Winning Space Race with Data Science

ADRIJE GUHA

July 24, 2023

Table Of Contents

Executive Summary

- SpaceY is a new commercial rocket launch provider who wants to bid against SpaceX.
- SpaceX advertises launch services starting at $62 million for missions that allow some fuel to be reserved for landing the 1st stage rocket booster, so that it can be reused.
- SpaceX public statements indicate a 1st stage Falcon 9 booster to cost upwards of $15 million to build without including R&D cost recoupment or profit margin.
- Given mission parameters such as payload mass and desired orbit, the models produced in this report were able to predict the first stage rocket booster landing successfully with an accuracy level of 83.3%.
- As a result, SpaceY will be able to make more informed bids against SpaceX by using 1st stage landing predictions as a proxy for the cost of a launch

Introduction: Project Background

- This report has been prepared as part of the Applied Data Science Capstone course.*
- In this capstone, I take the role of a data scientist working for a new rocket company called SpaceY.
- With the help of the data science findings and models in this report, SpaceY will be able to make more informed bids against SpaceX for a rocket launch.

* 10th course in the IBM Data Science Professional Certification

Introduction: Business Problems

- SpaceX advertises Falcon 9 rocket launches with a cost of 62 million dollars when the first stage of their rockets can be reused.
- The first stage is estimated to cost upwards of 15 million to build without including R&D cost recoupment or profit margin.
- Sometimes SpaceX will sacrifice the first stage due to mission parameters such as payload, orbit, and customer.
- Therefore this report aims to accurately predict the likelihood of the first stage rocket landing successfully as a proxy for the cost of a launch

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

API ~ Link to Jupyter Notebook on GitHub

- Acquired historical launch data from Open Source REST API for SpaceX
  - Requested and parsed the SpaceX launch data using the GET request
  - Filtered the dataframe to only include Falcon 9 launches
  - Replaced missing payload mass values from classified missions with mean

Web Scraping ~ Link to Jupyter Notebook on GitHub

- Acquired historical launch data from Wikipedia page 'List of Falcon 9 and Falcon Heavy Launches'
  - Requested the Falcon9 Launch Wiki page from its Wikipedia URL
  - Extracted all column/variable names from the HTML table header
  - Parsed the table and converted it into a Pandas data frame

Note: Falcon 9 launch dataset was limited to launches before December 7, 2020 per instructions.

Data Wrangling

Explored data to determine the label for training supervised models

- Calculated the number of launches on each site
- Calculated the number and occurrence of each orbit
- Calculated the number and occurrence of mission outcome per orbit type

Created a landing outcome training label from 'Outcome' column

- Training label: 'Class'
- Class = 0; first stage booster did not land successfully
- None None; not attempted
- None ASDS; unable to be attempted due to launch failure
- False ASDS; drone ship landing failed
- False Ocean; ocean landing failed
- False RTLS; ground pad landing failed
- Class = 1; first stage booster landed successfully
- True ASDS; drone ship landing succeeded
- True RTLS; ground pad landing succeeded
- True Ocean; ocean landing succeeded

Link to Jupyter Notebook on GitHub

EDA with Data Visualization

EDA with SQL

- Loaded data into an IBM DB2 instance
- Ran SQL queries to display and list information about
- Launch sites
- Payload masses
- Booster versions
- Mission outcomes
- Booster landings

Link to Jupyter Notebook on GitHub

EDA with visualization

Read the dataset into a Pandas dataframe
Used Matplotlib and Seaborn visualization libraries to plot
FlightNumber x PayloadMass †
FlightNumber x LaunchSite †
Payload x LaunchSite †
Orbit type x Success rate
FlightNumber x Orbit type †
Payload x Orbit type †
Year x Success rate
† = with Class overlayed (1st stage booster landing outcome

Link to Jupyter Notebook on GitHub

EDA with SQL

```
%%sql
SELECT "Landing_Outcome", count("Landing_Outcome") as "Total Number", count(Date) from SPACEXTBL
WHERE Date BETWEEN '04/06/2010' and '20/03/2017'
GROUP BY "Landing_Outcome"
ORDER BY "Total Numbe" DESC;
```

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

Landing_Outcome	Total Number	count(Date)
Success (ground pad)	7	7
Success (drone ship)	8	8
Success	20	20
No attempt	1	1
No attempt	9	9
Failure (parachute)	2	2
Failure (drone ship)	3	3
Failure	3	3
Controlled (ocean)	2	2

```
%%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.
```

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

Build an Interactive Map with Folium

Launch Sites Location Analysis ~ Link to Jupyter Notebook on GitHub

- Used Python interactive mapping library called Folium
- Marked all launch sites on a map
- Marked the successful/failed launches for each site on map
- Calculated the distances between a launch site to its proximities
- Railways
- Highways
- Coastlines
- Cities

Build a Dashboard with Plotly Dash

Launch Records Dashboard ~ Link to Dashboard
Code on GitHub

- Used Python interactive dashboarding library called Plotly Dash to enable stakeholders to explore and manipulate data in an interactive and real-time way
- Pie chart showing success rate
- Color coded by launch site
- Scatter chart showing payload mass vs. landing outcome
- Color coded by booster version
- With range slider for limiting payload amount
- Drop-down menu to choose between all sites and individual launch sites

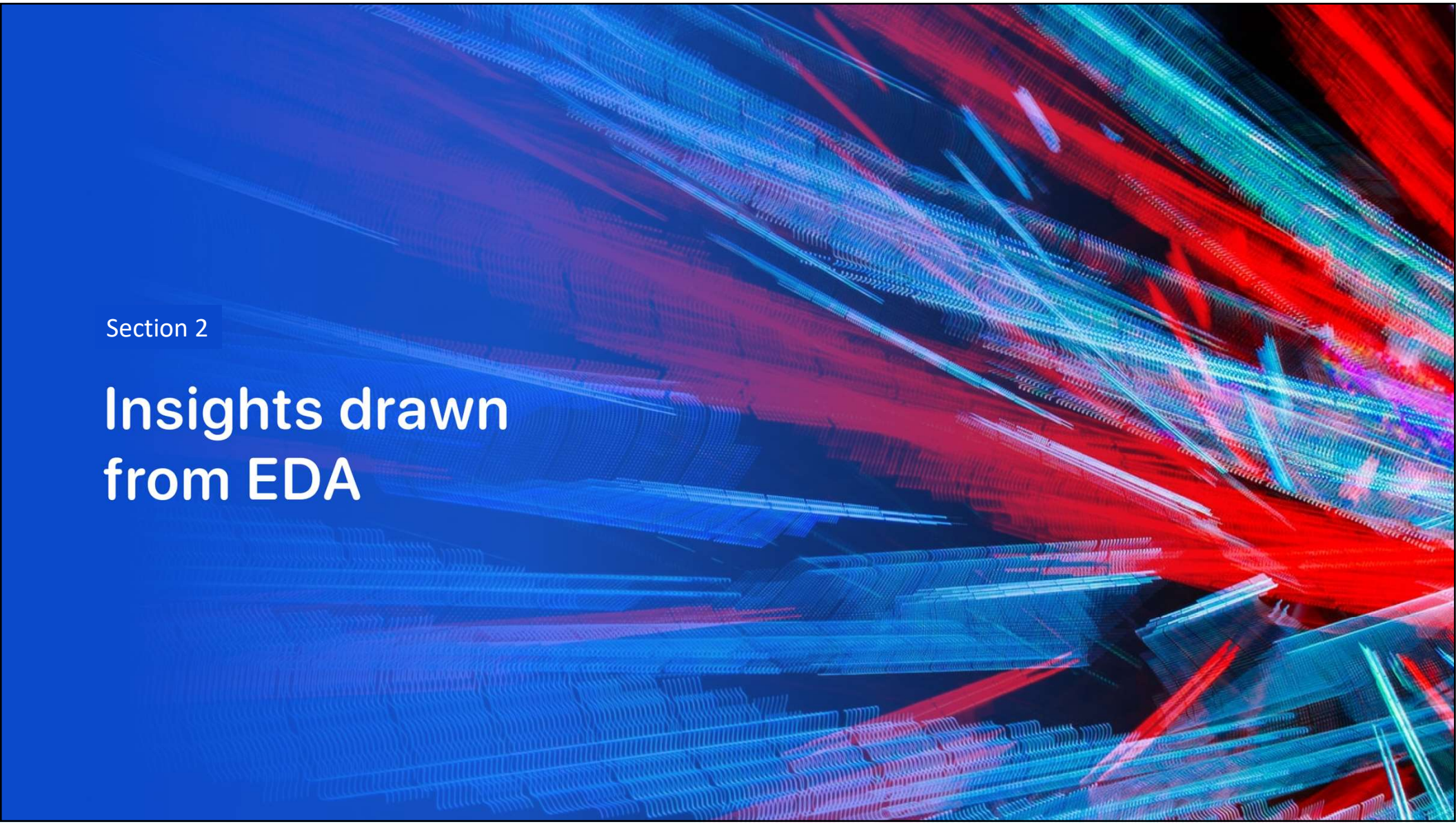
Predictive Analysis (Classification)

