IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

William Ubert
8/29/2024

Outline

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

Executive Summary / Methodologies Used

- The cost-effectiveness of SpaceX's Falcon 9 rocket is largely due to its ability to reuse the first stage, which significantly reduces the cost of subsequent launches. At a price of 62 million dollars per launch, Falcon 9 is a competitive alternative to other providers, who charge upward of 165 million dollars. Accurately predicting the success of the first stage landing is crucial, as it directly impacts the overall cost efficiency of the mission. In this project, we aim to develop a predictive model that determines whether the Falcon 9 first stage will successfully land. Our goal is to figure out the most cost effective factors of each launch
- The methodologies we used to complete this was calling API's, Web scraping and formatting, analysis using SQL and Pandas, building an interactive dashboard to get pie charts and other graphs, and using machine learning for predictive analysis

Data Collection

- For the data collection, we focused on the falcon 9 launches. To do this we used the python to collect the data using data API and web scraping, then we wrangled the API data.

Data Collection – SpaceX API

- Data:

FlightNumbe	Date	BoosterVersi	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCoun	Serial	Longitude	Latitude
1	6/4/10	Falcon 9	6123.547647	LEO	CCSFS SLC 4	None None	1	FALSE	FALSE	FALSE		1	0	B0003	-80.577366	28.5618571
2	5/22/12	Falcon 9	525	LEO	CCSFS SLC 4	None None	1	FALSE	FALSE	FALSE		1	0	B0005	-80.577366	28.5618571
3	3/1/13	Falcon 9	677	ISS	CCSFS SLC 4	None None	1	FALSE	FALSE	FALSE		1	0	B0007	-80.577366	28.5618571
4	9/29/13	Falcon 9	500	PO	VAFB SLC 4E	False Ocean	1	FALSE	FALSE	FALSE		1	0	B1003	-120.61083	34.632093
5	12/3/13	Falcon 9	3170	GTO	CCSFS SLC 4	None None	1	FALSE	FALSE	FALSE		1	0	B1004	-80.577366	28.5618571
6	1/6/14	Falcon 9	3325	GTO	CCSFS SLC 4	None None	1	FALSE	FALSE	FALSE		1	0	B1005	-80.577366	28.5618571
7	4/18/14	Falcon 9	2296	ISS	CCSFS SLC 4	True Ocean	1	FALSE	FALSE	TRUE		1	0	B1006	-80.577366	28.5618571
8	7/14/14	Falcon 9	1316	LEO	CCSFS SLC 4	True Ocean	1	FALSE	FALSE	TRUE		1	0	B1007	-80.577366	28.5618571
9	8/5/14	Falcon 9	4535	GTO	CCSFS SLC 4	None None	1	FALSE	FALSE	FALSE		1	0	B1008	-80.577366	28.5618571
10	9/7/14	Falcon 9	4428	GTO	CCSFS SLC 4	None None	1	FALSE	FALSE	FALSE		1	0	B1011	-80.577366	28.5618571
11	9/21/14	Falcon 9	2216	ISS	CCSFS SLC 4	False Ocean	1	FALSE	FALSE	FALSE		1	0	B1010	-80.577366	28.5618571
12	1/10/15	Falcon 9	2395	ISS	CCSFS SLC 4	False ASDS	1	TRUE	FALSE	TRUE	5e9e303238	1	0	B1012	-80.577366	28.5618571
13	2/11/15	Falcon 9	570	ES-L1	CCSFS SLC 4	True Ocean	1	TRUE	FALSE	TRUE		1	0	B1013	-80.577366	28.5618571
14	4/14/15	Falcon 9	1898	ISS	CCSFS SLC 4	False ASDS	1	TRUE	FALSE	TRUE	5e9e303238	1	0	B1015	-80.577366	28.5618571
15	4/27/15	Falcon 9	4707	GTO	CCSFS SLC 4	None None	1	FALSE	FALSE	FALSE		1	0	B1016	-80.577366	28.5618571
16	6/28/15	Falcon 9	2477	ISS	CCSFS SLC 4	None ASDS	1	TRUE	FALSE	TRUE	5e9e303238	1	0	B1018	-80.577366	28.5618571
17	12/22/15	Falcon 9	2034	LEO	CCSFS SLC 4	True RTLS	1	TRUE	FALSE	TRUE	5e9e303238	1	0	B1019	-80.577366	28.5618571
18	1/17/16	Falcon 9	553	PO	VAFB SLC 4E	False ASDS	1	TRUE	FALSE	TRUE	5e9e303338	1	0	B1017	-120.61083	34.632093
19	3/4/16	Falcon 9	5271	GTO	CCSFS SLC 4	False ASDS	1	TRUE	FALSE	TRUE	5e9e303238	1	0	B1020	-80.577366	28.5618571
20	4/8/16	Falcon 9	3136	ISS	CCSFS SLC 4	True ASDS	1	TRUE	FALSE	TRUE	5e9e303238	2	1	B1021	-80.577366	28.5618571
21	5/6/16	Falcon 9	4696	GTO	CCSFS SLC 4	True ASDS	1	TRUE	FALSE	TRUE	5e9e303238	2	0	B1022	-80.577366	28.5618571
22	5/27/16	Falcon 9	3100	GTO	CCSFS SLC 4	True ASDS	1	TRUE	FALSE	TRUE	5e9e303238	2	1	B1023	-80.577366	28.5618571
23	7/18/16	Falcon 9	2257	ISS	CCSFS SLC 4	True RTLS	1	TRUE	FALSE	TRUE	5e9e303238	2	1	B1025	-80.577366	28.5618571
24	8/14/16	Falcon 9	4600	GTO	CCSFS SLC 4	True ASDS	1	TRUE	FALSE	TRUE	5e9e303238	2	0	B1026	-80.577366	28.5618571
25	9/1/16	Falcon 9	5500	GTO	CCSFS SLC 4	None ASDS	1	TRUE	FALSE	TRUE	5e9e303238	3	0	B1028	-80.577366	28.5618571
26	1/14/17	Falcon 9	9600	PO	VAFB SLC 4E	True ASDS	1	TRUE	FALSE	TRUE	5e9e303338	3	1	B1029	-120.61083	34.632093
27	2/19/17	Falcon 9	2490	ISS	KSC LC 39A	True RTLS	1	TRUE	FALSE	TRUE	5e9e303238	3	1	B1031	-80.603956	28.6080585
28	3/16/17	Falcon 9	5600	GTO	KSC LC 39A	None None	1	FALSE	FALSE	FALSE		3	0	B1030	-80.603956	28.6080585
29	3/30/17	Falcon 9	5300	GTO	KSC LC 39A	True ASDS	2	TRUE	TRUE	TRUE	5e9e303238	2	1	B1021	-80.603956	28.6080585
30	5/1/17	Falcon 9	6123.547647	LEO	KSC LC 39A	True RTLS	1	TRUE	FALSE	TRUE	5e9e303238	3	1	B1032	-80.603956	28.6080585
31	5/15/17	Falcon 9	6070	GTO	KSC LC 39A	None None	1	FALSE	FALSE	FALSE		3	0	B1034	-80.603956	28.6080585
32	6/3/17	Falcon 9	2708	ISS	KSC LC 39A	True RTLS	1	TRUE	FALSE	TRUE	5e9e303238	3	1	B1035	-80.603956	28.6080585

- Github: [CapstoneIBM/jupyter-labs-spacex-data-collection-api.ipynb at main · willubert/CapstoneIBM (github.com)]

- Github CSV:[CapstoneIBM/dataset_part_1.csv at main · willubert/CapstoneIBM (github.com)]

FLOWCHART

Define API Endpoint: Identify the relevant SpaceX REST API endpoints.

Send GET Request: Perform GET requests to fetch launch data.

Parse JSON Response: Extract relevant fields (e.g., flight_number, launch_date, rocket, launchpad, payload, success).

Data Transformation: Convert extracted data into a structured format, such as a DataFrame.

Data Storage: Store the data in a database or CSV file for analysis.

Data Cleaning: Handle missing values, standardize column names, and filter data based on specific criteria.

.

