Winning Space Race with Data Science

Xiuwen Tu
3/28/2020

Outline

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

Executive Summary

- Data collection, wrangling, analysis, analytics, and machine learning models are all carried out with Python, including the following modules:
  - Python3 standard library, Pandas, Numpy, Requests, BeautifulSoup, matplotlib, seaborn, folium, plotly, dash, scikit-learn.
- Summary of results: The test prediction accuracy for the landing outcome is 83.3% for 3 machine learning models:
  - Logistic Regression,
  - SVM, and
  - KNN.
- The accuracy of Decision Tree is lower at 66.7%.

Introduction

- SpaceX is able to land first-stage rockets after launch, resulting in great cost savings per launch, from about US$165M to about US$62M.
- We want to collect data and build machine-learning models to predict the landing outcome of the Falcon 9 first-stage.
- Accurate models of this kind can help quantify the launch cost, which can be crucial for a hypothetical alternate startup trying to compete with SpaceX.

Section 1

Methodology

Methodology - Summary

- Data collection is done with two methods:
  - i) calling the SpaceX REST API, and
  - ii) web-scraping of a Wikipedia page.
- Data wrangling is carried out with
  - Pandas: df.dtypes, df.isnull(), df.count(), df.col_name.value_counts(), df.col_name.to_list(), df.groupby(),
  - Python list comprehension for creating a ‘Class’ label for landing outcome, 1 or 0.
- Exploratory data analysis (EDA) is performed with
  - Visualization of the data with catplot() of seaborn, or bar chart and line chart of matplotlib.
  - SQL queries of a database table after manually importing the CSV file into IBM Db2.
- Spacial analytics is done using Folium to map the launch sites and outcomes.
- Interactive analytics is completed by building a dashboard with Plotly Dash.
- Predictive analysis is done using scikitlearn’s train/test split, cross validation, confusion matrix for 4 classification models:
  - Logistic Regression, SVM, KNN, and Decision Tree.

Data Collection

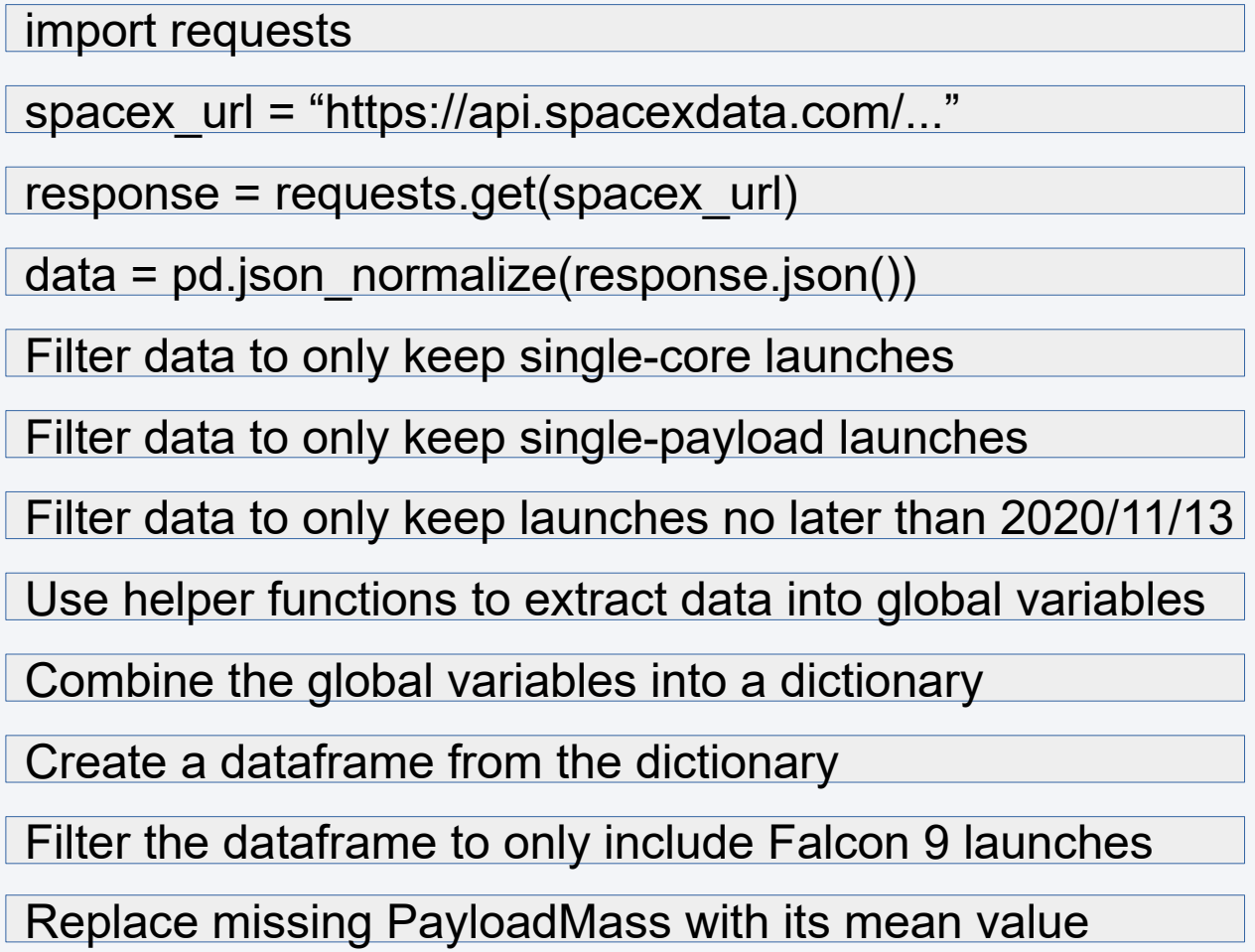
- Datasets are collected with two methods:
  - Calling the SpaceX REST API, and
  - Web scraping of a Wikipedia page.