- Imported libraries and defined function to create confusion matrix
  - Pandas
  - Numpy
  - Matplotlib
  - Seaborn
  - Sklearn
- Loaded the dataframe created during data collection
- Created a column for our training label 'Class' created during data wrangling
- Standardized the data
- Split the data into training data and test data
- Fit the training data to various model types
  - Logistic Regression
  - Support Vector Machine
  - Decision Tree Classifier
  - K Nearest Neighbors Classifier
- Used a cross-validated grid-search over a variety of hyperparameters to select the best ones for each model
  - Enabled by Scikit-learn library function GridSearchCV
- Evaluated accuracy of each model using test data to select the best model

Link to Jupyter Notebook on GitHub

Results

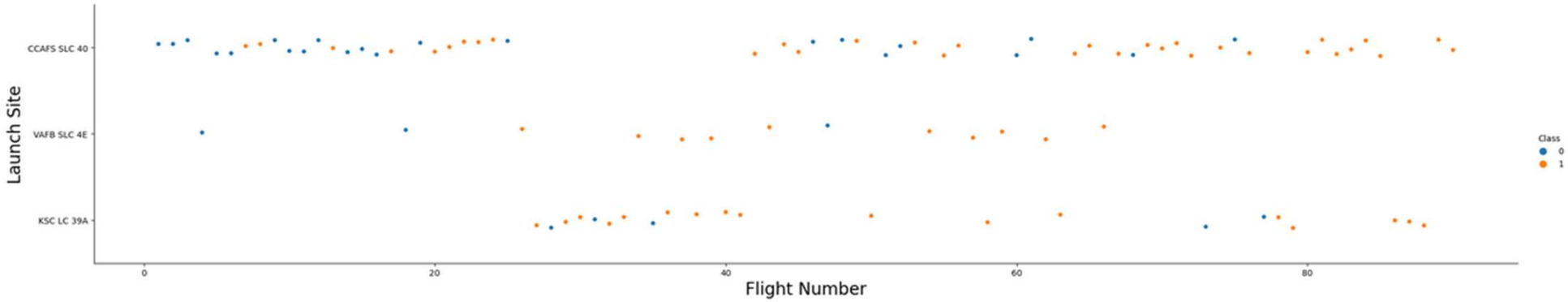
- Biggest opportunities going forward to make even more informed bids:
  - Freeze the best performing combination of model and hyperparameters and re-fit using the whole dataset instead of just the training data
    - Potentially better than using only part of the data to fit the model, but you would no longer be able to measure the accuracy of the resulting model
  - Incorporate additional launch data to the dataset and model as it becomes available
  - Subdivide the current model into two models
    - Predict if SpaceX will ATTEMPT to land the 1st stage
    - Predict if SpaceX will SUCCEED in their attempt
- Create a related model that predicts if SpaceX will launch using a previously-flown 1st stage booster
  - Would enable SpaceY to take into account when the SpaceX bid would likely include a discount


Section 2
Insights drawn
from EDA

Flight Number vs. Launch Site

In [8]:

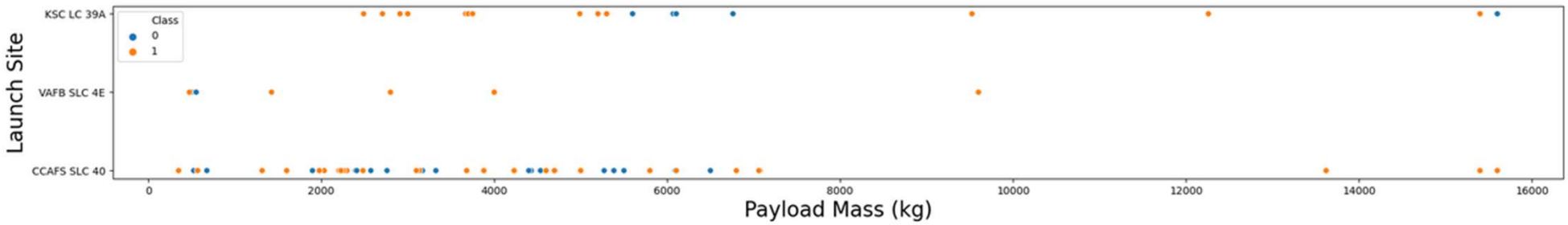
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Payload vs. Launch Site

In [22]:

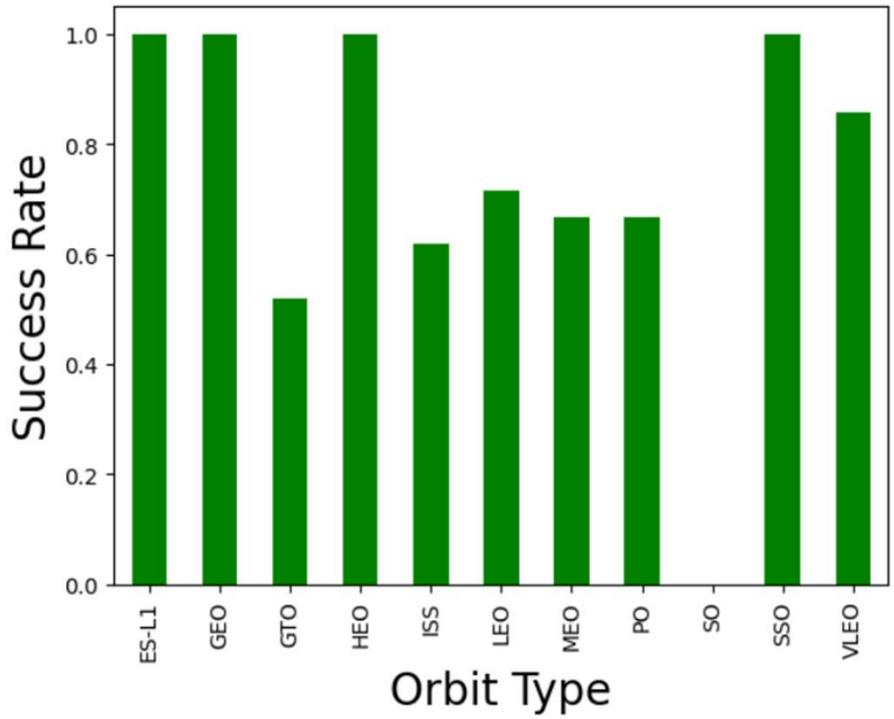
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class
plt.figure(figsize=(25, 3))
sns.scatterplot(data=df, y="LaunchSite", x="PayloadMass", hue="Class")
plt.xlabel("Payload Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Success Rate vs. Orbit Type