Data Collection - Scraping

- Data:

Flight No.	Date	Time	Version Boos	Launch Site	Payload	Payload mas:	Orbit	Customer	Launch outco	Booster landing
1	4-Jun-10	18:45	F9 v1.07B000	CCAFS	Dragon Spac	0	LEO	SpaceX		Failure
2	8-Dec-10	15:43	F9 v1.07B000	CCAFS	Dragon	0	LEO	NASA	Success	Failure
3	22-May-12	7:44	F9 v1.07B000	CCAFS	Dragon	525 kg	LEO	NASA	Success	
4	8-Oct-12	0:35	F9 v1.07B000	CCAFS	SpaceX CRS-	4,700 kg	LEO	NASA		No attempt
5	1-Mar-13	15:10	F9 v1.07B000	CCAFS	SpaceX CRS-	4,877 kg	LEO	NASA		
6	29-Sep-13	16:00	F9 v1.17B100	VAFB	CASSIOPE	500 kg	Polar orbit	MDA	Success	Uncontrolled
7	3-Dec-13	22:41	F9 v1.1	CCAFS	SES-8	3,170 kg	GTO	SES	Success	No attempt
8	6-Jan-14	22:06	F9 v1.1	CCAFS	Thaicom 6	3,325 kg	GTO	Thaicom	Success	No attempt
9	18-Apr-14	19:25	F9 v1.1	Cape Canave	SpaceX CRS-	2,296 kg	LEO	NASA		Controlled
10	14-Jul-14	15:15	F9 v1.1	Cape Canave	Orbcomm-O(	1,316 kg	LEO	Orbcomm	Success	Controlled
11	5-Aug-14	8:00	F9 v1.1	Cape Canave	AsiaSat 8	4,535 kg	GTO	AsiaSat	Success	No attempt
12	7-Sep-14	5:00	F9 v1.1[	Cape Canave	AsiaSat 6	4,428 kg	GTO	AsiaSat	Success	
13	21-Sep-14	5:52	F9 v1.1[	Cape Canave	SpaceX CRS-	2,216 kg	LEO	NASA	Success	Uncontrolled
14	10-Jan-15	9:47	F9 v1.1[	Cape Canave	SpaceX CRS-	2,395 kg	LEO	NASA	Success	Failure
15	11-Feb-15	23:03	F9 v1.1[	Cape Canave	DSCOVR	570 kg	HEO	USAF		Controlled
16	2-Mar-15	3:50	F9 v1.1[	Cape Canave	ABS-3A	4,159 kg	GTO	ABS		No attempt
17	14-Apr-15	20:10	F9 v1.1[	Cape Canave	SpaceX CRS-	1,898 kg	LEO	NASA		Failure
18	27-Apr-15	23:03	F9 v1.1[	Cape Canave	T√ºrkmen√Ñ	4,707 kg	GTO			No attempt
19	28-Jun-15	14:21	F9 v1.1[	Cape Canave	SpaceX CRS-	1,952 kg	LEO	NASA	Failure	Precluded
20	22-Dec-15	1:29	F9 FT[	Cape Canave	Orbcomm-O(	2,034 kg	LEO	Orbcomm		Success
21	17-Jan-16	18:42	F9 v1.1[	VAFB	Jason-3	553 kg	LEO	NASA		Failure
22	4-Mar-16	23:35	F9 FT[	Cape Canave	SES-9	5,271 kg	GTO	SES		Failure
23	8-Apr-16	20:43	F9 FT[	Cape Canave	SpaceX CRS-	3,136 kg	LEO	NASA	Success	Success
24	6-May-16	5:21	F9 FT[	Cape Canave	JCSAT-14	4,696 kg	GTO	SKY Perfect J:		Success
25	27-May-16	21:39	F9 FT[	Cape Canave	Thaicom 8	3,100 kg	GTO	Thaicom		Success
26	15-Jun-16	14:29	F9 FT[	Cape Canave	ABS-2A	3,600 kg	GTO	ABS		Failure
27	18-Jul-16	4:45	F9 FT[	Cape Canave	SpaceX CRS-	2,257 kg	LEO	NASA		Success
28	14-Aug-16	5:26	F9 FT[	Cape Canave	JCSAT-16	4,600 kg	GTO	SKY Perfect J:		Success
29	14-Jan-17	17:54	F9 FT[	VAFB	Iridium NEXT	9,600 kg	Polar	Iridium Comr		Success

- Github: [CapstoneIBM/jupyter-labs-webscraping.ipynb at main · willubert/CapstoneIBM (github.com)]

- Github CSV: [CapstoneIBM/spacex_web_scraped.csv at main · willubert/CapstoneIBM (github.com)]

1. **Identify Target Website**: Determine the website or specific pages from which data needs to be scraped.
2. **Inspect HTML Structure**: Analyze the HTML structure to identify tags and classes containing the desired data.
3. **Send HTTP Request**: Use libraries like requests to send HTTP GET requests to the target URLs.
4. **Parse HTML Content**: Use a parser like BeautifulSoup to navigate the HTML structure and extract relevant data.
5. **Data Extraction**: Extract specific data fields the parsed HTML.
6. **Data Cleaning**: Clean and format the extracted data to ensure consistency and usability.
7. **Data Storage**: Save the cleaned data in a structured format, such as a CSV file or database.

Data Wrangling

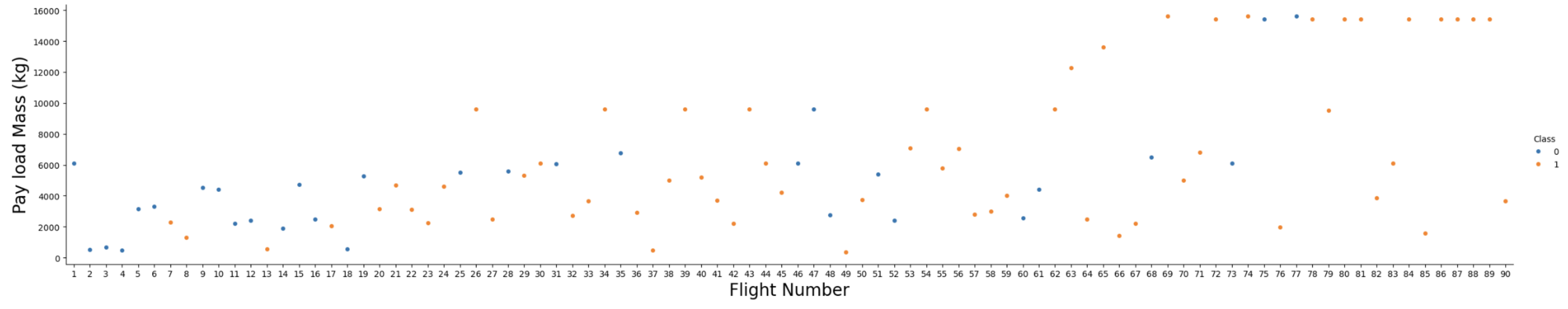
- **Data Collection:**
- Import raw data from CSV files or APIs.
- Load the dataset using Pandas.
- **Data Cleaning:**
- Handle missing values (e.g., drop or impute missing data).
- Remove duplicates.
- Standardize data formats (e.g., dates, strings).
- **Data Transformation:**
- Convert data types as necessary (e.g., strings to datetime).
- Create new columns if needed (e.g., extract features from timestamps).
- Normalize or scale numerical data

- GitHub: CapstoneIBM/labs-jupyter-spacex-Data wrangling (1).ipynb at main · willubert/CapstoneIBM (github.com)

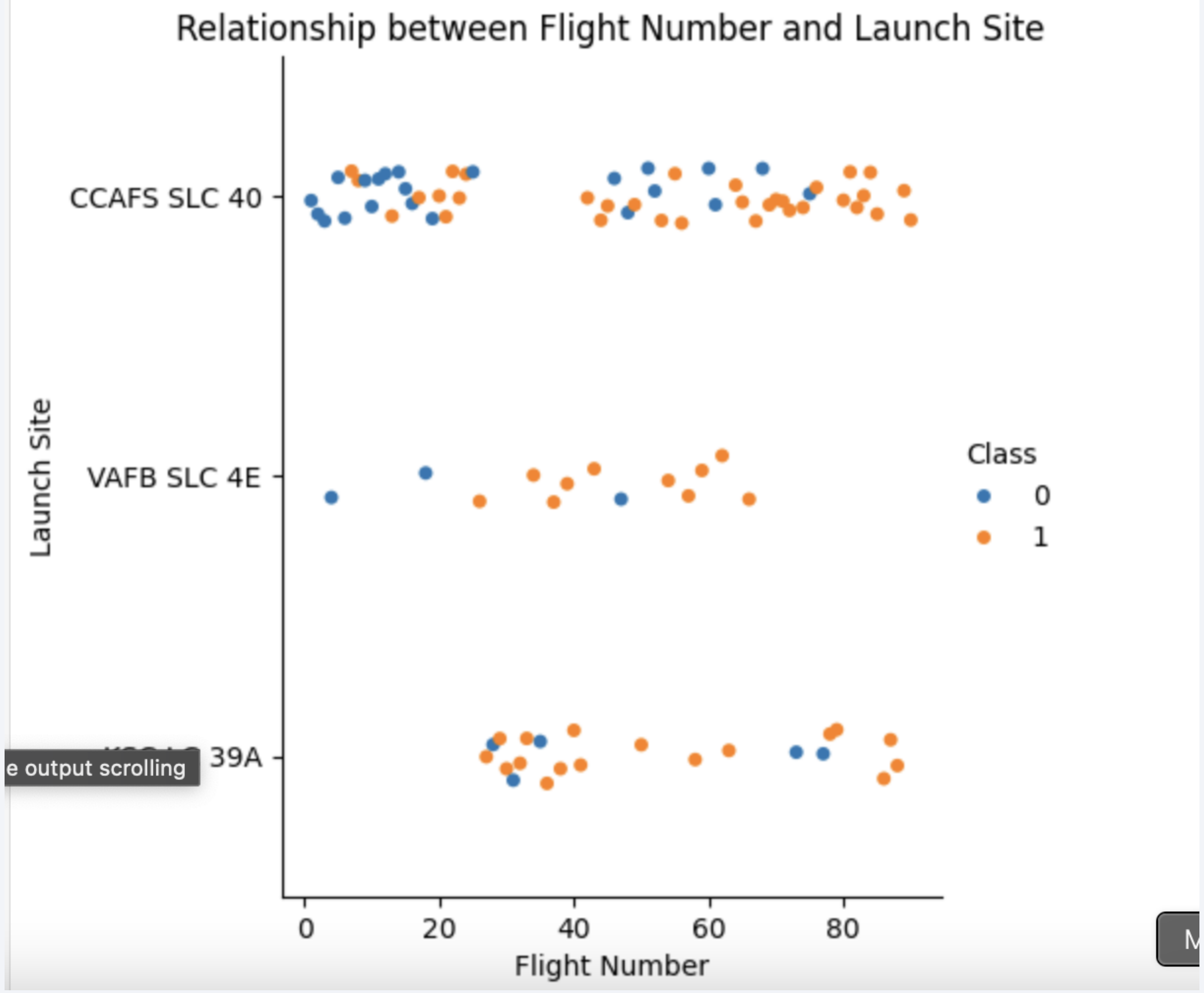
EDA with Data Visualization

- We were able to visualize a lot of the data using mathplotlib and seaborn. The next couple of slides will show the various charts and graphs used for this analysis
- Github: CapstoneIBM/edadataviz.ipynb at main · willubert/CapstoneIBM (github.com)

