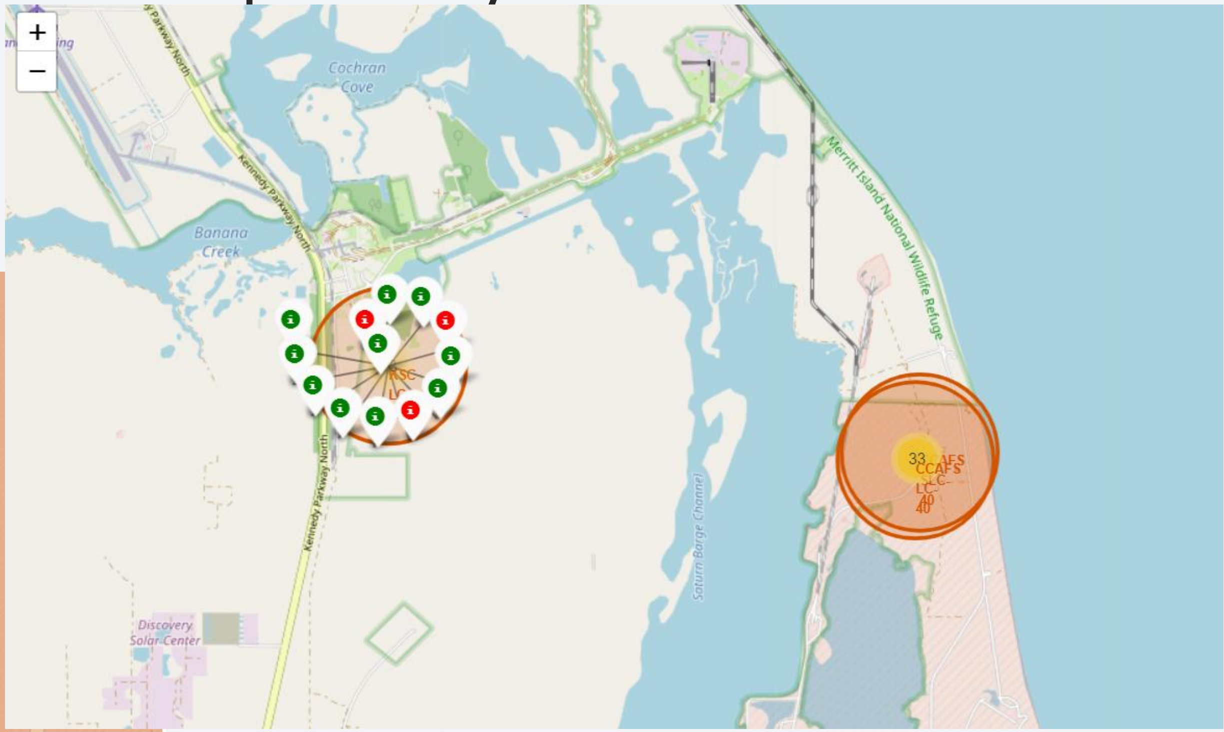




Success/Fail launches for each site

- Launches are grouped into clusters with green or red icons indicating successful or failed launces respectively.





