IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Seth Nelson
06-14-2023

Outline

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

Executive Summary

This project gathered and explored data related to SpaceX launch missions. The data was collected via API’s as well as from webscraping. Data analysis was completed using pandas and insights were gleamed using visuals from various plots, a dashboard, and maps.

Using a few machine learning models, we were able to assess some parsed data related to whether a particular rocket stage would land or not. Logistic, SVM, KNN, and Decision Tree classifiers were used on the data. It was found that the decision tree model gave the best prediction for the landing outcome based on the data at hand.

Introduction

In this project, we evaluated if SpaceX’s Falcon 9 first stage will land successfully. 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.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected via SpaceX's API at "https://api.spacexdata.com"
- Perform data wrangling
  - After parsing the API html output data, the information was compiled into a pandas DataFrame for further analysis.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Various classification methods were used to determine the best approach for estimating the success rate. The methods included logistic regression, SVM, KNN, and decision tree.

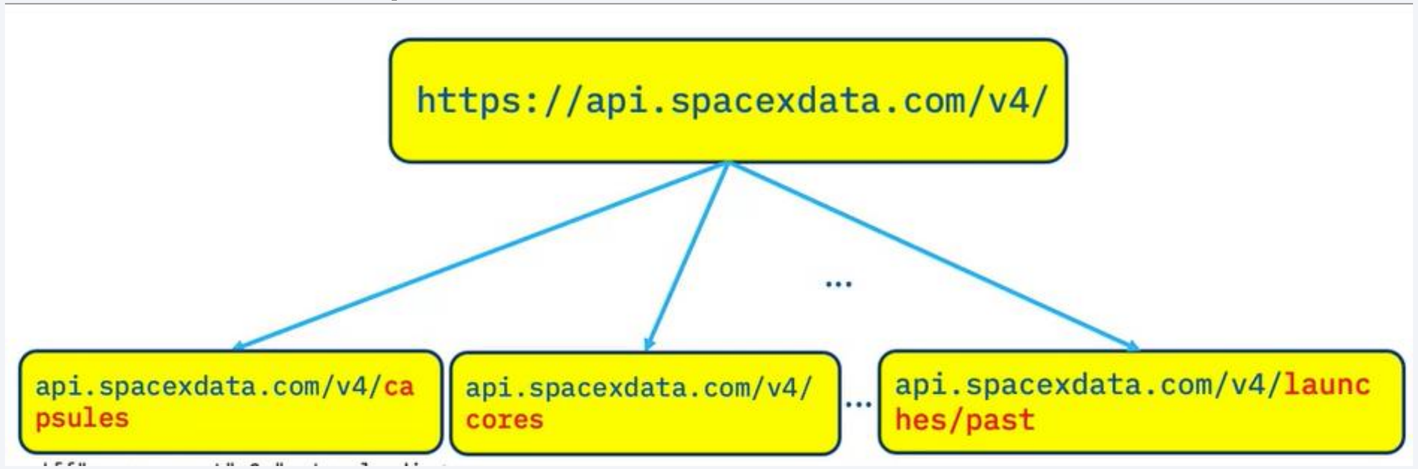
Data Collection

Data was collected via the SpaceX API as well as by webscraping an html table of data from a Wikipedia page on SpaceX launch history.

Data Collection – SpaceX API

- Data is collected using the API as shown below and sent to a pandas DataFrame for further processing.

SpaceX API Data Locations

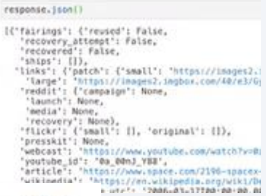



```
url="https://api.spacexdata.com/v4/launches/past"

response =requests.get(url)

response.json()

data = pd.json_normalize(response.json())
```



- Github link for completed Notebook: https://github.com/nelsonseth/IBM_DS/blob/main/capstone/capstone%20-%20data%20collection.ipynb

Data Collection - Scraping

- Using BeautifulSoup, html web data was collected from a wiki table, cleaned, and converted into a pandas DataFrame for further analysis.

Html Table

[hide] Flight No.	Date and time (UTC)	Version, Booster[b]	Launch site	Payload[c]	Payload mass	Orbit	Customer	Launch outcome	Booster landing
78	7 January 2020, 02:19:21[492]	F9 B5 ♺ B1049.4	CCAFS, SLC-40	Starlink 2 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)
	Third large batch and second operational flight of Starlink constellation. One of the 60 satellites included a test coating to make the satellite less reflective, and thus less likely to interfere with ground-based astronomical observations.[493]								
79	19 January 2020, 15:30[494]	F9 B5 ♺ B1046.4	KSC, LC-39A	Crew Dragon in-flight abort test[495] (Dragon C205.1)	12,050 kg (26,570 lb)	Sub-orbital[496]	NASA (CTS)[497]	Success	No attempt
	An atmospheric test of the Dragon 2 abort system after Max Q. The capsule fired its SuperDraco engines, reached an apogee of 40 km (25 mi), deployed parachutes after reentry, and splashed down in the ocean 31 km (19 mi) downrange from the launch site. The test was previously slated to be accomplished with the Crew Dragon Demo-1 capsule;[498] but that test article exploded during a ground test of SuperDraco engines on 20 April 2019.[419] The abort test used the capsule originally intended for the first crewed flight.[499] As expected, the booster was destroyed by aerodynamic forces after the capsule aborted.[500] First flight of a Falcon 9 with only one functional stage — the second stage had a mass simulator in place of its engine.								
80	29 January 2020, 14:07[501]	F9 B5 ♺ B1051.3	CCAFS, SLC-40	Starlink 3 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)
	Third operational and fourth large batch of Starlink satellites, deployed in a circular 290 km (180 mi) orbit. One of the fairing halves was caught, while the other was fished out of the ocean.[502]								
81	17 February 2020, 15:05[503]	F9 B5 ♺ B1056.4	CCAFS, SLC-40	Starlink 4 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Failure (drone ship)

DataFrame

[44]:
```
df.head()
```

[44]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX\n	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	\nNASA (COTS)\nNRO\n\n	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA (COTS)\n	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA (CRS)\n	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA (CRS)\n	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

- Github link for completed Notebook: https://github.com/nelsonseth/IBM_DS/blob/main/capstone/capstone%20-%20webscraping.ipynb

Data Wrangling

- Using the pandas DataFrame, the data was cleaned of null values. The original data contained additional outcome parameters such as landing zone type and whether or not the mission attempted to land at all. In order to apply this information to a classification scheme, the landing outcome definitions were compiled into a simple binary result: 0 for failure and 1 for success.

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	F
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

- Github link for completed Notebook: https://github.com/nelsonseth/IBM_DS/blob/main/capstone/capstone%20-%20data%20wrangling.ipynb

EDA with Data Visualization

- Data charts were used to explore the data:
  - Scatter plots of Flight Number vs. Payload Mass and Launch Site and Orbit Type
    - By coloring the markers by Class (success or fail), we could see if any trends occurred as more flights were conducted.
  - Bar chart of success rate for each Orbit type.
  - Scatter plot to show relationship between Orbit type and Payload.
  - Line chart showing success rate by year

- Github link for completed Notebook: https://github.com/nelsonseth/IBM_DS/blob/main/capstone/capstone%20-%20data%20viz.ipynb

EDA with SQL

- Some of the SQL Queries performed on the database include:
  - Determined the unique Launch Site names
  - Displayed 5 records from launch sites with “CAA” in the name
  - Determined the total payload mass and avg. payload for specific booster type.
  - Determined the total number of successes and failures
  - Determined which booster types carried the maximum payload.