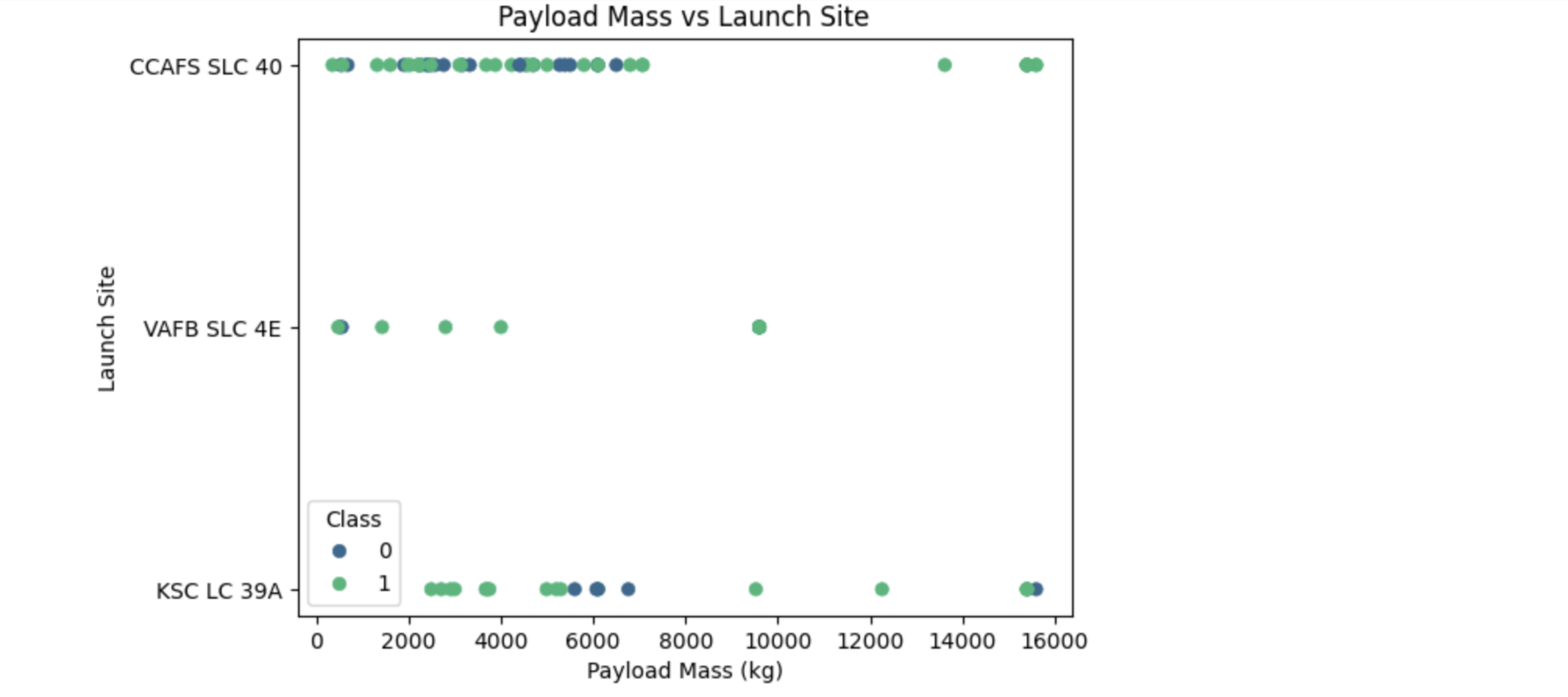
EDA with Data Visualization



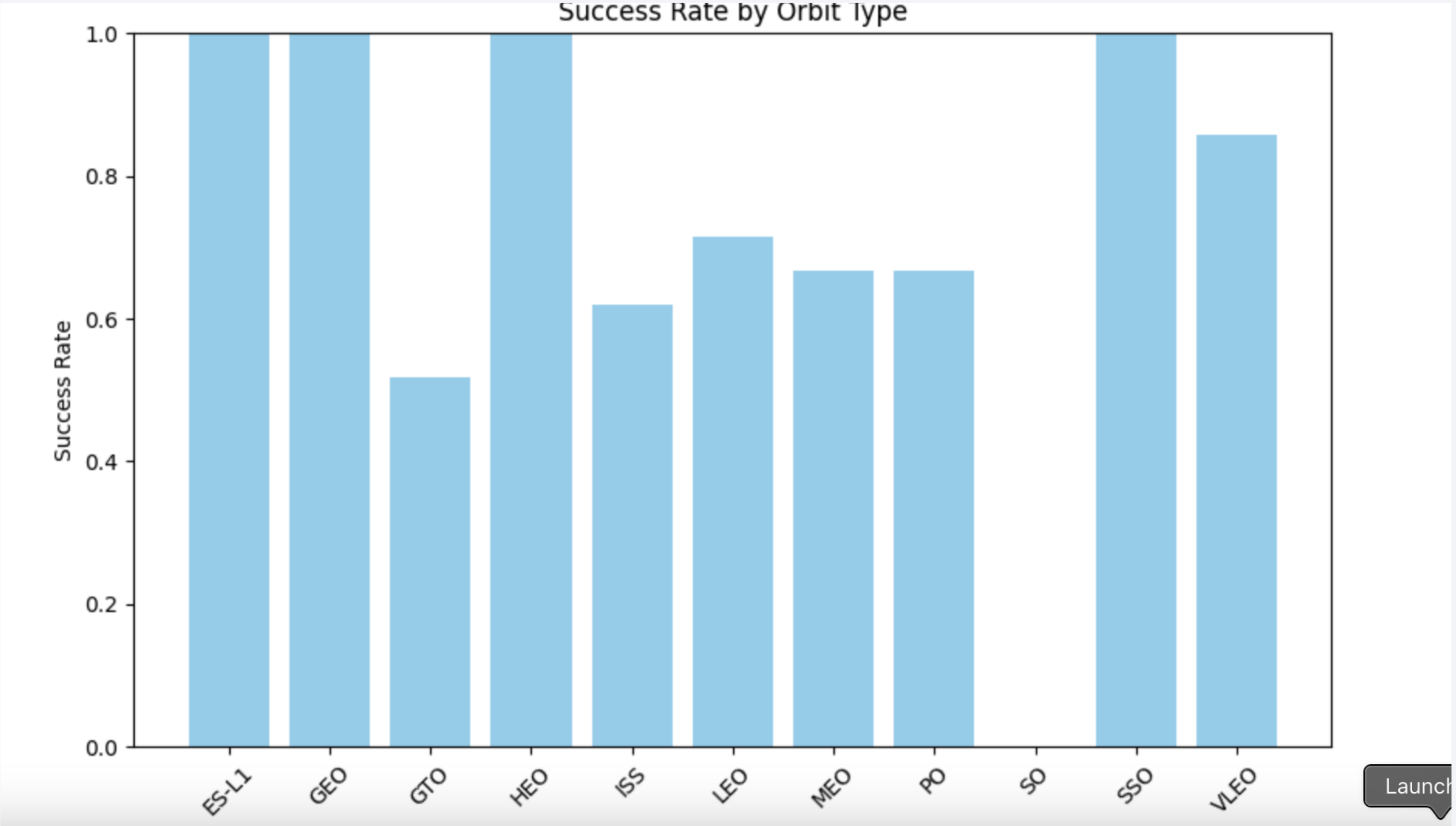
EDA with Data Visualization



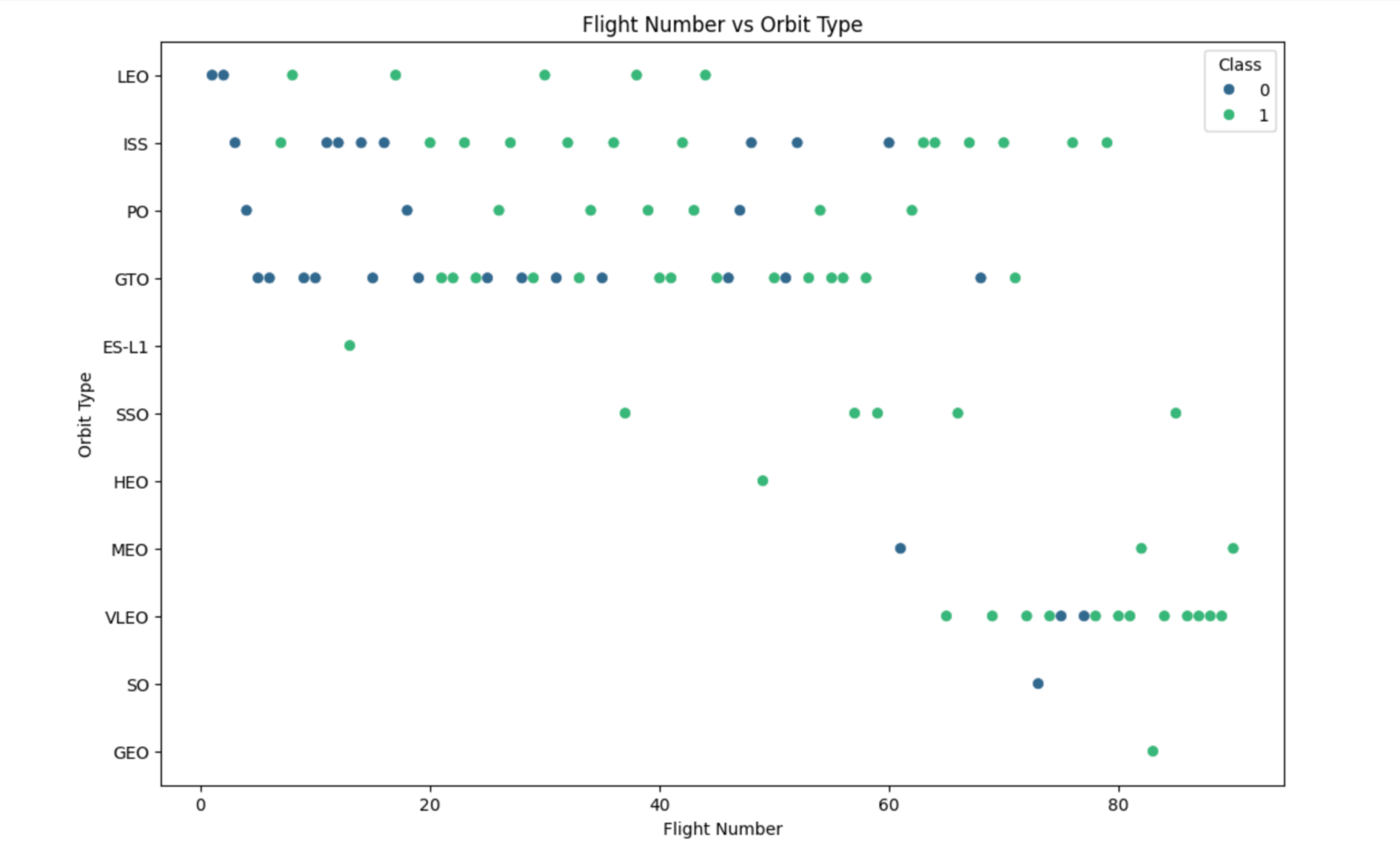
EDA with Data Visualization



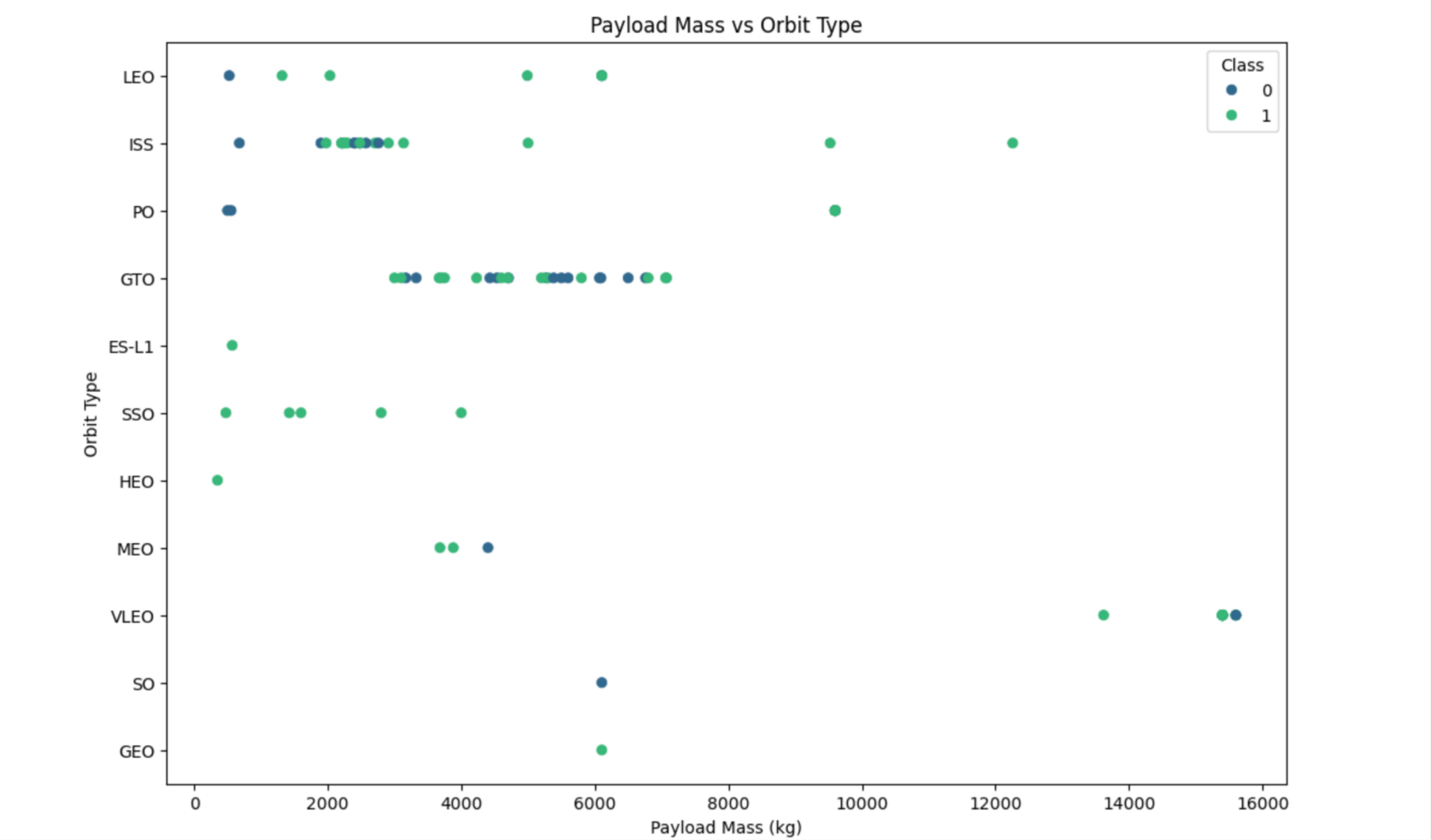
EDA with Data Visualization



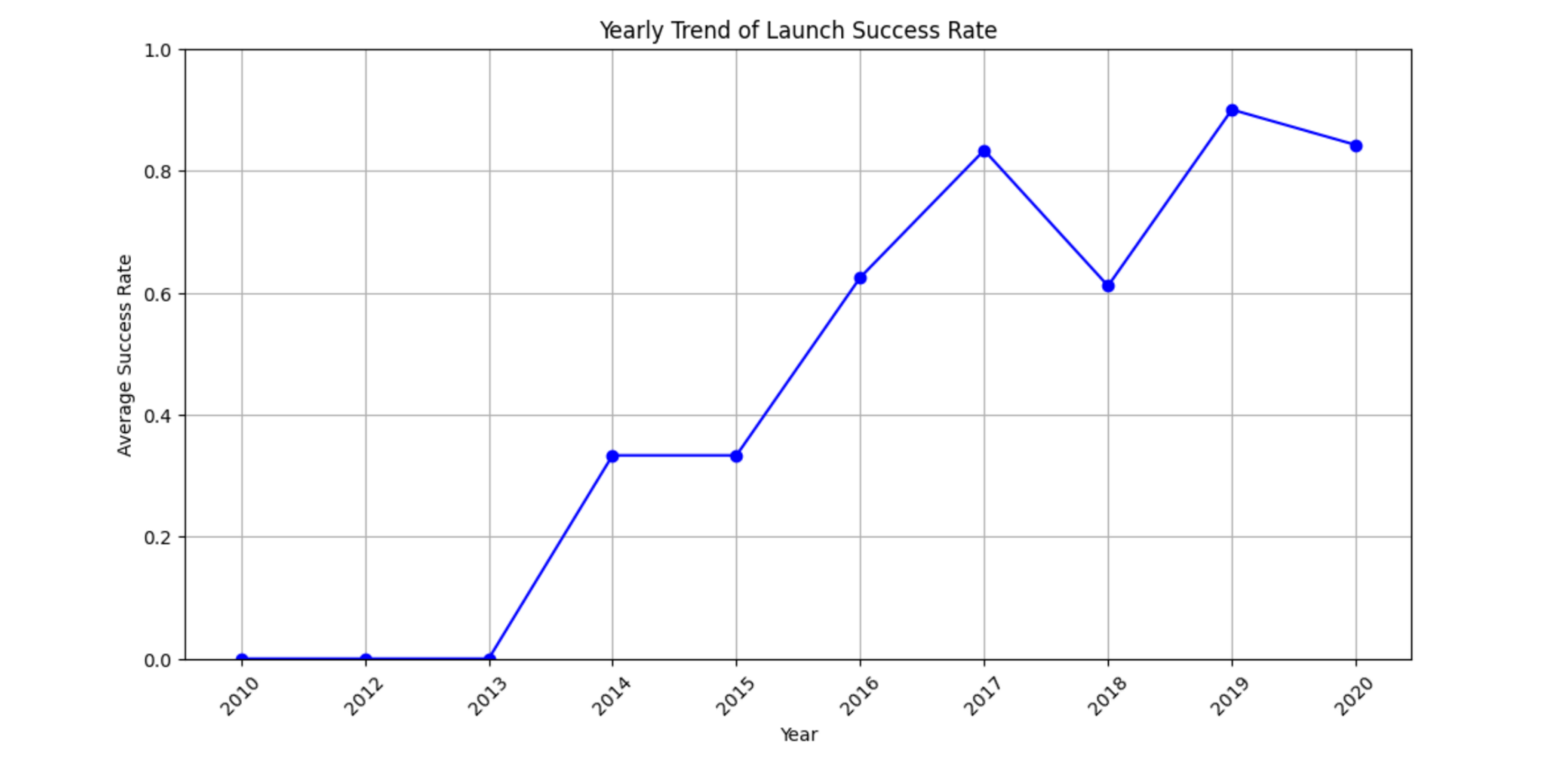
EDA with Data Visualization



EDA with Data Visualization



EDA with Data Visualization



EDA with SQL

- **Query to List Successful and Failed Mission Outcomes**
- **Query to Display Unique Values in Each Column**
- **Query to Find Booster Versions with Maximum Payload Mass**
- **Query to List Records for 2015 Failures on Drone Ships**
- **Query to Rank Landing Outcomes Between 2010-06-04 and 2017-03-20**
- **Query to Retrieve up to 20 Records**
- **Query to Display Minimum Payload Mass**
- **Query to Calculate Total Payload Mass by Booster Versions**
- **Query to Display Mission Outcome Counts for Each Launch Site**
- **Query to Display Unique Launch Sites**

- **Git:** CapstoneIBM/jupyter-labs-eda-sql-coursera_sqllite (1).ipynb at main · willubert/CapstoneIBM (github.com)