Proximity of Launch sites to other points of intetest

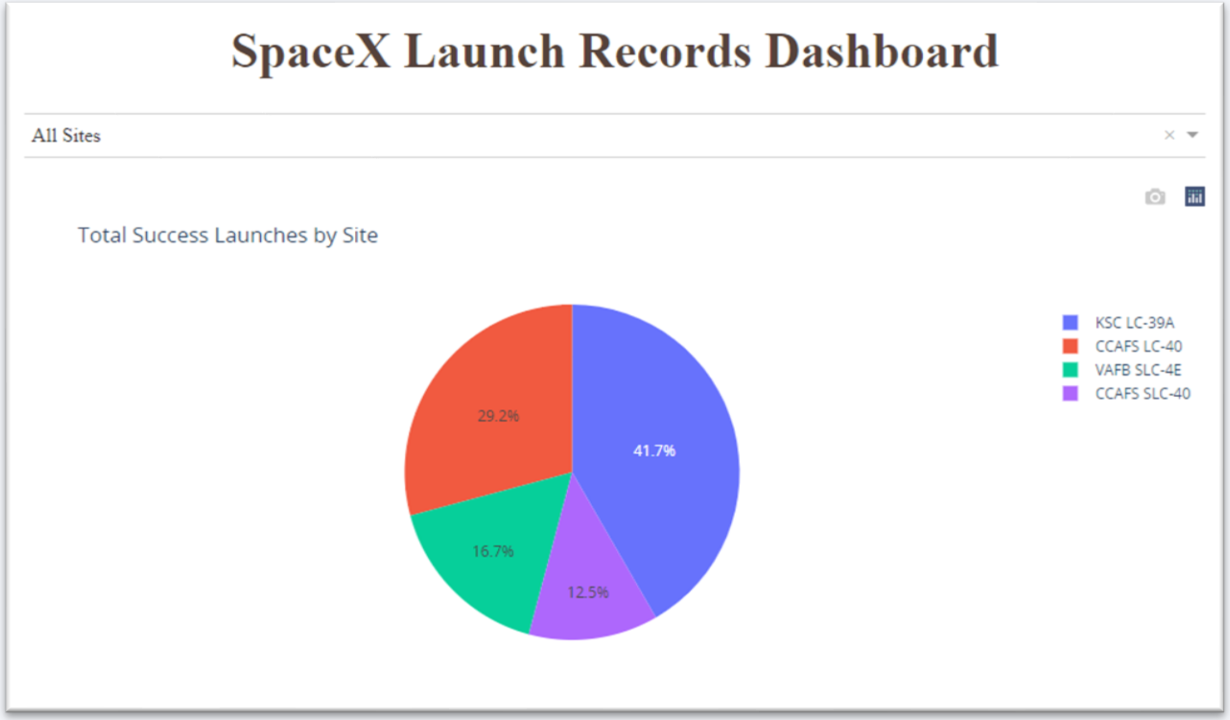
- Using Poly lines to determine the proximity of launch sites to the coastline.
- CCAFS SLC-40 = 0.86 km to coastline