- Github link for completed Notebook: https://github.com/nelsonseth/IBM_DS/blob/main/capstone/capstone%20-%20SQL.ipynb

Build an Interactive Map with Folium

- Folium was used to explore information related to the launch sites.
- The locations of the launch sites was marked and labeled on a map.
    - One location is in California and the other three are in Florida.
- Additional marker clusters were added to showcase the various successful and failed landing outcomes at each site. By zooming in and clicking on each site, you can see more detail as a result of the clusters.
- Polylines were used to show distances between the sites and nearby locations, such as a coastline.

- Github link for completed Notebook: https://github.com/nelsonseth/IBM_DS/blob/main/capstone/capstone%20-%20site%20locations%20Folium.ipynb

Build a Dashboard with Plotly Dash

- Using dash, we built a dashboard that is accessible via a web browser when run.
- In the dashboard were a number of items:
  - A dropdown list allowing the user to select one of the launch sites or select all of them.
  - A bar chart showing the success rate of each site or all the sites, depending on the dropdown list selection.
  - A slider allowing the user to select different ranges of payload values.
  - A scatter plot showing the landing outcome for the launch sites.
    - The scatter plot either showed all the data or specific site data for a given payload range, depending on the user selections.

- Github link for completed dash python code: https://github.com/nelsonseth/IBM_DS/blob/main/capstone/capstone%20-%20spacex_dash_app.py

Predictive Analysis (Classification)

- Four different classification schemes were used on the data.
  - Logistic Regression
  - SVM
  - KNN
  - Decision Tree
- Each method was optimized via a GridSearchCV function that iterates through a list of parameters to find which combination performed the best.
- A confusion matrix was then plotted for each method when used on the test data to see how well each method performed after fitting.

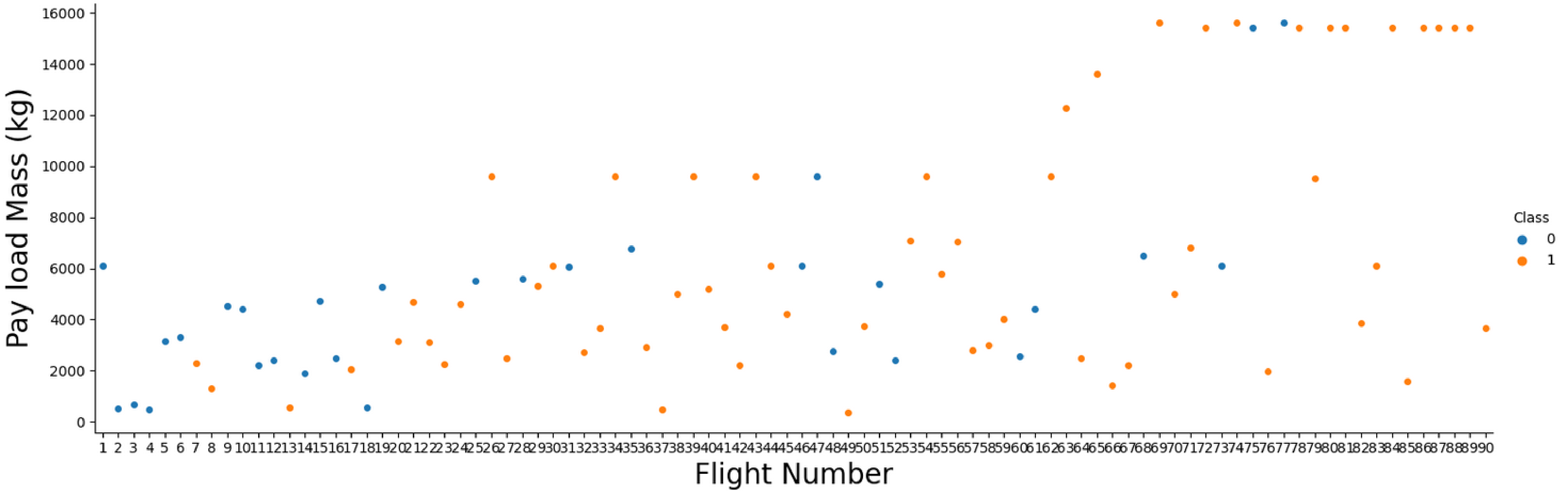
- Github link for completed Notebook: https://github.com/nelsonseth/IBM_DS/blob/main/capstone/capstone%20-%20Machine%20Learning.ipynb