In [25]:

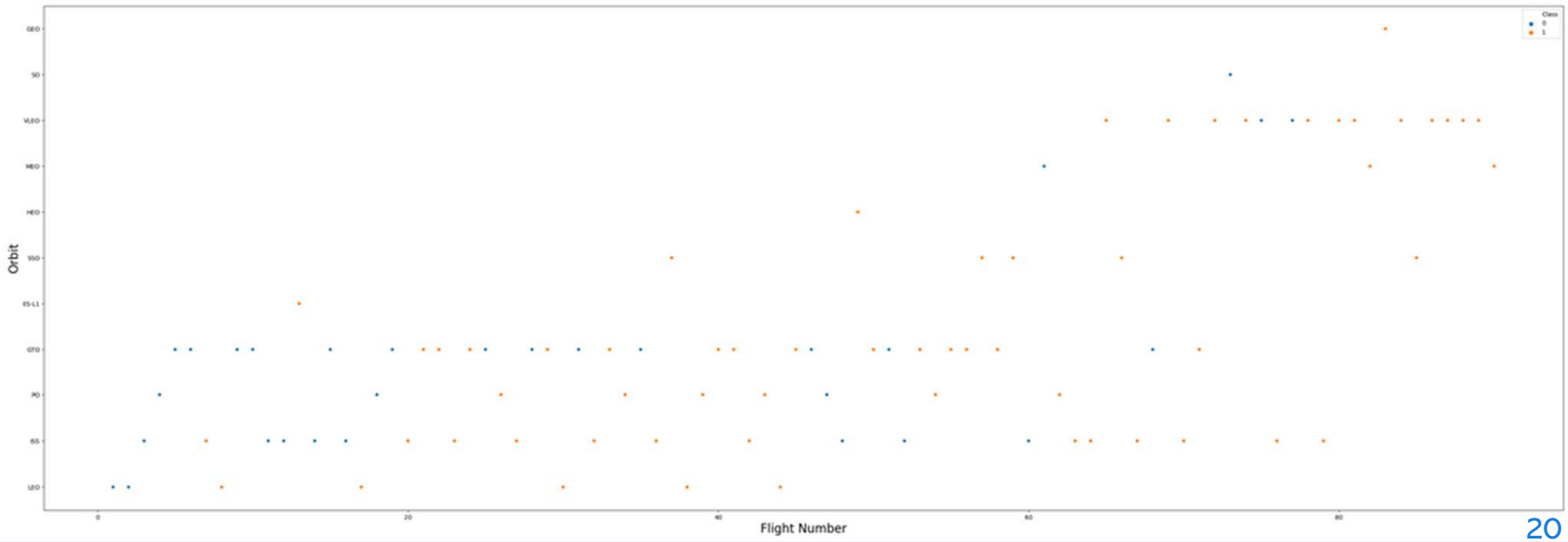
```
# HINT use groupby method on Orbit column and get the mean of Class column
df.groupby("Orbit").mean()['Class'].plot(kind='bar', color='g')
plt.xlabel("Orbit Type",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```



Flight Number vs. Orbit Type

In [33]:

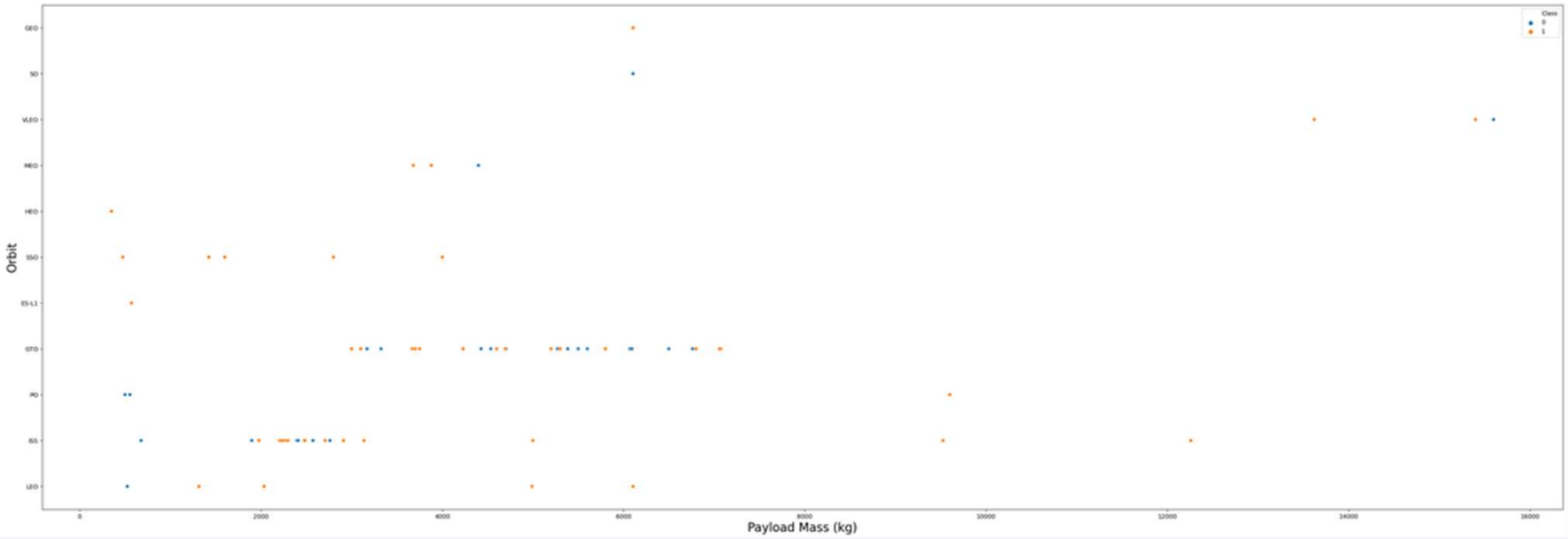
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
plt.figure(figsize=(45, 15))
sns.scatterplot(data=df, y="Orbit", x="FlightNumber", hue="Class")
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Payload vs. Orbit Type

In [34]:

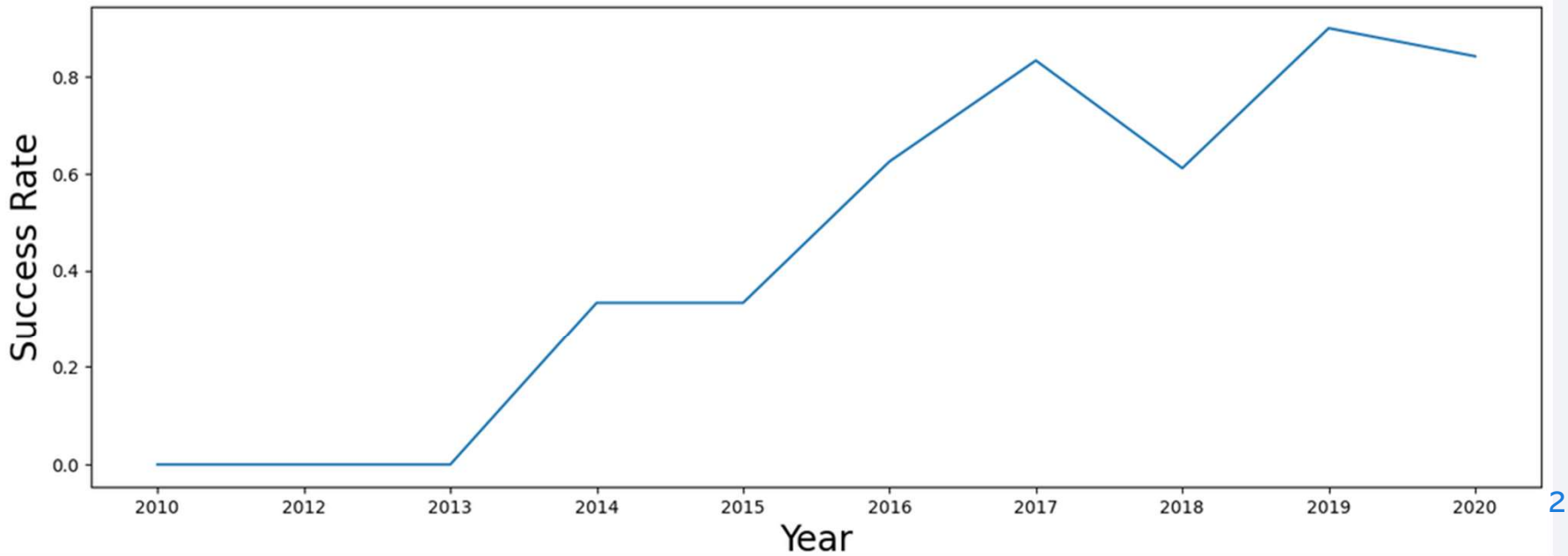
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
plt.figure(figsize=(45, 15))
sns.scatterplot(data=df, y="Orbit", x="PayloadMass", hue="Class")
plt.xlabel("Payload Mass (kg)",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Launch Success Yearly Trend

In [43]:

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
plt.figure(figsize=(15, 5))
sns.lineplot(data=df, x=df['Date'].unique(), y=df.groupby(['Date'])['Class'].mean())
plt.xlabel("Year",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```



All Launch Site Names

In [23]:
```
%%sql
SELECT DISTINCT "Launch_Site" from SPACEXTBL;
```

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

Out[23]:

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

There are four different launch sites through out USA and all the locations are coastal regions.

Launch Site Names Begin with 'CCA'

In [29]:

```
%%sql
SELECT * from SPACEXTBL
WHERE "Launch_Site" LIKE "CCA%"
LIMIT 5;
```

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

Out[29]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (paracl
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 (paracl
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 atte
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No atte
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No atte

There are 60 records where launch sites name begin with the string 'CCA'.

Total Payload Mass

In [31]:
```
%%sql
SELECT SUM("PAYLOAD_MASS__KG_") from SPACEXTBL
WHERE "Customer"="NASA (CRS)";
```

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

Out[31]:

SUM("PAYLOAD_MASS__KG_")
45596.0

The total payload mass carried by boosters launched by NASA (CRS) is 45,596 Kgs

Average Payload Mass by F9 v1.1

In [32]:
```
%%sql
SELECT AVG("PAYLOAD_MASS__KG_") from SPACEXTBL
WHERE "Booster_Version"="F9 v1.1";
```

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

Out[32]:

AVG("PAYLOAD_MASS__KG_")
2928.4

The average payload mass carried by booster version F9 v1.1 is 2,928.4 Kgs

First Successful Ground Landing Date

In [103…

```
%%sql
SELECT min(Date) from SPACEXTBL
WHERE (
    SELECT min(substr(Date,7,4)) from SPACEXTBL
    WHERE ("Landing_Outcome" = 'Success (ground pad)'))=substr(Date,7,4)
and "Landing_Outcome" = 'Success (ground pad)';
```

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

Out[103…

min(Date)
22/12/2015

22/12/2015 is the date when the first succesful landing outcome in ground pad was achieved.

Successful Drone Ship Landing with Payload between 4000 and 6000

In [82]:
```
%%sql
SELECT "Booster_Version"from SPACEXTBL
WHERE "Landing_Outcome"='Success (drone ship)' and 4000<"PAYLOAD_MASS__KG_" and "PAYLOAD_MASS__KG_"<6000;
```

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

Out[82]:

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

Five boosters has achieved success in drone ship and have payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

In [84]:

```
%%sql
SELECT "Mission_Outcome", COUNT("Mission_Outcome") from SPACEXTBL
GROUP BY trim("Mission_Outcome");
```

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

Out[84]:

Mission_Outcome	COUNT("Mission_Outcome")
None	0
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

In [88]:
```
%%sql
SELECT DISTINCT("booster_version") from SPACEXTBL
WHERE (
    SELECT max("PAYLOAD_MASS__KG_") from SPACEXTBL)="PAYLOAD_MASS__KG_";
```

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

Out[88]:

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

In [102…

```
%%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.
```

Out[102…

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

In [114...