Section 4

Build a Dashboard with Plotly Dash

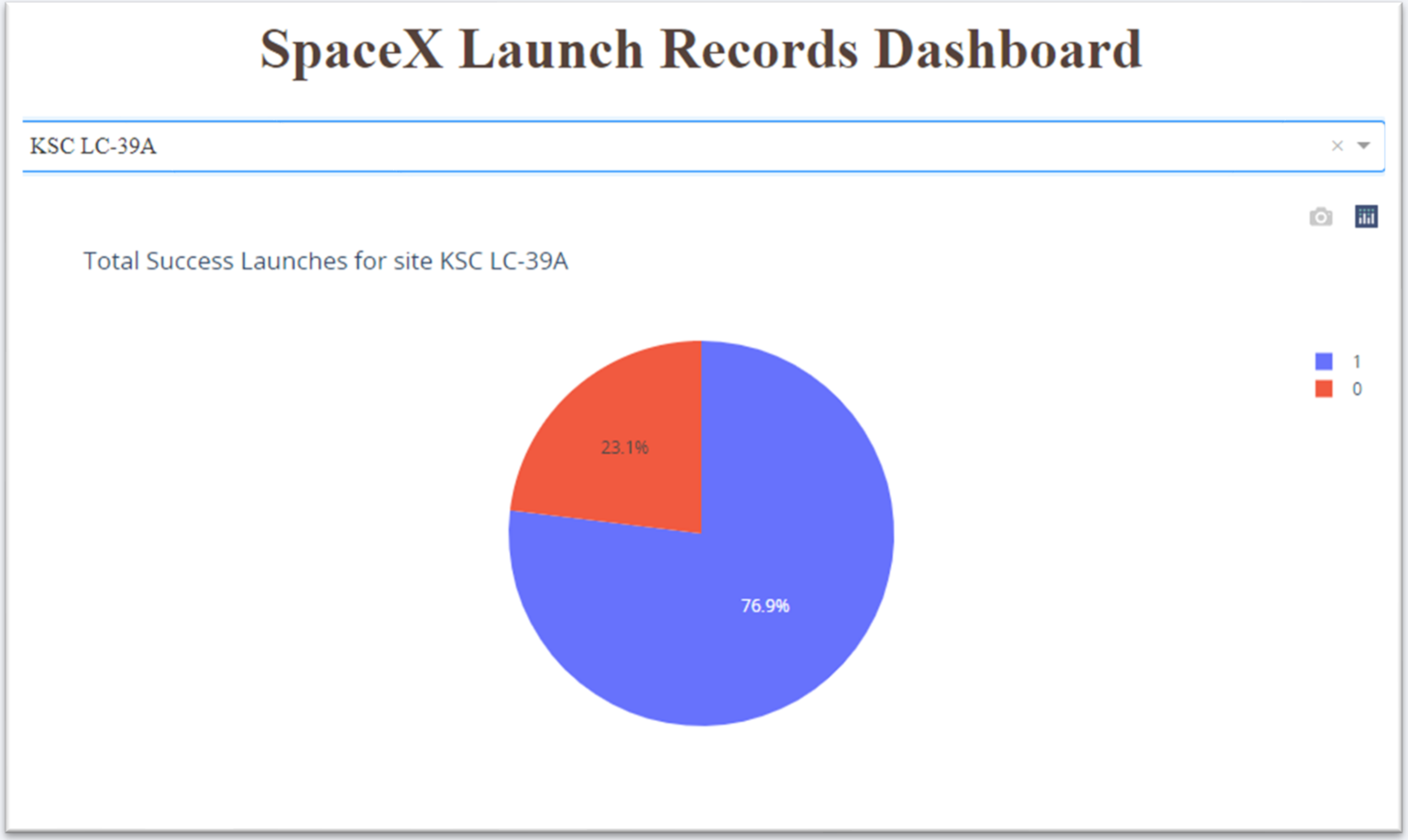
Launch success percent for all sites

- The launch site KSC LC-39 A has the most successful launches with 41.7%



Launch site with highest launch success ration

- KSC LC-39A had the highest rate of successful launches with 76.9%



Launch outcome vs. Payload scatter plot for all sites

- Success rate for massive pay loads is lower than that for low payloads.

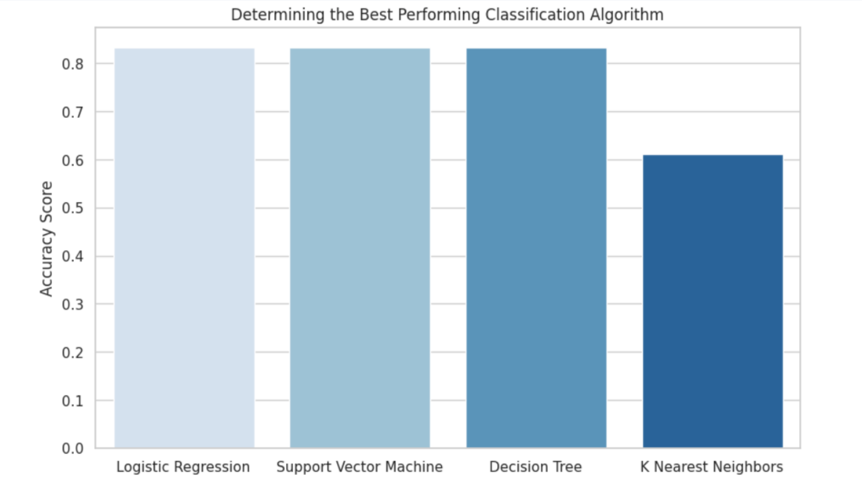
Section 5

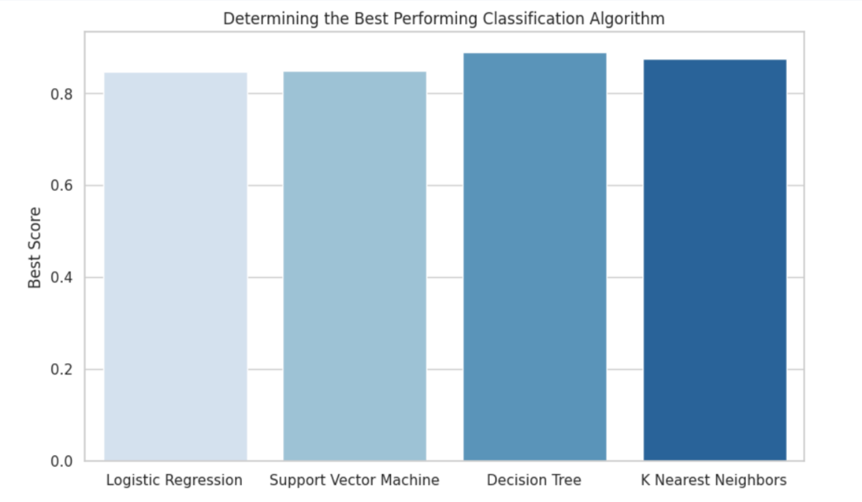
Predictive Analysis (Classification)