Section 2

Insights drawn from EDA

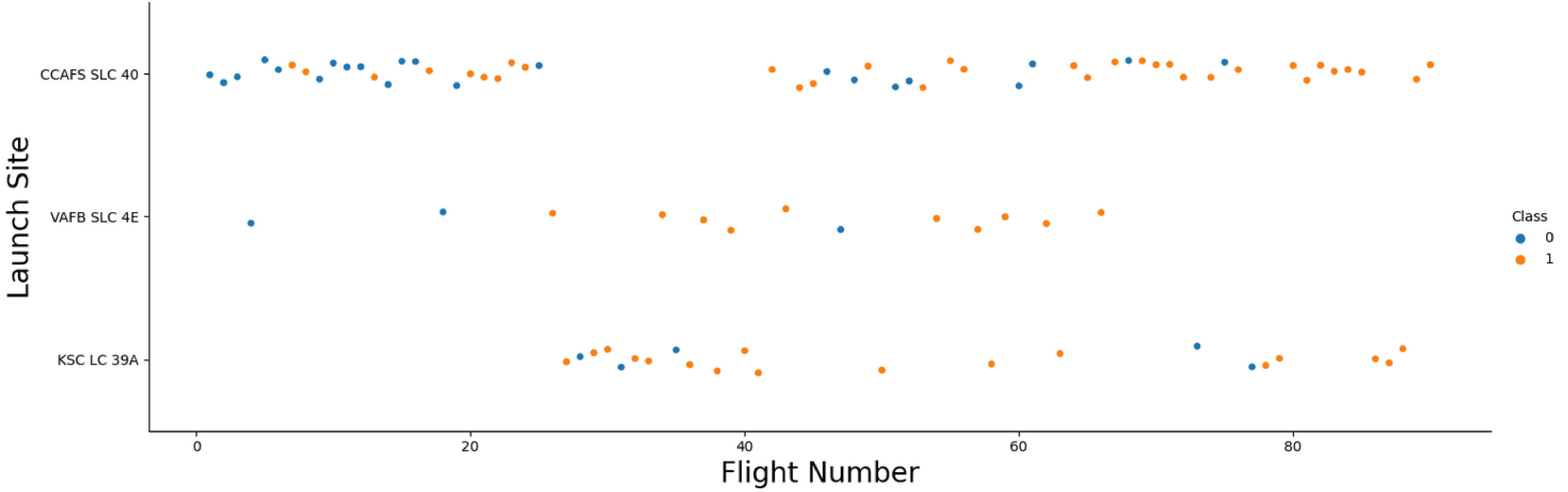
Flight Number vs. Payload Mass

```
]: sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 3)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



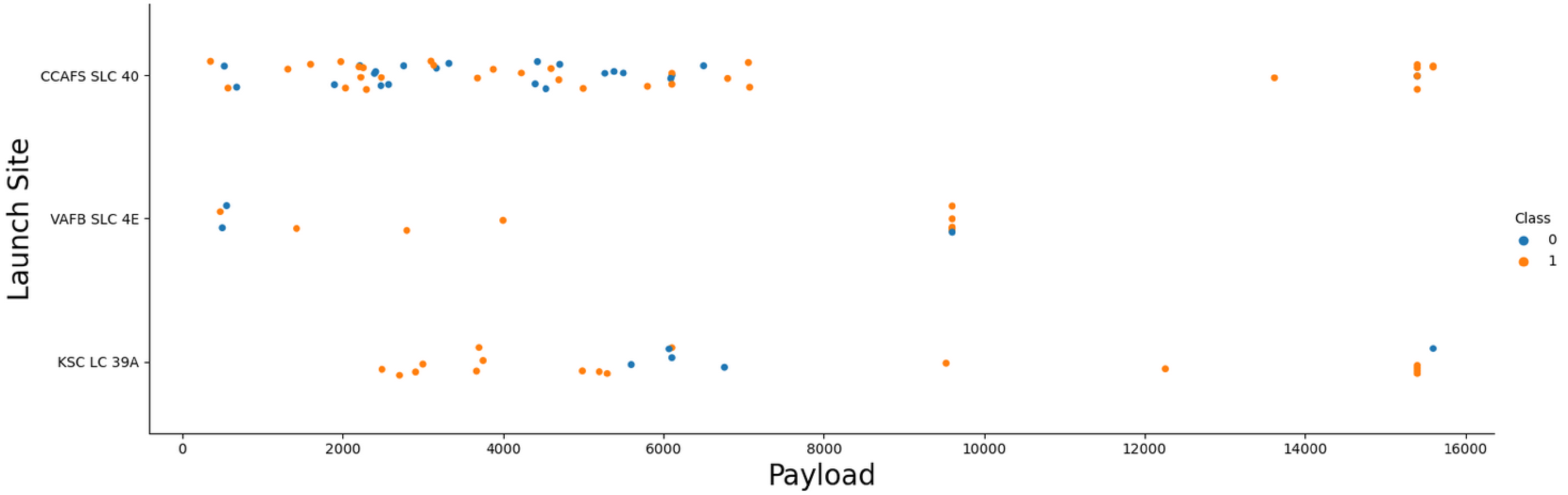
Flight Number vs. Launch Site

```
[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 value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect=3)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



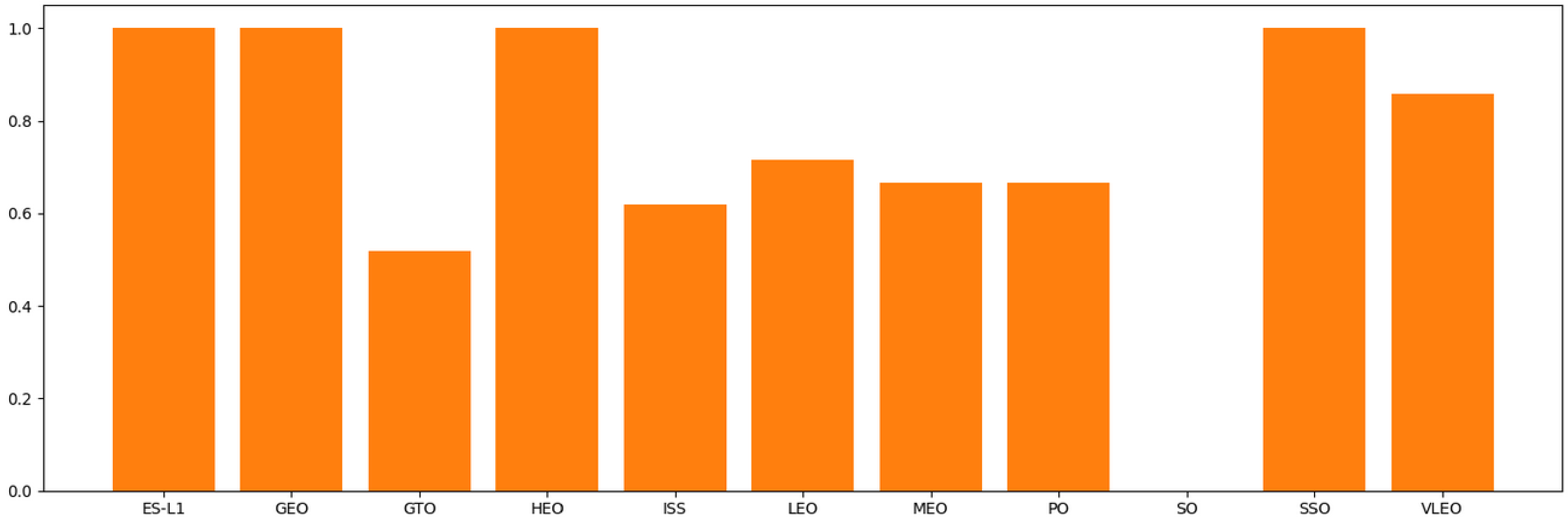
Payload vs. Launch Site

```
[8]: # 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=3)
plt.xlabel("Payload",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



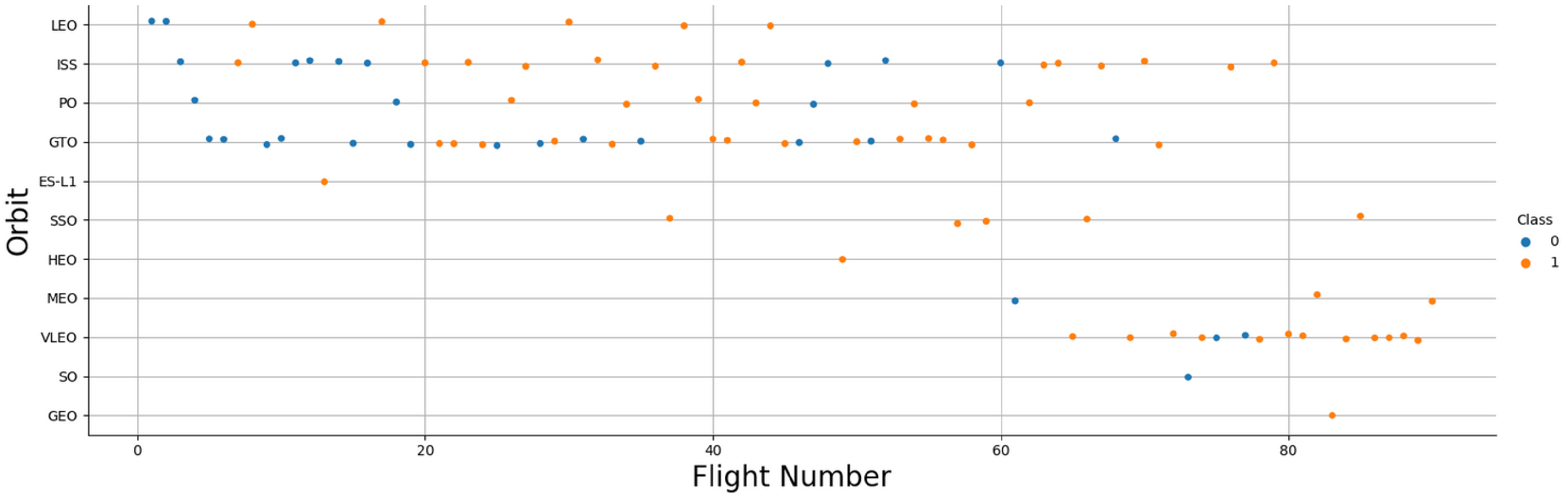
Success Rate vs. Orbit Type