```
%%sql
SELECT "Landing_Outcome", count("Landing_Outcome") as "Total Number", count(Date) from SPACEXTBL
WHERE Date BETWEEN '04/06/2010' and '20/03/2017'
GROUP BY "Landing_Outcome"
ORDER BY "Total Numbe" DESC;
```

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

Out[114...

Landing_Outcome	Total Number	count(Date)
Success (ground pad)	7	7
Success (drone ship)	8	8
Success	20	20
No attempt	1	1
No attempt	9	9
Failure (parachute)	2	2
Failure (drone ship)	3	3
Failure	3	3
Controlled (ocean)	2	2


Section 3
Launch Sites
Proximities Analysis

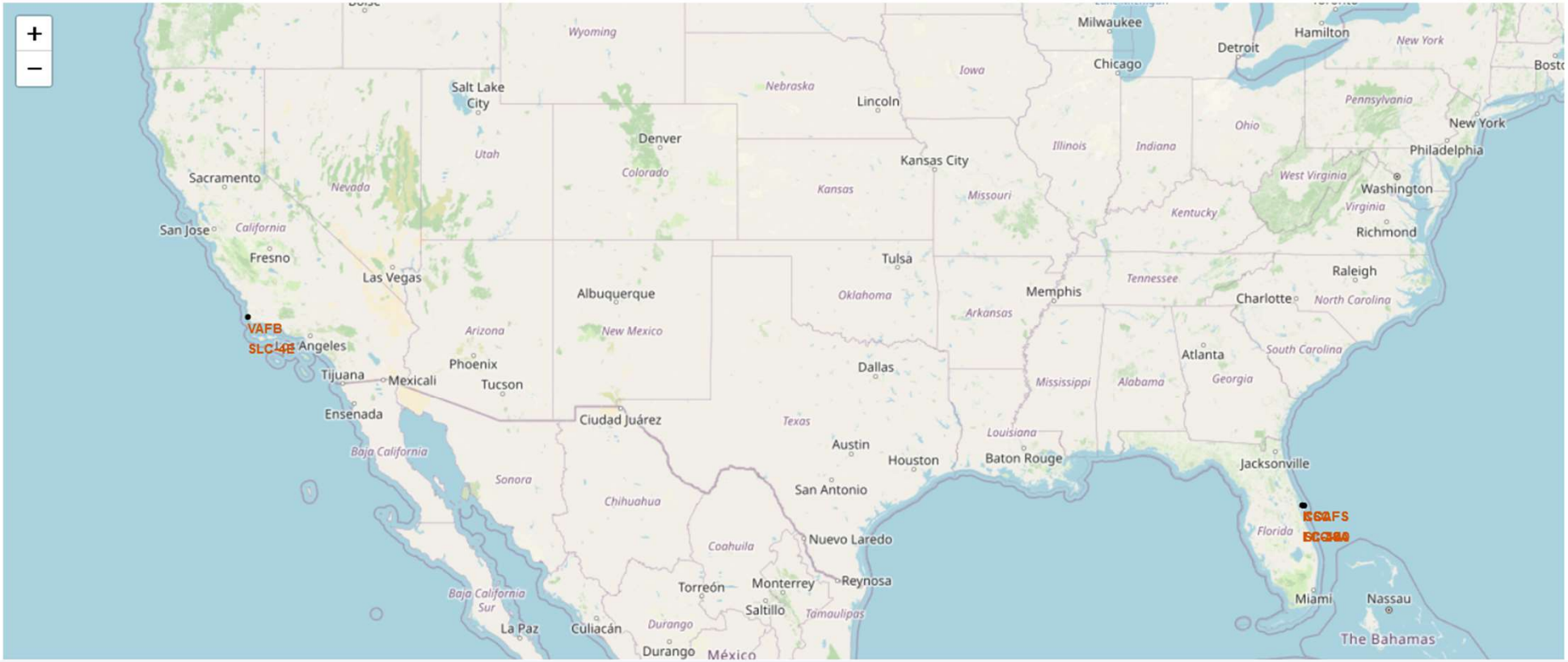
lab_jupyter_launch_site_location.jupyterlite.ipynb

File Edit View Run Kernel Git Tabs Settings Help

Markdown git Run as Pipeline Python

[13]:
```
site_map
```

[13]:



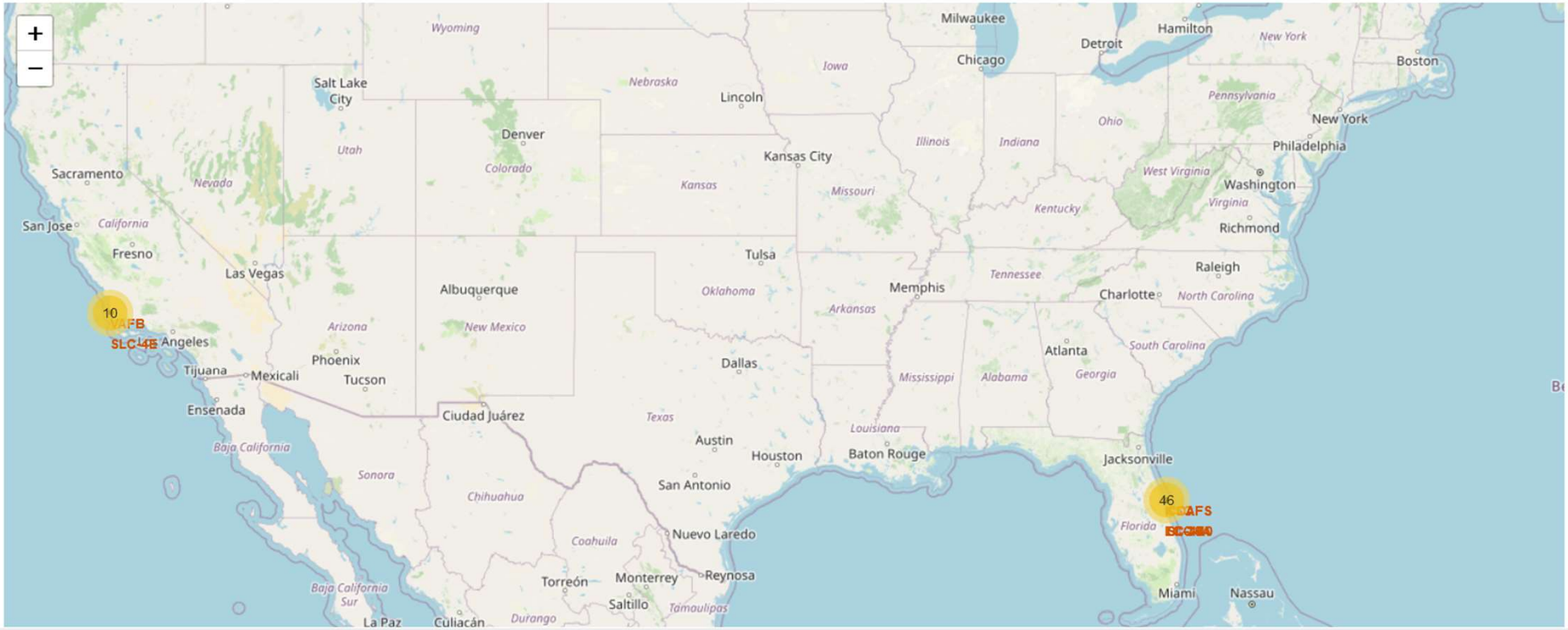
lab_jupyter_launch_site_location.jupyterlite.ipynb

File Edit View Run Kernel Git Tabs Settings Help

Markdown git Run as Pipeline Python