Classification Accuracy

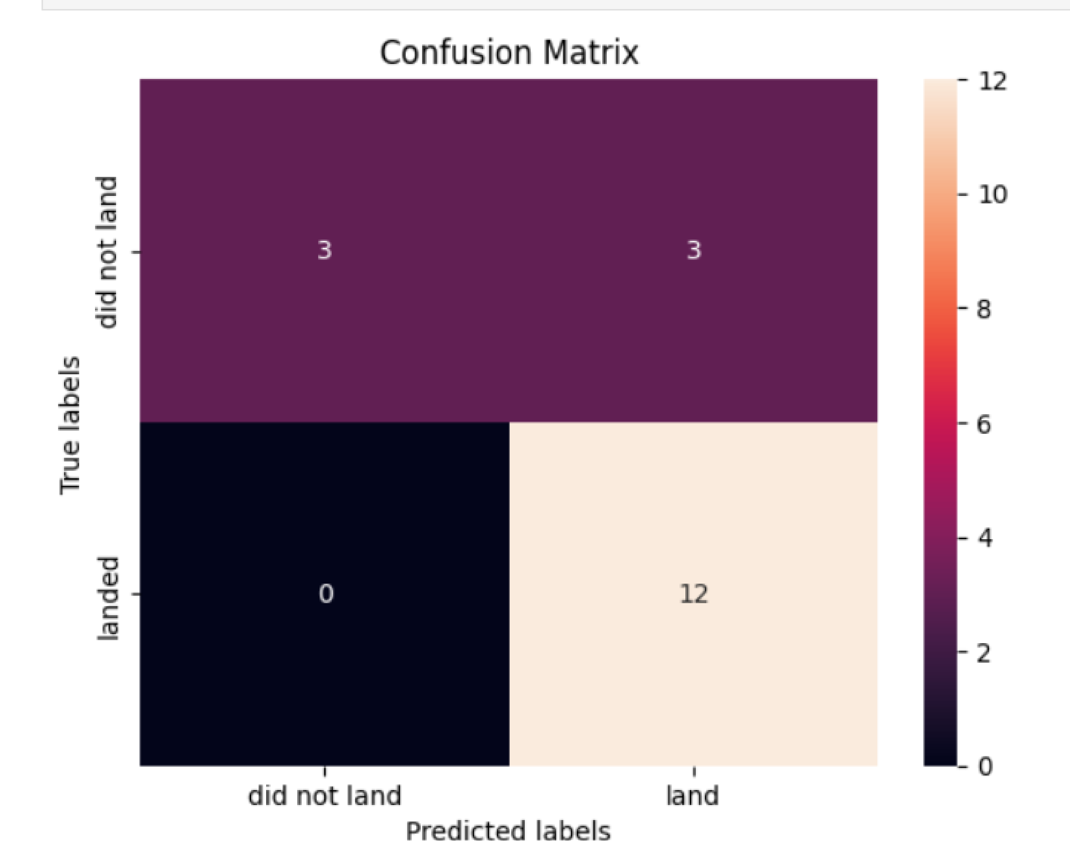
- 3 models have the same accuracy score of 0.83%
- The decision tree model has the best score of 0.89%

	Algorithm	Accuracy Score	Best Score
0	Logistic Regression	0.833333	0.846429
1	Support Vector Machine	0.833333	0.848214
2	Decision Tree	0.833333	0.889286
3	K Nearest Neighbors	0.611111	0.875000





Confusion Matrix



Conclusions

1. As the number of flights increases, the rate of success at a launch site increases, with most early flights being unsuccessful. I.e. with more experience, the success rate increases.

2. Orbit types ES-L1, GEO, HEO, and SSO, have the highest (100%) success rate.

3. The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches, and also the highest rate of successful launches, with a 76.9% success rate.

4. The success for massive payloads (over 4000kg) is lower than that for low payloads.

5. The best-performing classification model is the Decision Tree model, with an accuracy of 94.44%.

Appendix

- Useful Machine Learning libraries to import

```
import piplite
await piplite.install(['numpy'])
await piplite.install(['pandas'])
await piplite.install(['seaborn'])
```

We will import the following libraries for the lab

```
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
# NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collecti
import numpy as np
# Matplotlib is a plotting library for python and pyplot gives us a MatLab like plotting framework. We will use this in our plotter function to plot d
import matplotlib.pyplot as plt
#Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statist
import seaborn as sns
# Preprocessing allows us to standarsize our data
from sklearn import preprocessing
# Allows us to split our data into training and testing data
from sklearn.model_selection import train_test_split
# Allows us to test parameters of classification algorithms and find the best one
from sklearn.model_selection import GridSearchCV
# Logistic Regression classification algorithm
from sklearn.linear_model import LogisticRegression
# Support Vector Machine classification algorithm
from sklearn.svm import SVC
# Decision Tree classification algorithm
from sklearn.tree import DecisionTreeClassifier
# K Nearest Neighbors classification algorithm
from sklearn.neighbors import KNeighborsClassifier
```

Thank you!