```
# HINT use groupby method on Orbit column and get the mean of Class column
bardata = df.groupby(['Orbit']).mean()['Class']
x = bardata.keys()
h = bardata.values
plt.bar(x=x, height=h)
plt.show()
```



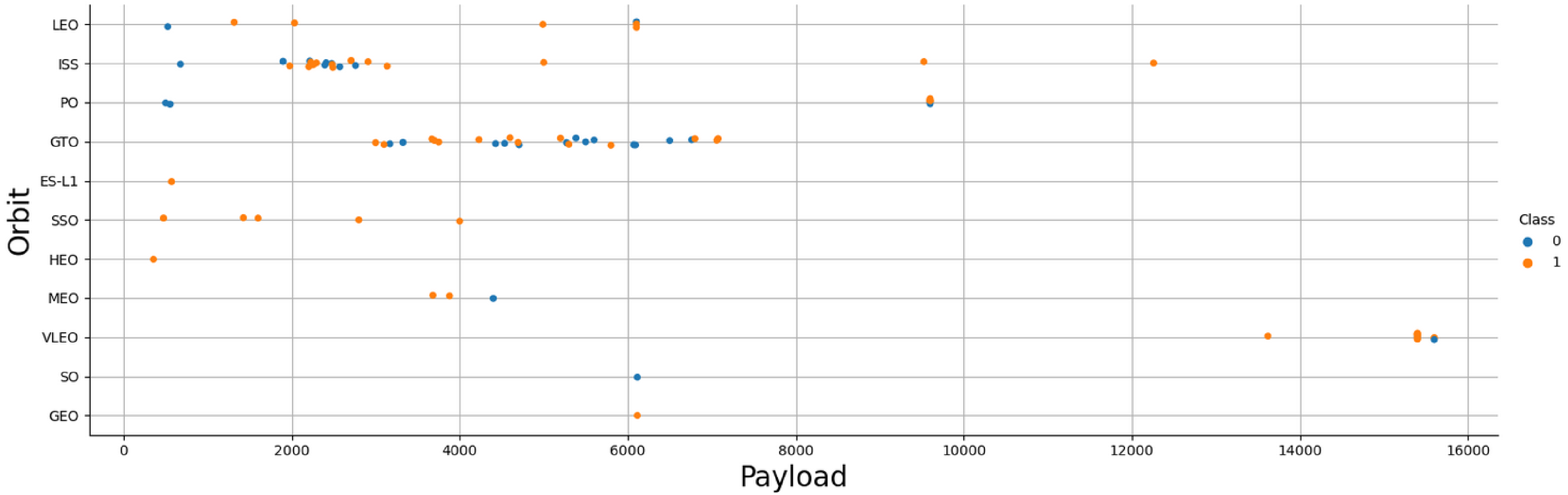
Flight Number vs. Orbit Type

```
[11]: # 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.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect=3)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.grid()
plt.show()
```



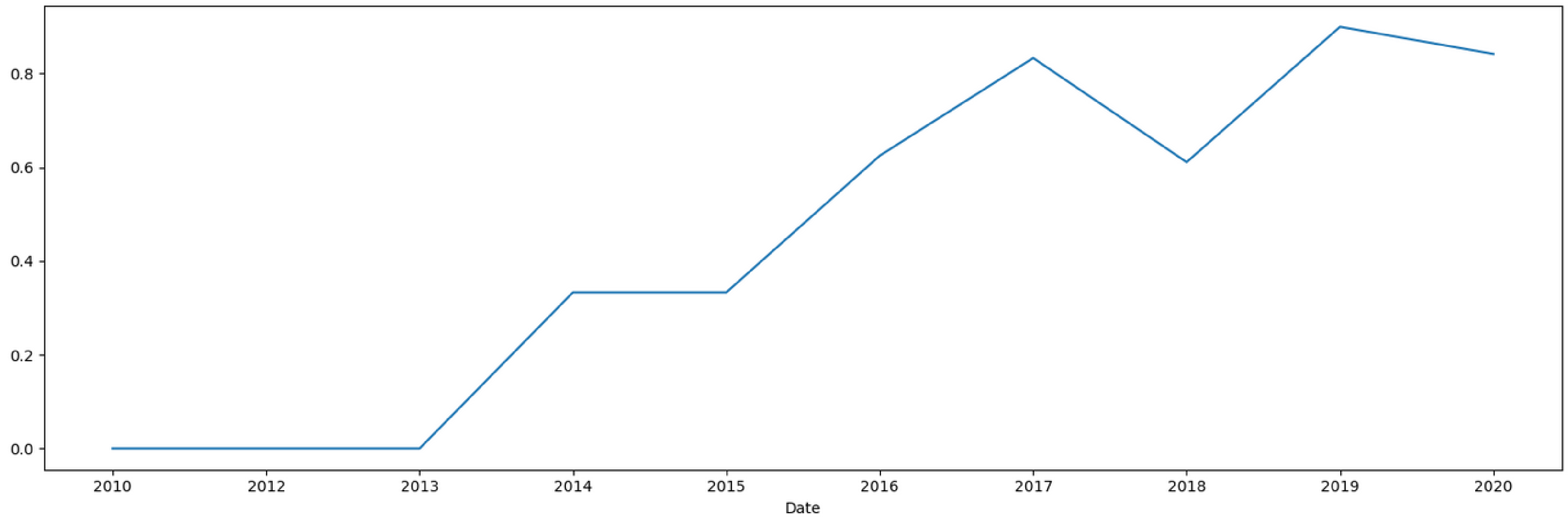
Payload vs. Orbit Type

```
[12]: # 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=3)
      plt.xlabel("Payload",fontsize=20)
      plt.ylabel("Orbit",fontsize=20)
      plt.grid()
      plt.show()
```



Launch Success Yearly Trend

```
]: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
linedata = df.groupby(['Date']).mean()['Class']
x = linedata.keys()
y = linedata.values
sns.lineplot(x=x, y=y)
plt.show()
```



All Launch Site Names

```
[15]: %sql select distinct "Launch_Site" from SPACEXTBL;

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

[15]:

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
None

Launch Site Names Begin with 'CCA'

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

[16]:
```
%%sql
select * from SPACEXTBL
where "Launch_Site" like "CCA%"
limit 5;
```

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

[16]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

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

[23]:
```
%%sql
select sum("PAYLOAD_MASS__KG_") from SPACEXTBL
where "Customer" like "%NASA (CRS)%";
```

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

[23]:

sum("PAYLOAD_MASS__KG_")
48213.0

Average Payload Mass by F9 v1.1

Task 4

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

[25]:
```
%%sql
select avg("PAYLOAD_MASS__KG_") from SPACEXTBL
where "Booster_Version" like "F9 v1.1%";
```

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

[25]:

avg("PAYLOAD_MASS__KG_")
2534.6666666666665

First Successful Ground Landing Date

```
[ ]: # the min function will not work with these dates as formatted. The database needs to be re-formatted with usable datatime information.
     # Since the data is small, we can just pull all ground pad successes and find the earliest date, which is 22/12/2015.
```

```
[20]: %%sql
      select "Date", "Landing_Outcome" from SPACEXTBL
      where "Landing_Outcome" == "Success (ground pad)";
```

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

[20]:

Date	Landing_Outcome
22/12/2015	Success (ground pad)

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

[30]:
```
%%sql
select distinct "Booster_Version" from SPACEXTBL
where "Landing_Outcome" == "Success (drone ship)" and "payload_mass__kg_" between 4000 and 6000;
```

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

[30]:

Booster_Version
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

[35]:
```
%%sql
select "mission_outcome", count("mission_outcome") as "Count" from SPACEXTBL
group by "mission_outcome";
```

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

[35]:

Mission_Outcome	Count
None	0
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

[37]:
```
%%sql
select distinct "booster_version" from SPACEXTBL
where "payload_mass__kg_" == (select max("payload_mass__kg_") from SPACEXTBL);
```

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

[37]:

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

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, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

[38]:
```
%%sql
select substr("Date", 4,2) as "Month", "landing_outcome", "booster_version", "launch_site" from SPACEXTBL
where "landing_outcome" == "Failure (drone ship)" and substr("Date", 7,4) == "2015";
```

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

[38]:

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

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

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

[ ]:
```
# again, given the datetime information provided and the limited ability to manipulate it, this is a pretty painful work-around.
```

22]:
```
%%sql
select "landing_outcome", count("landing_outcome") as "Count" from SPACEXTBL
where "landing_outcome" like "%Success%" and "Date" between strftime('%d/%m/%Y', "2010-06-04") and strftime('%d/%m/%Y', "2017-03-20")
group by "landing_outcome"
order by "Count" Desc;
```

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

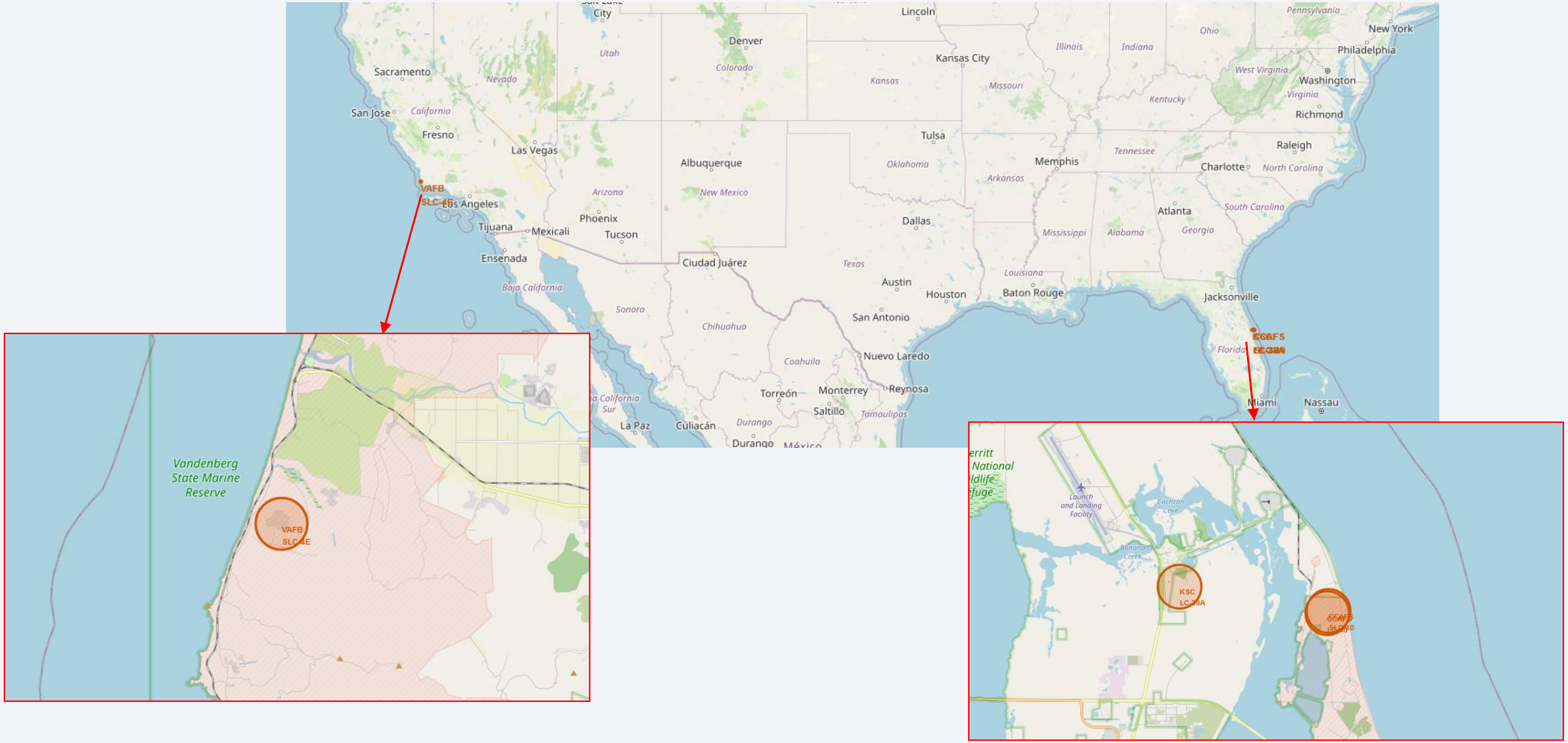
22]:

Landing_Outcome	Count
Success	20
Success (drone ship)	8
Success (ground pad)	7

Section 3

Launch Sites Proximities Analysis

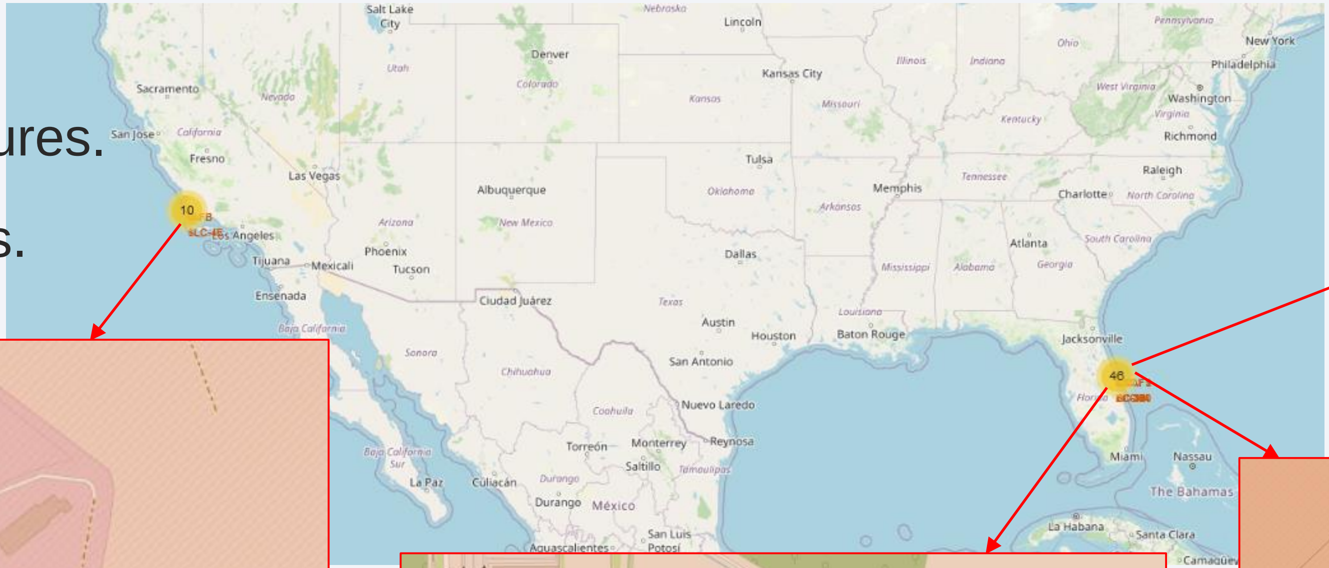
Folium Markers Showing the Launch Site Locations

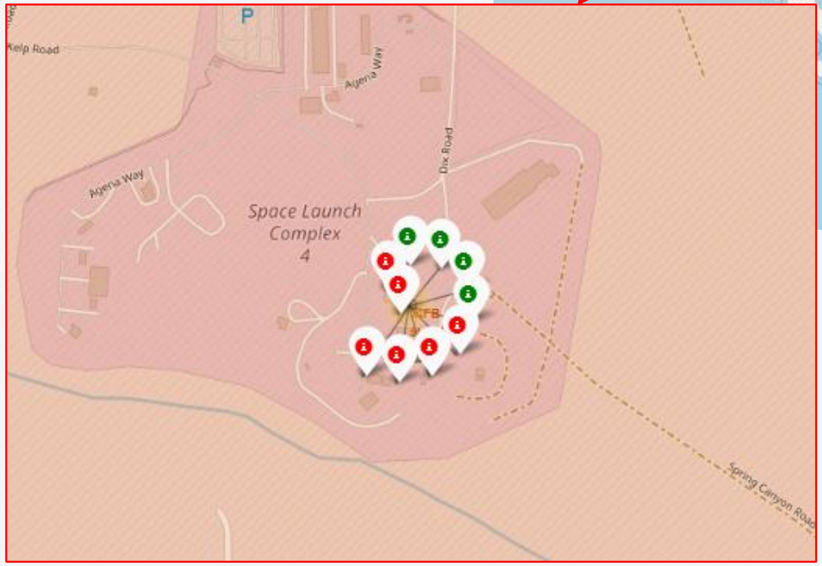


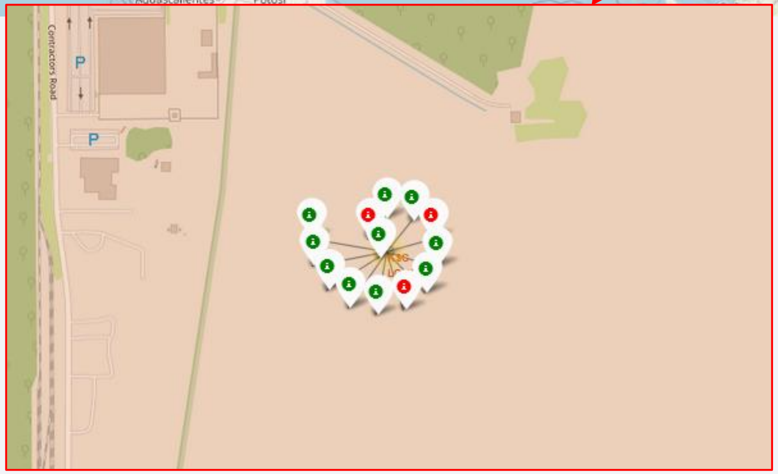
Folium Marker Clusters showing the success and failures at each site.

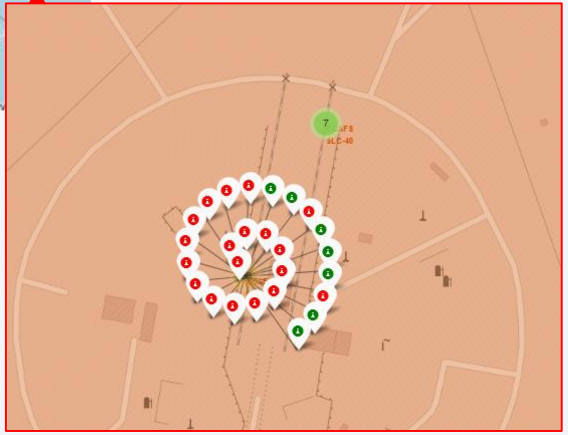
Red markers are failures.

Green are successes.