Build an Interactive Map with Folium

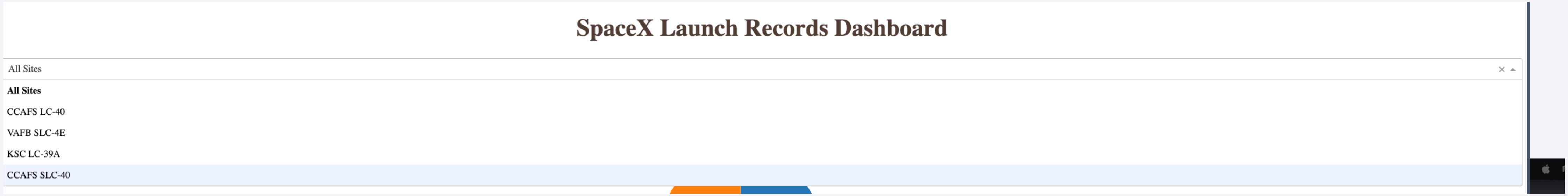
1. **Markers**:
   1. **Launch Site Marker**: A blue marker was added at the coordinates of the launch site. This marker helps in identifying the exact location of the launch site on the map.
   2. **Coastline Marker**: A red marker was added at the manually selected closest coastline point. This marker shows where the closest point on the coastline is relative to the launch site.
   3. **Point of Interest (POI) Marker**: A purple marker was added at the location of a point of interest (e.g., city, railway, highway). This marker helps identify important landmarks or infrastructures near the launch site.
2. **Circles**:
   1. **Space Center Circles**: A circle was drawn around Space Centers to highlight important locations on the map. The circle helps visualize the area around the center and make it stand out.
3. **Polylines**:
   1. **Launch Site to Coastline Polyline**: A green line was drawn connecting the launch site to the closest coastline marker. This line visually represents the distance and direction between the two points.
   2. **Launch Site to POI Polyline**: A blue line was drawn connecting the launch site to the point of interest (city, railway, highway). This line shows the relationship between the launch site and nearby infrastructures or landmarks.
4. **Marker Cluster**:
   1. **Launch Records Markers**: A cluster of markers was added to represent the locations of multiple launches, colored green for successful launches and red for failures. This clustering simplifies the visualization when multiple launches occur at the same coordina tes.

Git: [CapstoneIBM/lab_jupyter_launch_site_location.ipynb at main · willubert/CapstoneIBM (github.com)]()

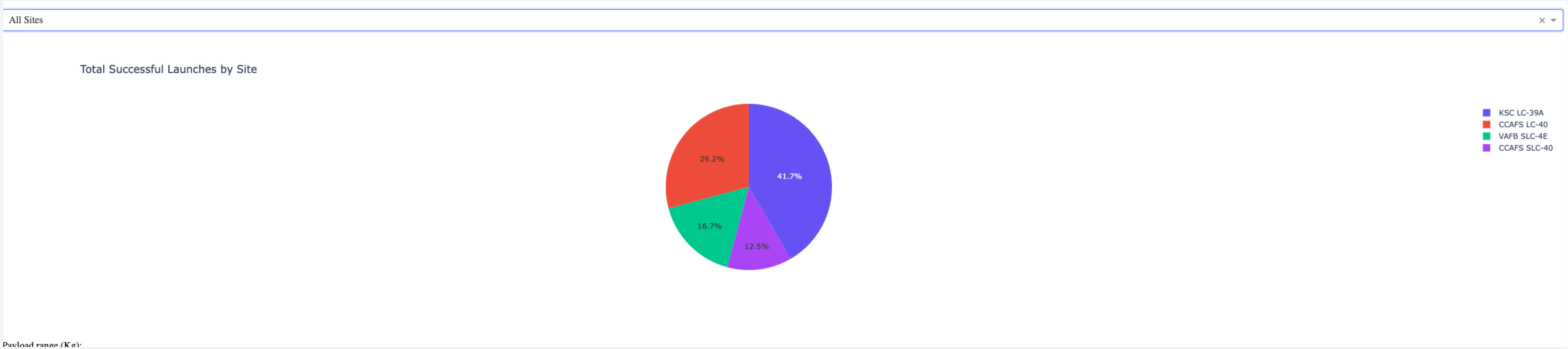
Build a Dashboard with Plotly Dash

- Similar to the data visualization we did using python, the graphs in the next couple of slides will show some of the charts created in the dash
- Git: [CapstoneIBM/spacex_dash_app.py at main · willubert/CapstoneIBM (github.com)]