```
site_map
```

[18]:

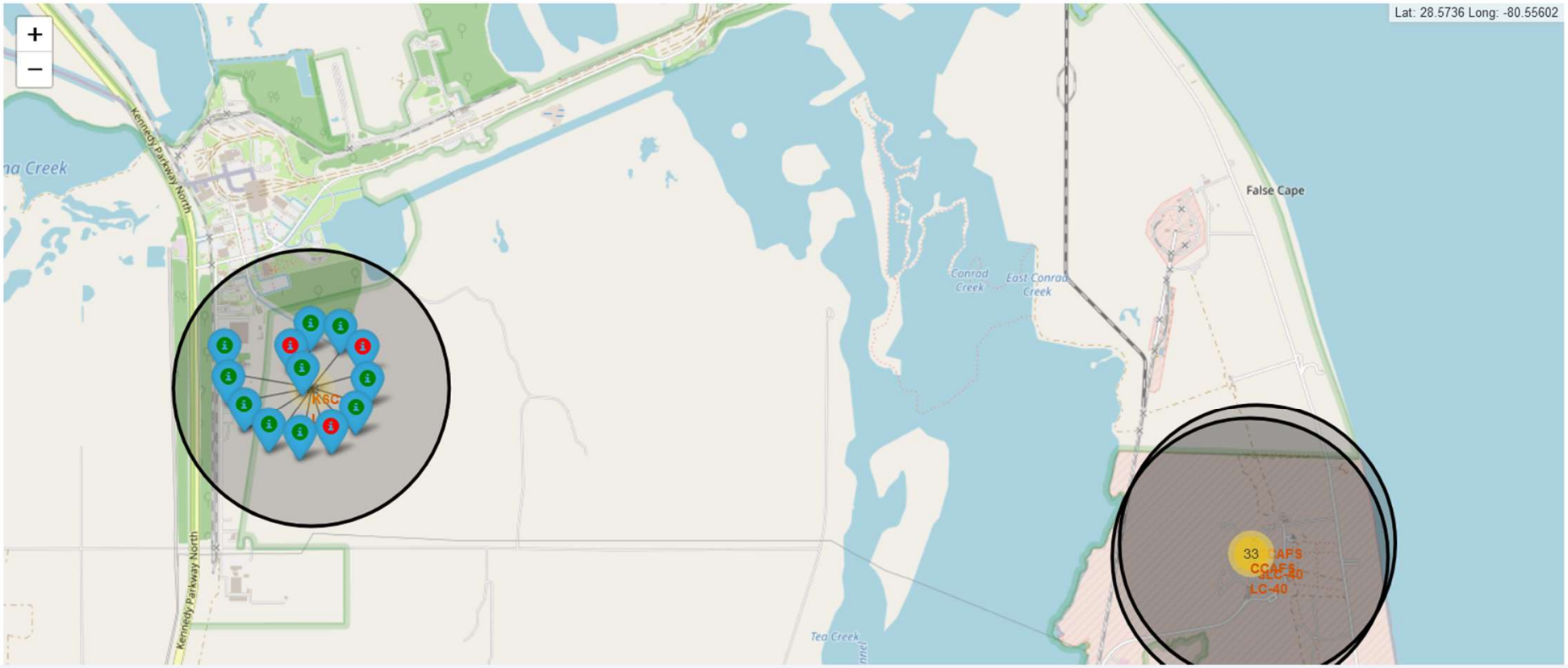


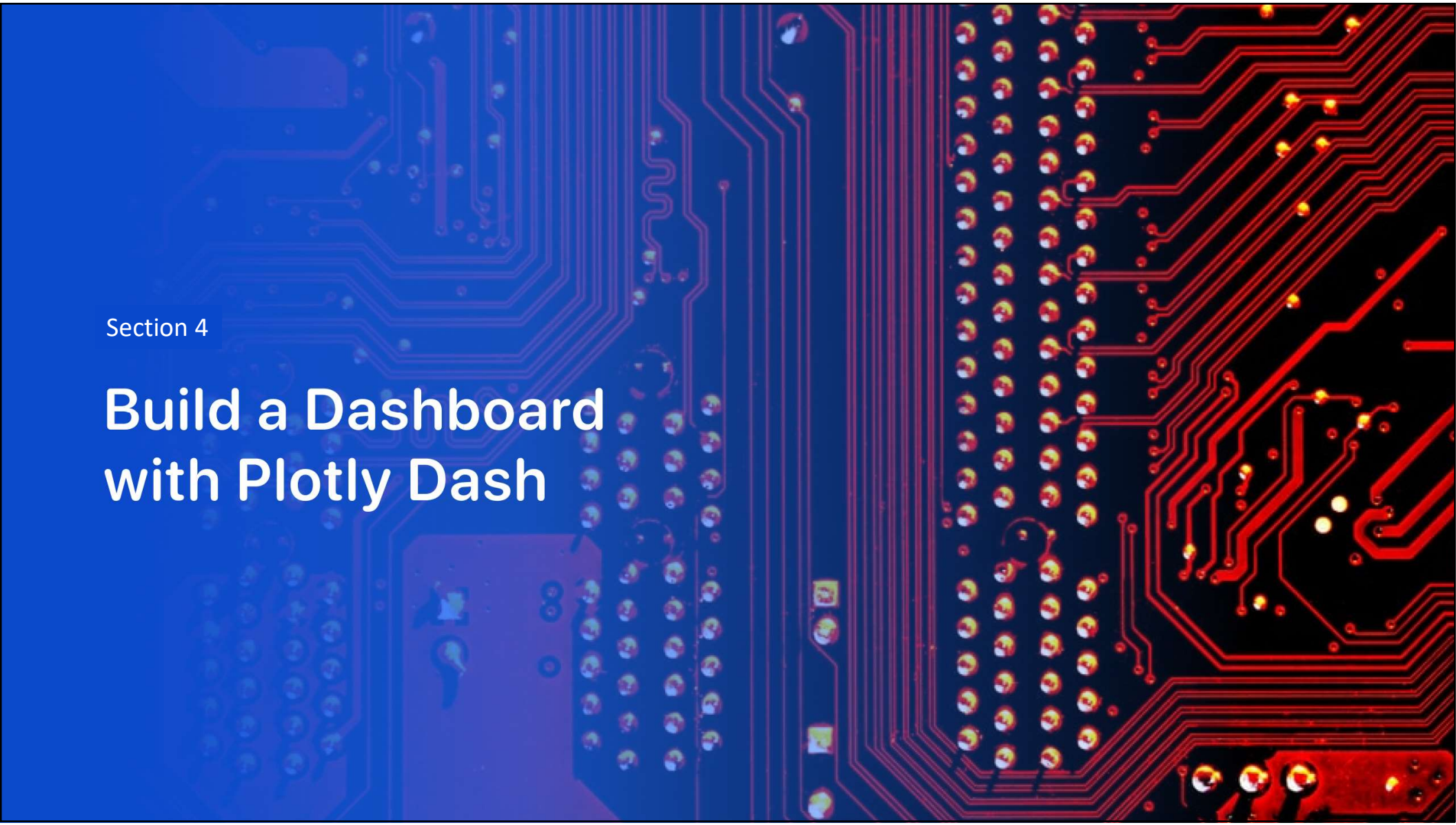
lab_jupyter_launch_site_location.jupyterlite.ipynb

File Edit View Run Kernel Git Tabs Settings Help

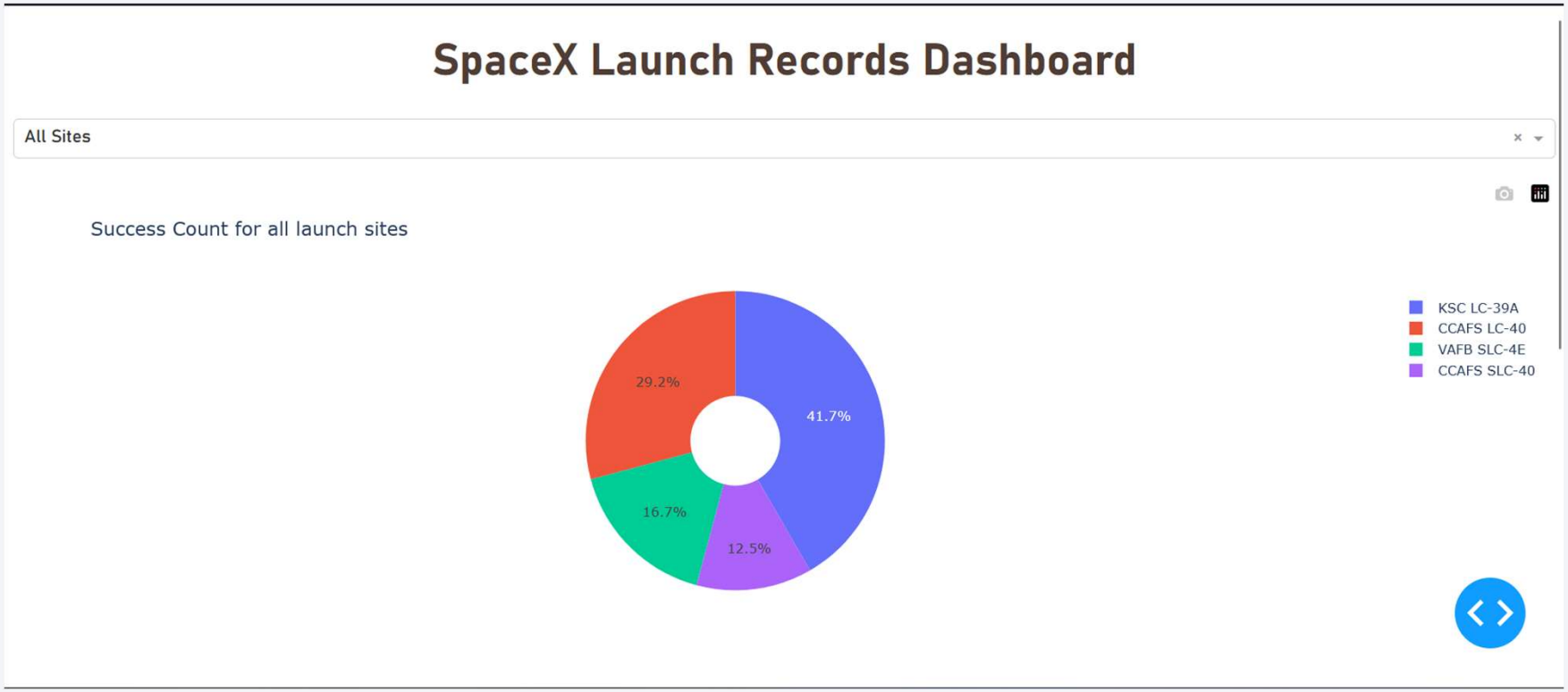
Markdown git Run as Pipeline Python

[19]:

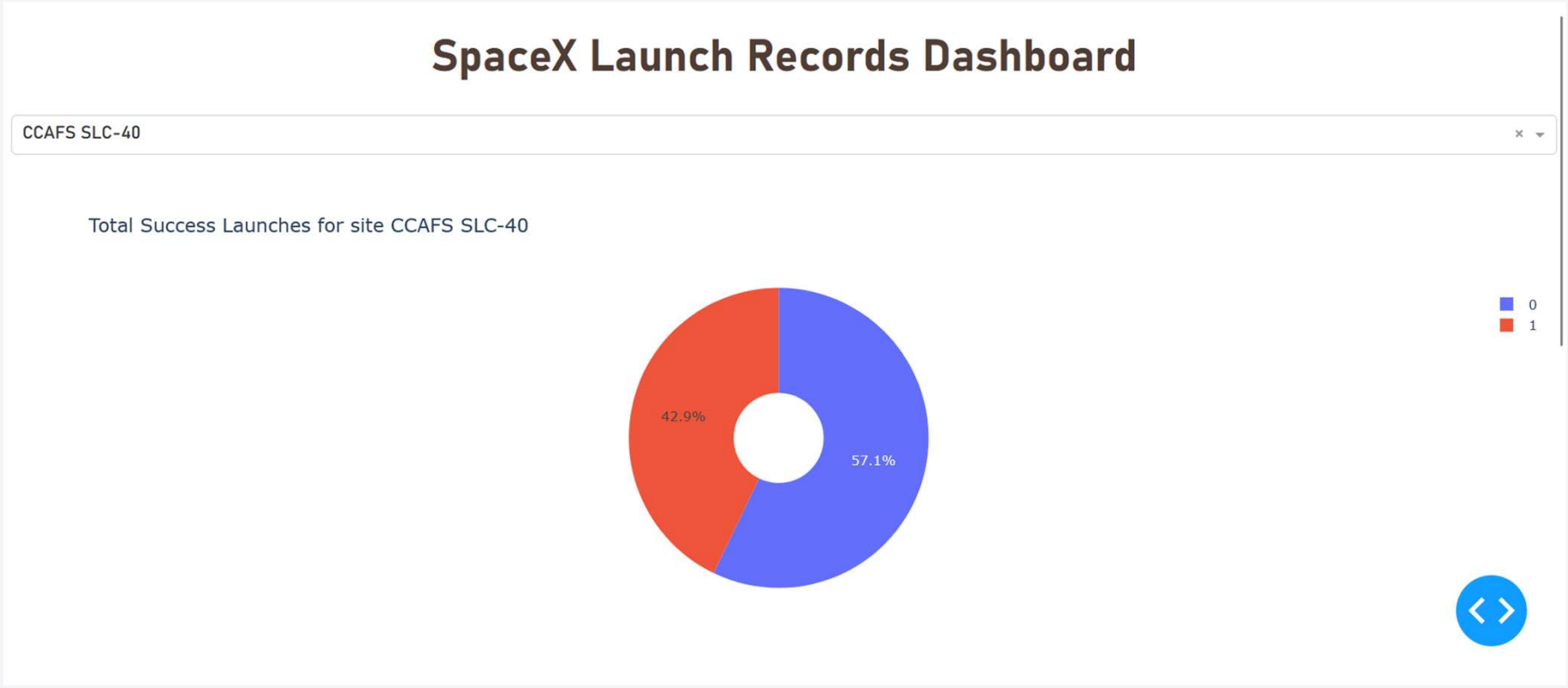



Section 4
Build a Dashboard with Plotly Dash

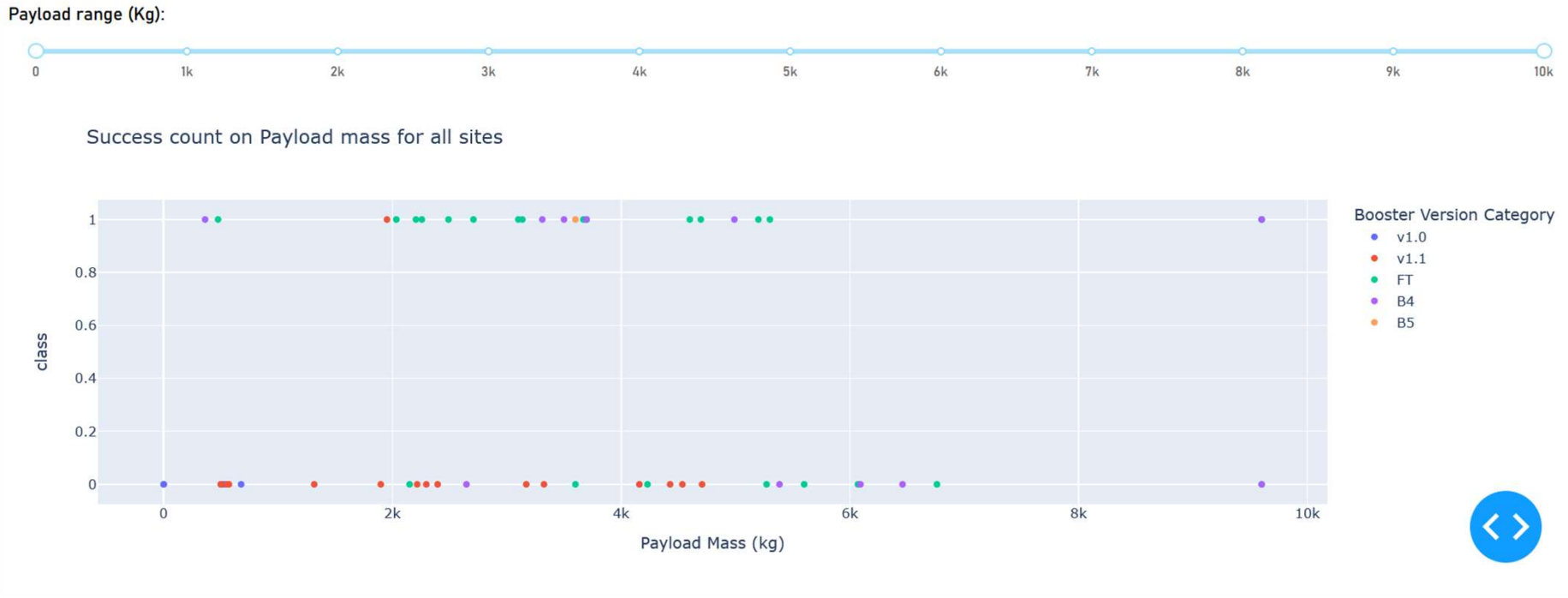
Dashboard: Pie Chart for all sites count of success launch



Dashboard: Pie Chart for CCAFS SLC-40 site launch count



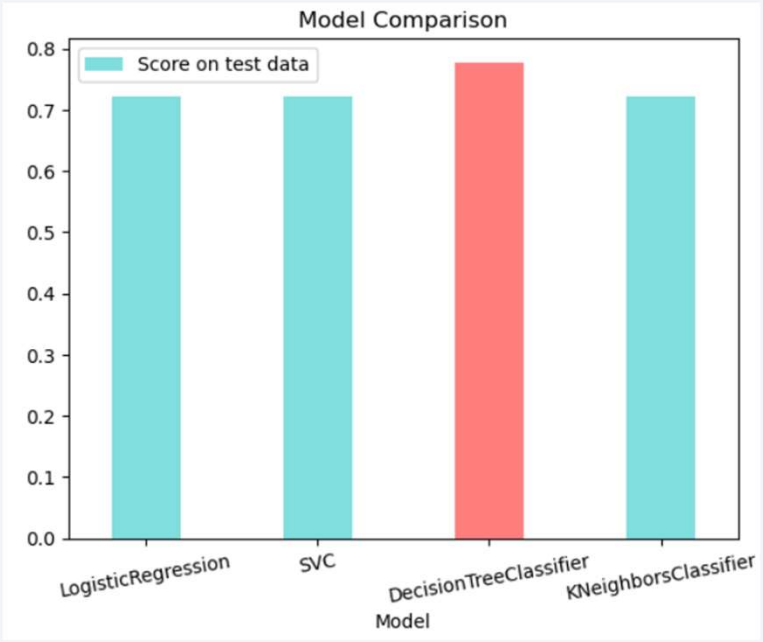
Dashboard: Scatter Pot of Payload vs Mass for all launch sites

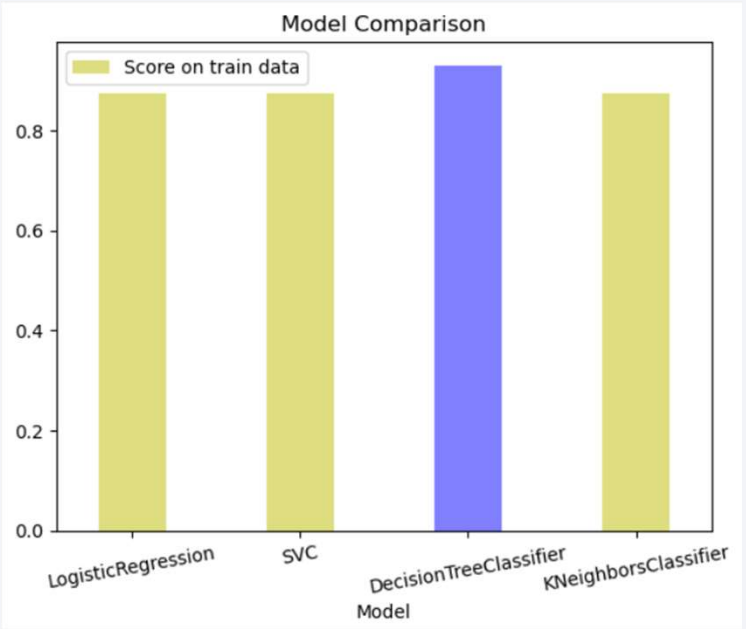


Section 5

Predictive Analysis (Classification)

Classification Accuracy

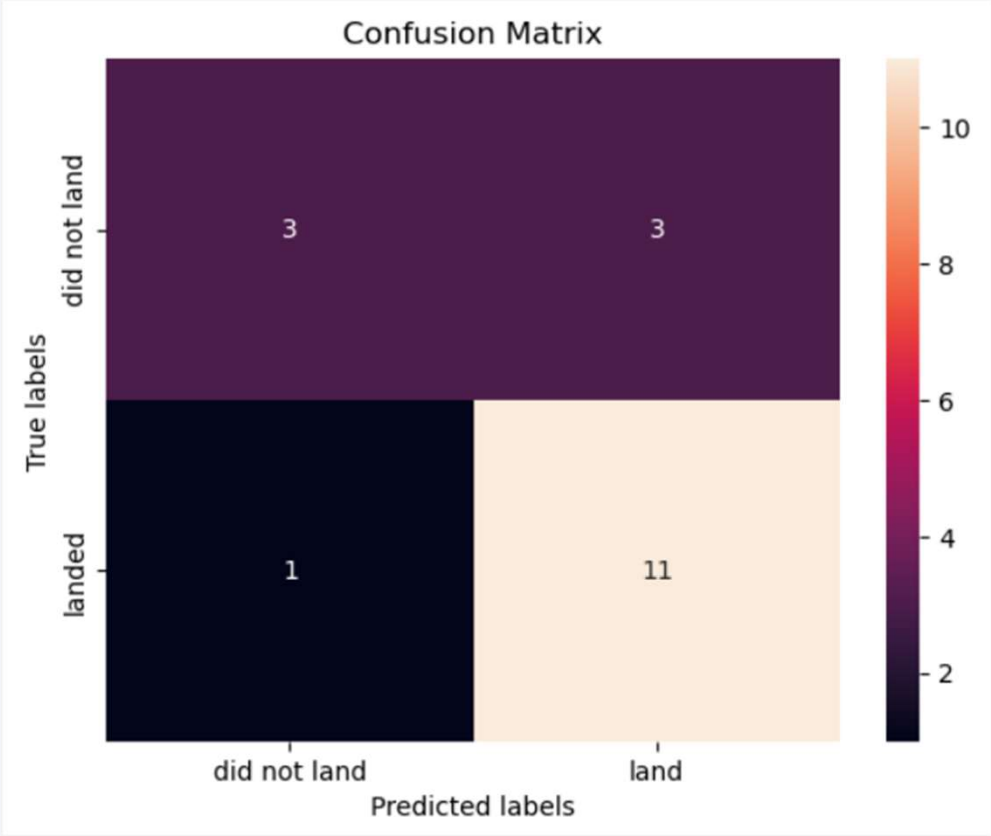




	Model	Score on test data	Score on train data
0	LogisticRegression	0.722222	0.875000
1	SVC	0.722222	0.875000
2	DecisionTreeClassifier	0.777778	0.930357
3	KNeighborsClassifier	0.722222	0.875000

We can see that the Decision Tree Classifier performs the best with an accuracy score of 77% and 93% approx. on test and train data respectively.