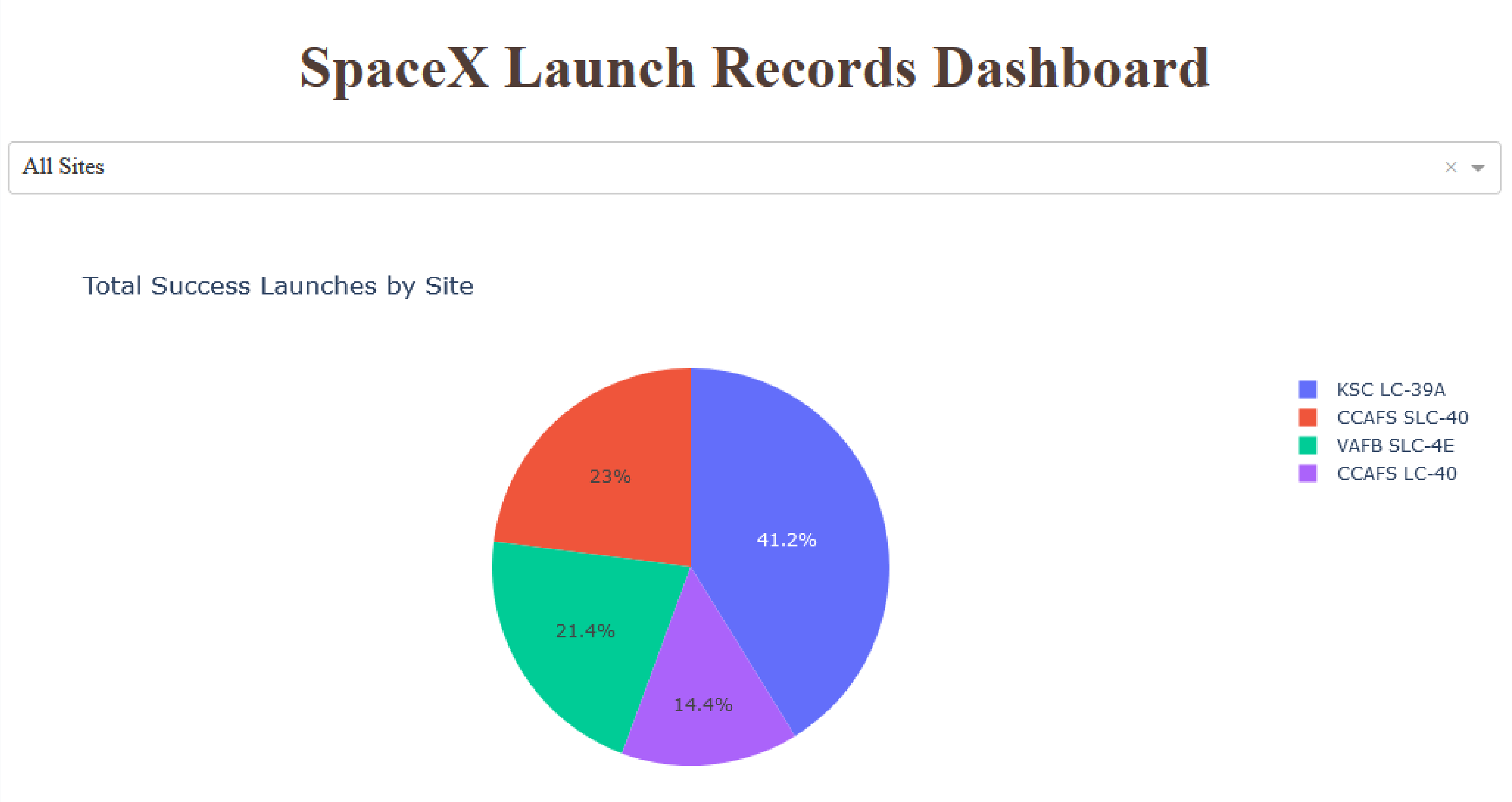
Folium Polyline to draw distance to nearby coastline.

- Example of a drawn line from a launch site to a nearby coastline. The text is small, but the distance is also shown as 0.51 km.

Section 4

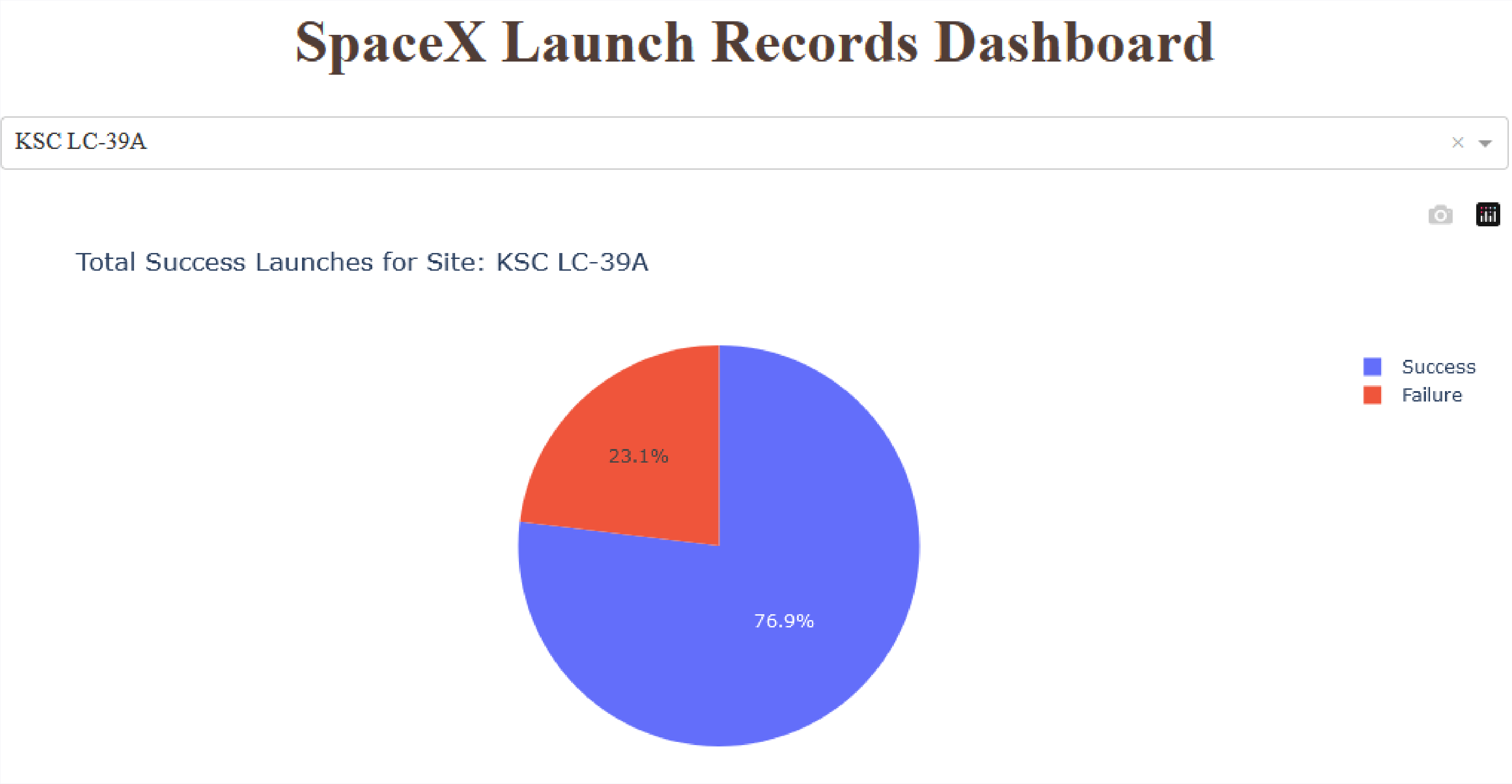
Build a Dashboard with Plotly Dash

Success Rates for All Launch Sites



- Launch Site KSC LC-39A has the highest rate of successful launches.

Most Successful Launch Site



- Launch Site KSC LC-39A has the highest rate of successful launches overall, but still had some failures. About 23% of launches failed from this site.