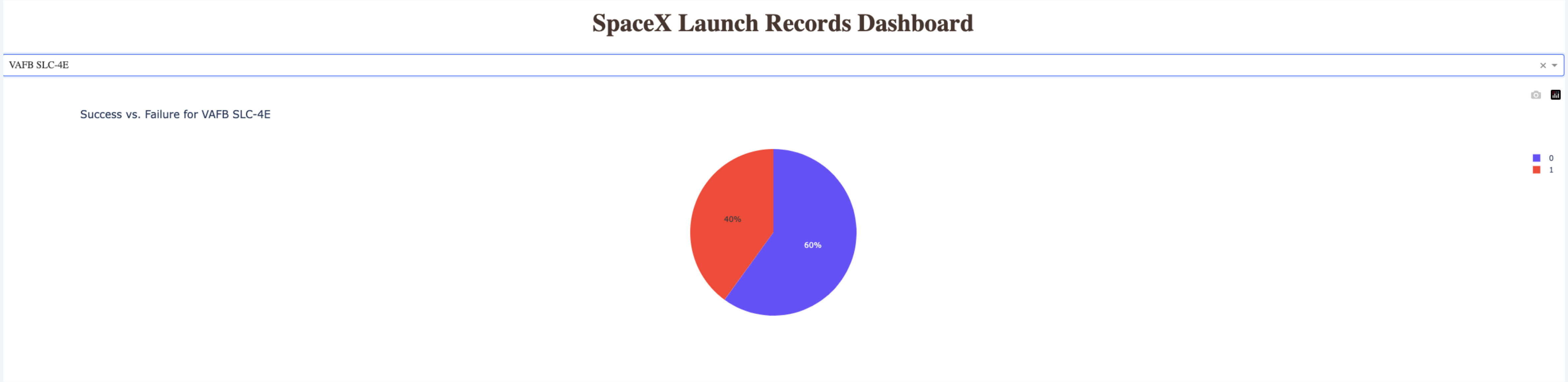
Build a Dashboard with Plotly Dash



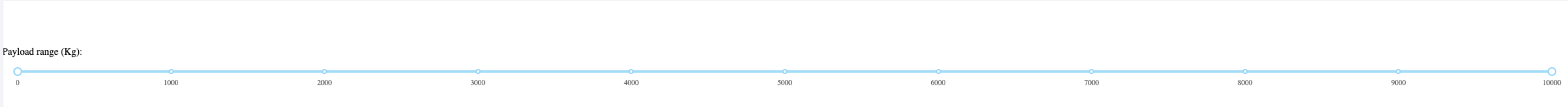
Build a Dashboard with Plotly Dash

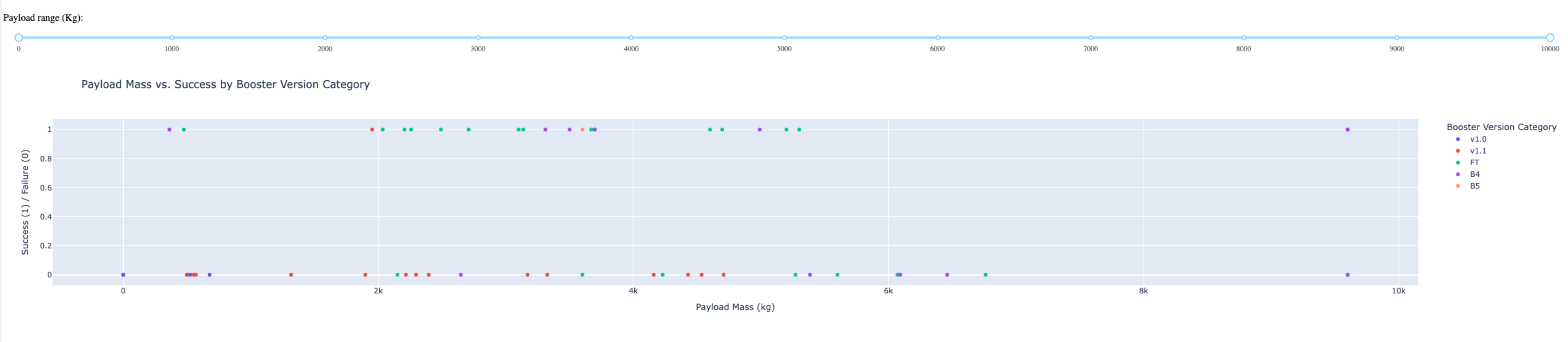


Build a Dashboard with Plotly Dash

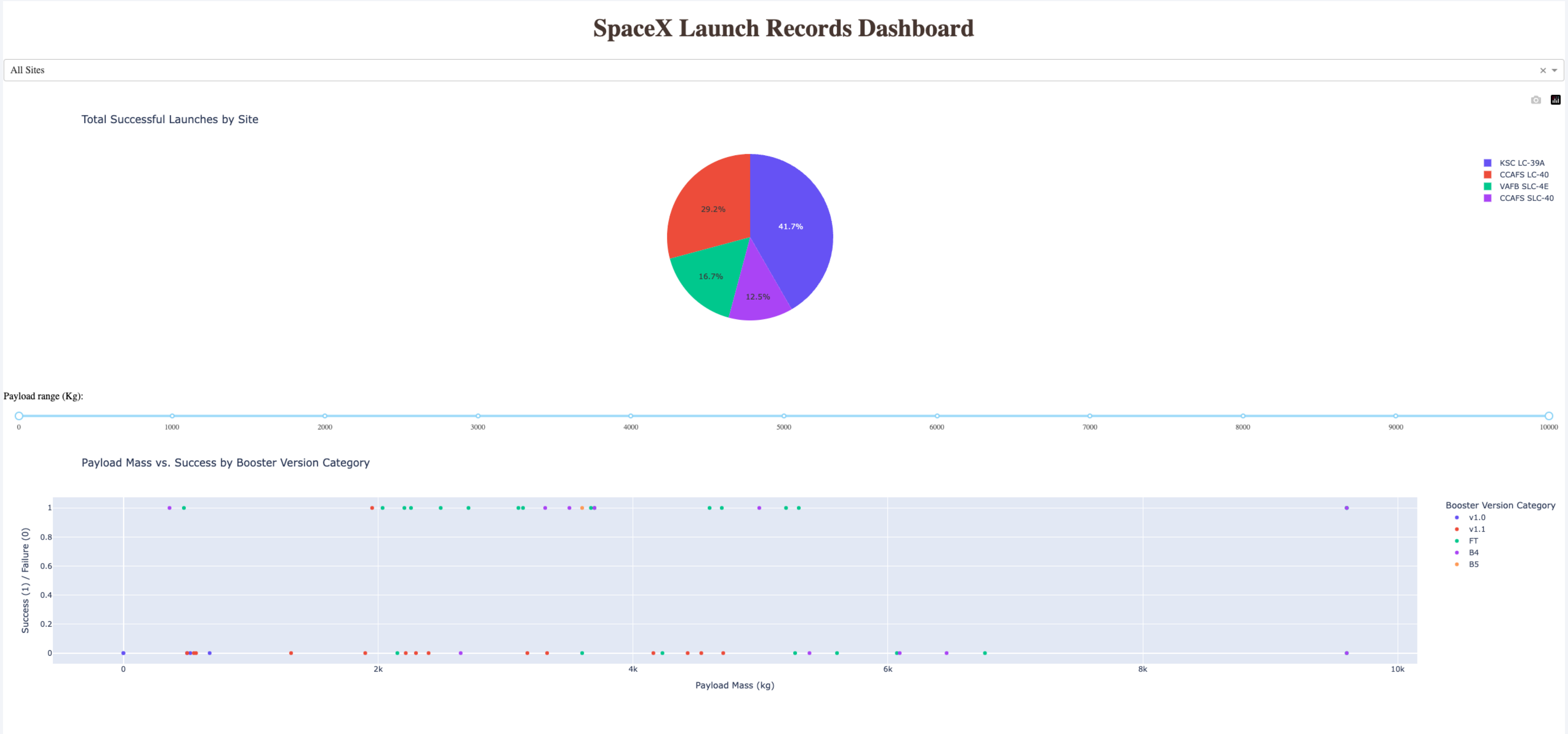


Build a Dashboard with Plotly Dash





Build a Dashboard with Plotly Dash



Predictive Analysis (Classification)

- **Standardization**: Applied StandardScaler to standardize the features in the dataset (X), ensuring all features had a mean of 0 and a standard deviation of 1.
- **Train-Test Split**: Used train_test_split to split the data into training and testing sets, with 80% for training and 20% for testing (X_train, X_test, Y_train, Y_test).
- **Logistic Regression**: Created a LogisticRegression object, and used GridSearchCV with a parameter grid to find the best hyperparameters.
- **Support Vector Machine (SVM)**: Created an SVC object, applied GridSearchCV to explore different kernels (linear, rbf, sigmoid) and other hyperparameters.
- **Decision Tree**: Built a DecisionTreeClassifier, and employed GridSearchCV to find the optimal criterion, splitter, and other tree-specific parameters.
- **K-Nearest Neighbors (KNN)**: Though not detailed earlier, a similar process would involve creating a KNeighborsClassifier and tuning its parameters with GridSearchCV.
- **Cross-Validation**: Used 10-fold cross-validation (cv=10) in GridSearchCV for each model to ensure robustness and avoid overfitting.
- **Accuracy Calculation**: Calculated accuracy for each model using the score method on the test set (X_test, Y_test).
- **Best Parameters Identification**: For each model, retrieved the best parameters from GridSearchCV using the best_params_ attribute.
- **Model Comparison**: Compared the test set accuracy of all models to identify the best performing model.
- **Kernel Selection for SVM**: Identified the best kernel for the SVM model by examining the best_params_ and validation scores.
- **Best Model Selection**: Selected the model with the highest accuracy on the test set as the final model for deployment or further analysis.

- Git: CapstoneIBM/SpaceX_Machine Learning Prediction_Part_5.ipynb at main · willubert/CapstoneIBM (github.com)

All Launch Site Names

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

[14]:
```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;
```

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

[14]:

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

Launch Site Names Begin with 'CCA'

Task 2

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

```
[31]: %sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;
```

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

[31]:

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

Total Payload Mass

- %sql SELECT SUM("PAYLOAD_MASS__KG_") AS total_payload_mass FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)';
- Total Payload Mass: 45596

Average Payload Mass by F9 v1.1

- %sql SELECT AVG("PAYLOAD_MASS__KG_") AS average_payload_mass FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1’;
- Average Payload Mass: 2928.4

First Successful Ground Landing Date

- %sql SELECT MIN("Date") AS first_successful_landing_date FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)';
- First Successful Landing Date: 12-22-2015

Successful Drone Ship Landing with Payload between 4000 and 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;

[28]:

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

Total Number of Successful and Failure Mission Outcomes

- %sql SELECT "Mission_Outcome", COUNT(*) AS total_count FROM SPACEXTABLE GROUP BY "Mission_Outcome";

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

Boosters Carried Maximum Payload

- %sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_")FROM SPACEXTABLE);

[32]:

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

- %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' WHEN '10' THEN 'October' WHEN '11' THEN 'November' WHEN '12' THEN 'December' END AS Month_Name, "Landing_Outcome", "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE substr("Date", 1, 4) = '2015' AND "Landing_Outcome" LIKE 'Failure (drone ship)';

[33]:

Month_Name	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

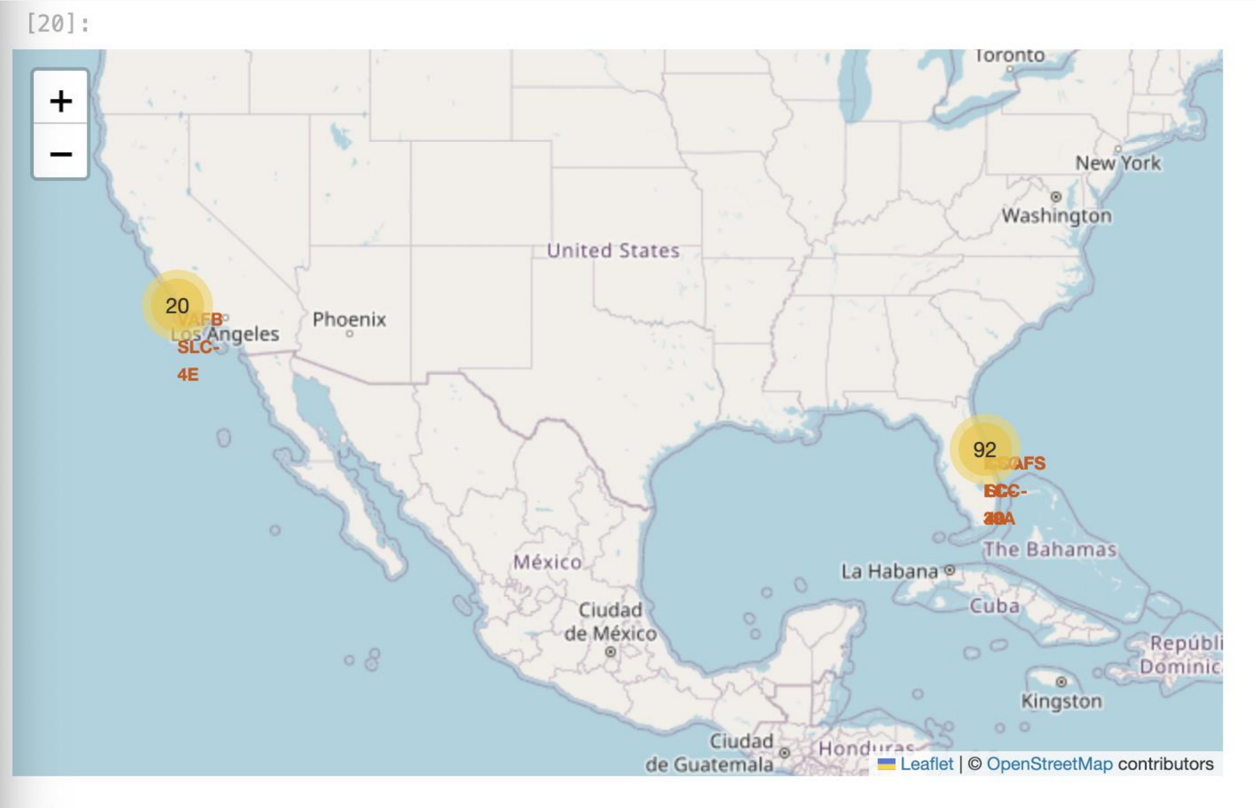
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- %sql SELECT "Landing_Outcome", COUNT(*) as Outcome_Count FROM SPACEXTBL WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY "Landing_Outcome" ORDER BY Outcome_Count DESC;

[15]:

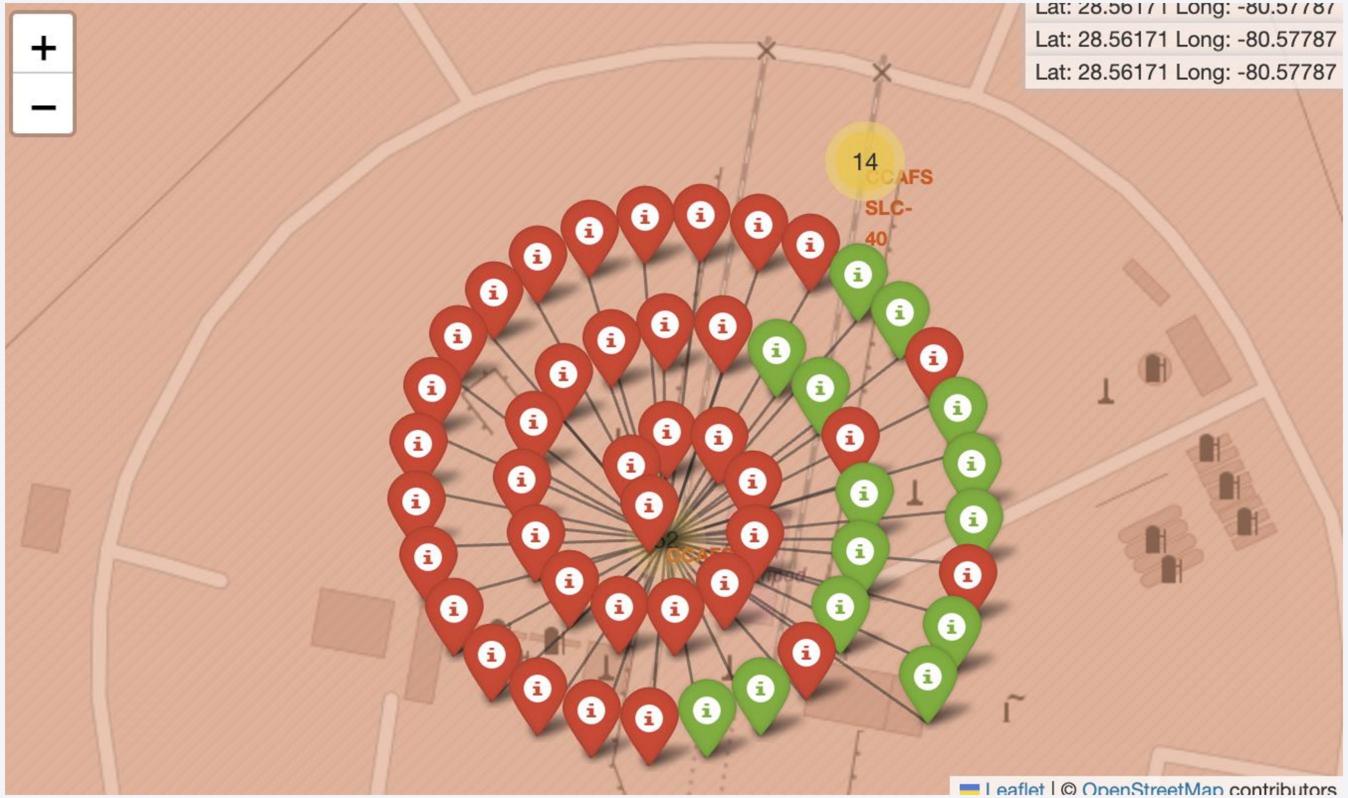
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

All launch sites on a map



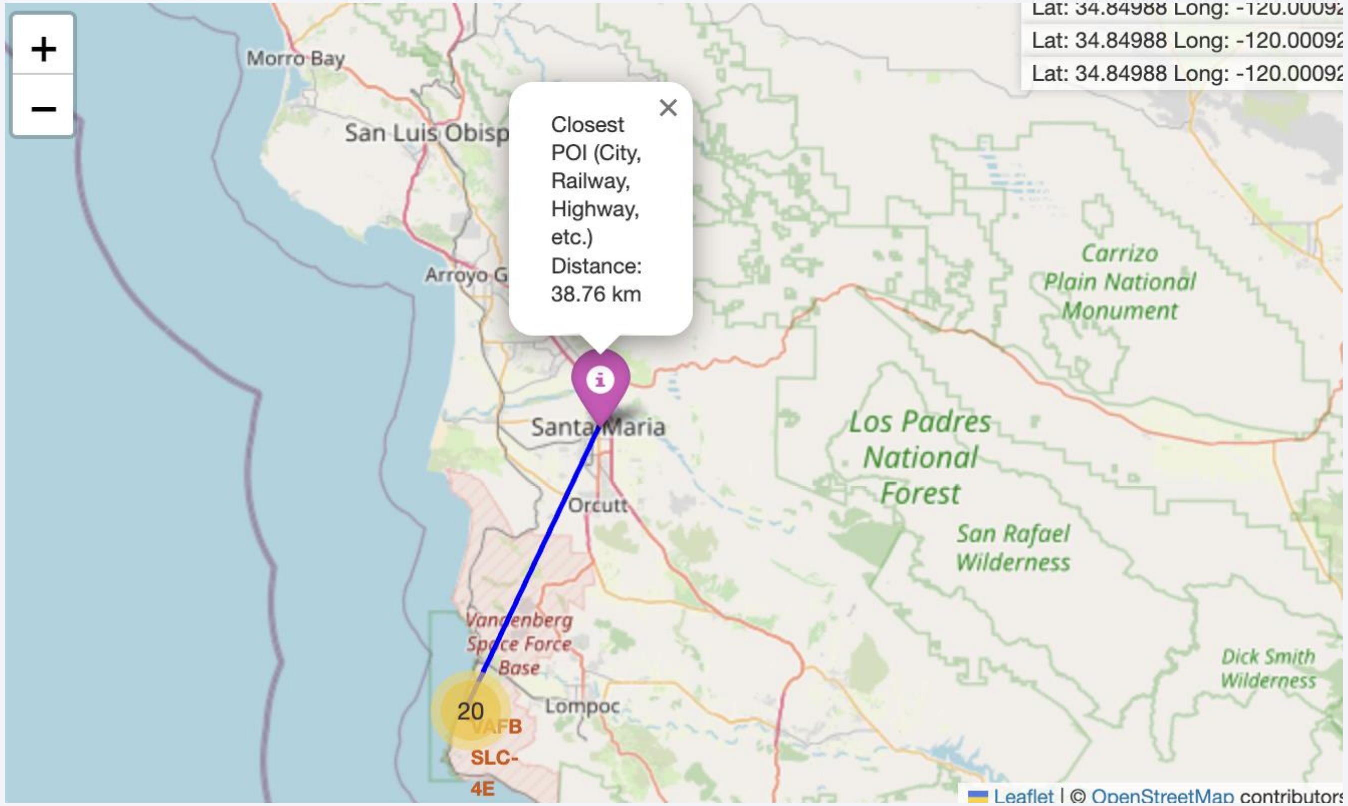
- This screenshot shows all the different launch sites, and as you can see, all of them are by the ocean. This is to prevent catastrophic impact of a failed rocket.