Confusion Matrix



Our model predicts essentially well with true-negative values, i.e., our model correctly predicted the landing, while the prediction for not landing is quite not accurate.

Conclusions

- Using the models from this report, SpaceY can predict when SpaceX will successfully land the 1st stage booster with 83.3% accuracy
- SpaceX public statements indicate the 1st stage booster costs upwards of $15 million to build
- This will enable SpaceY to make more informed bids against SpaceX, since they will have a good idea when to expect the SpaceX bid to include the cost of a sacrificed 1st stage booster
- With a list price of $62 million per launch, sacrificing the $15+ million 1st stage, would put the SpaceX bid at upwards of $77 million
- Biggest opportunities going forward to make even more informed bids:
  - Freeze the best performing combination of model and hyperparameters and re-fit using the whole dataset instead of just the training data
    - Potentially better than using only part of the data to fit the model, but you would no longer be able to measure the accuracy of the resulting model
  - Incorporate additional launch data to the dataset and model as it becomes available
  - Subdivide the current model into two models
    - Predict if SpaceX will ATTEMPT to land the 1st stage
    - Predict if SpaceX will SUCCEED in their attempt
- Create a related model that predicts if SpaceX will launch using a previously-flown 1st stage booster
  - Would enable SpaceY to take into account when the SpaceX bid would likely include a discount

Appendix & Acknowledgments

- Thank you to Joseph Santarcangelo and Yan Lao at IBM for creating the course and materials
- The code and necessary files used for this project can be found in the following GitHub repo: [Link to repo]
- •References
  - https://aviationweek.com/defense-space/space/podcast-interview-spacexs-elon-musk
    - Interview with Elon Musk where he discloses the 1st stage booster to cost upwards of $15 million
  - https://datascience.stackexchange.com/a/33050
    - Explanation of why you would rebuild your model using the full dataset
  - https://www.spacex.com/vehicles/falcon-9/
    - Source of SpaceX's advertised $62 million launch price

Thank you!