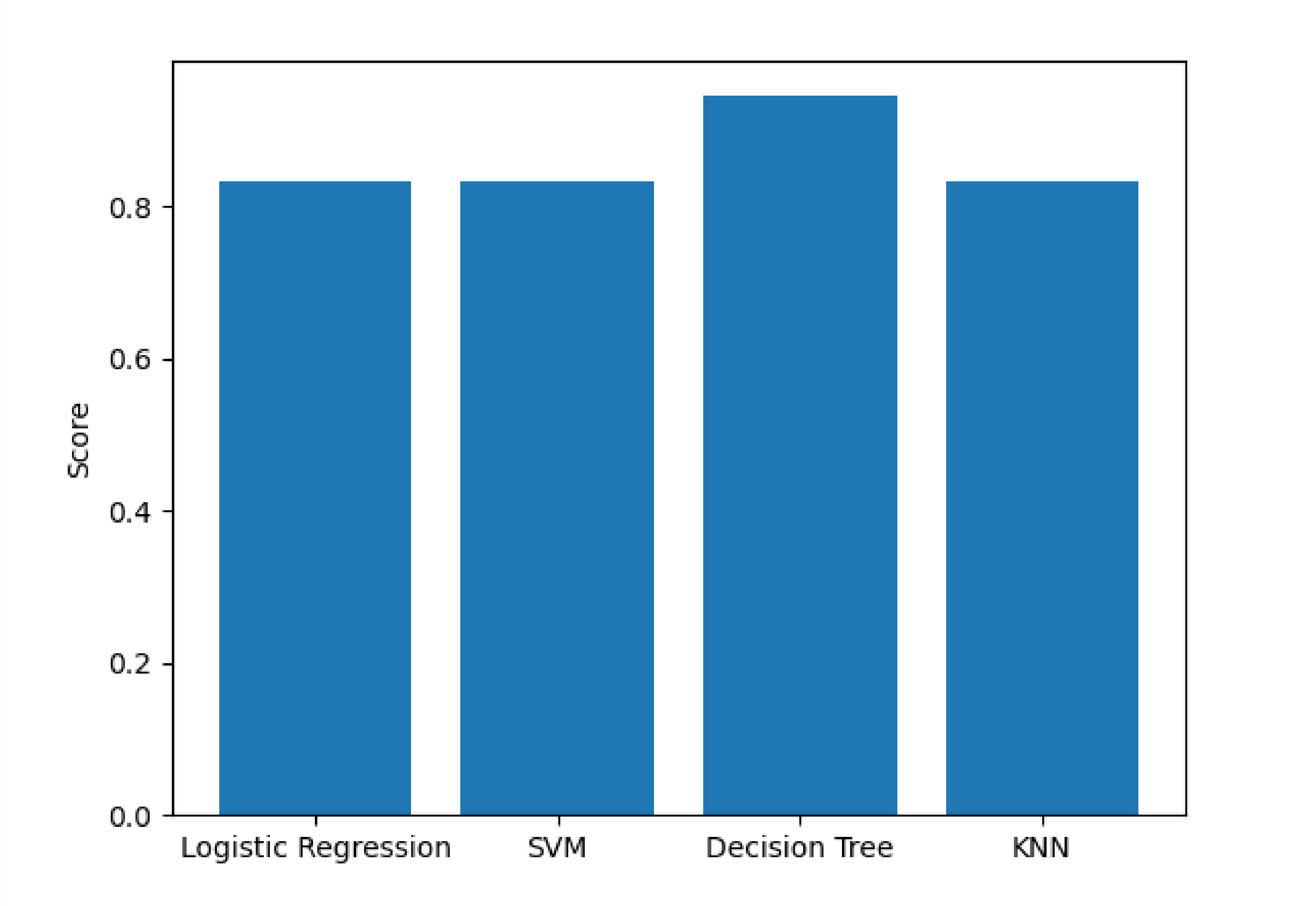
Success Rate for Different Payloads

- Both successful and failed launches were recorded across the range of tested payloads. The most successful booster version appears to be in the FT category.

Section 5

Predictive Analysis (Classification)

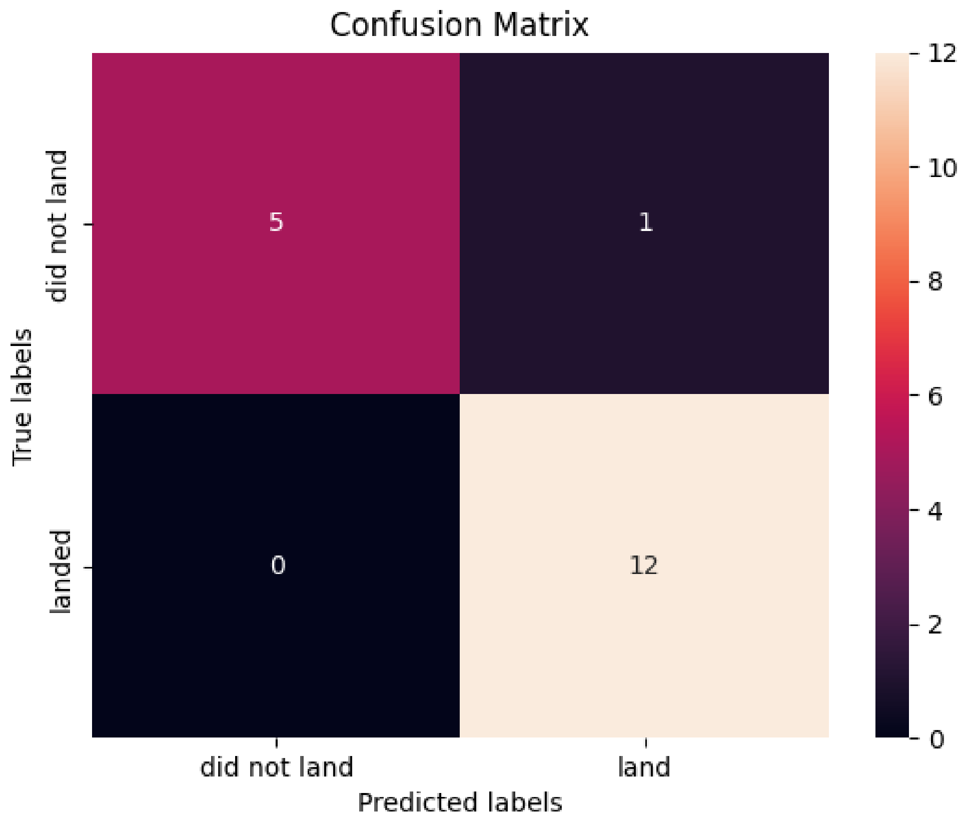
Classification Accuracy



The Logistic, KNN, and SVM models all resulted in an accuracy score of 0.833. The Decision Tree performed the best with an accuracy score of 0.9444 when used on the test data.

Confusion Matrix

```
yhat3 = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat3)
```



The confusion matrix for the decision tree model shows why it performed the best. For the 6 failed landings, only 1 record was falsely labeled as successful.

For the successful landings, all 12 records were correctly labeled.

Conclusions

- We have shown that publicly available data can be assessed to draw meaningful conclusions. Using the data from SpaceX, we have learned about the various types of boosters, timelines, launch sites, landing outcomes, and payloads for various flights.
- Using a few machine learning models, we were able to assess some parsed data related to whether a particular rocket stage would land or not.
  - Logistic, SVM, KNN, and Decision Tree classifiers were used on the data.
  - It was found that the decision tree model gave the best prediction for the landing outcome based on the data at hand.

Thank you!