Launches at each site



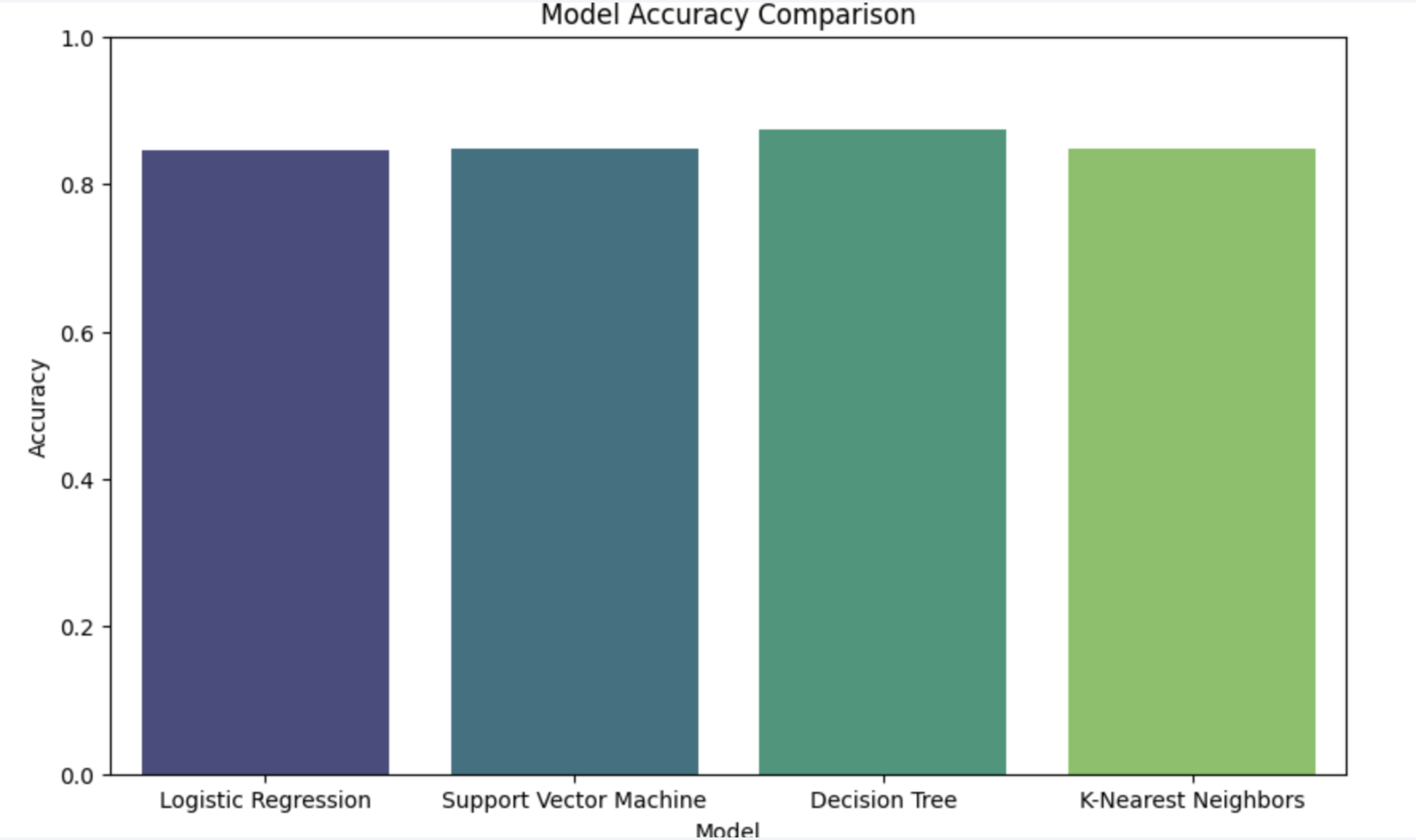
- This is a very useful map image of all the flights that succeeded and failed out of each launch site. This can help in our analysis when selecting a launch site for our company

Distance from closest city



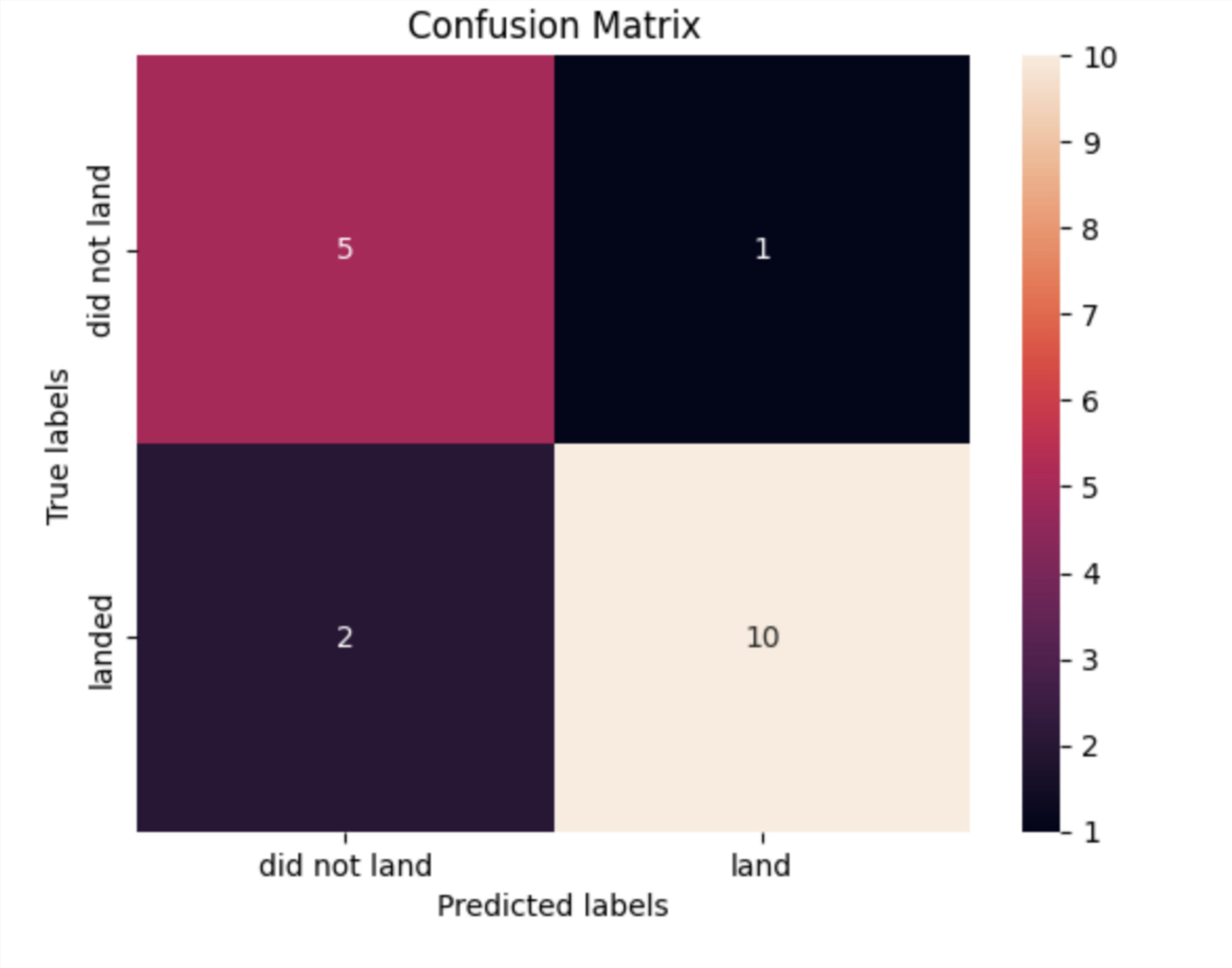
- This is important due to safety of launch and the distance from the nearest big city in case of catastrophic failure.

Classification Accuracy



- Decision Tree will have the most accurate model

Confusion Matrix



Conclusions

- We want to launch our rockets from site from CCAFS SLC-40 because that site has the highest percent of successful launches.
- Ww would want to keep the payload between 2000K and 4000K due to the highest number of successes falling in that range compared to outside of that range where failures outrank successes.
- When looking at boosters, we want to stick with Model FT because that has the highest track record of successes.

Appendix

- Git Repo: willubert/CapstoneIBM (github.com)

Thank you!