- See the next 2 slides for more details.

Data Collection – SpaceX API

- Data collection with SpaceX REST API is done with the Requests module.
- See the right for a flowchart.
- GitHub URL of the completed SpaceX API calls notebook:

https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module11_jupyter-labs-spacex-data-collection-api-mine.ipynb



Data Collection – Web Scraping

- Web-scraping is done with Requests and Beautiful Soup modules.
- See the right for a flowchart.

- GitHub URL of the completed web scraping notebook:

https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module12-jupyter-labs-spacex-data-collection-webscraping-mine.ipynb

Import 2 key packages together with a few others

```
import requests
from bs4 import BeautifulSoup
```

Request the Falcon9 Launch Wiki Page from Its URL

```
static_url = "https://en.wikipedia.org/...”
```

```
response = requests.get(static_url)
```

Create a BeautifulSoup object to parse the HTML

```
soup = BeautifulSoup(response.text)
```

Find the HTML launch tables and extract column_names

```
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]
```

Parse the launch HTML tables and build up launch_dict

Create a DataFrame from launch_dict

Data Wrangling

- Data wrangling is done with Pandas and NumPy.
- The values of the landing outcome label ‘Class’ are worked out with Python list comprehension.
- See the right for a flowchart.
- GitHub URL of the completed data wrangling related notebook:

https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module13_jupyter-labs-spacex-data-wrangling-mine.ipynb

Import pandas and numpy with conventional aliases

Load the dataset and check its shapes, dtypes, null values

```
df = pd.read_csv(“https://cf-courses-data...”)
```

```
df.shape
df.dtypes
df.isnull().sum()
```

Calculate the number of launches by 3 attributes

```
df.LaunchSite.value_counts()
df.Orbit.value_counts()
df.Outcome.value_counts()
```

Create a landing class label from Outcome column

```
landing_class = [0 if outcome in bad_outcomes
    else 1 for outcome in outcome_list]
df['Class'] = landing_class
```

EDA with SQL

- First, a CSV data file is manually loaded into a table at IBM Db2.
- Then, the following 10 SQL queries are run to analyze the data as part of the EDA.
- GitHub URL of the completed EDA with SQL notebook:

https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module21_jupyter-labs_spacex-eda_with_sql.ipynb

```
%sql SELECT UNIQUE launch_site FROM eda_with_sql
```

```
%sql SELECT * FROM eda_with_sql
WHERE launch_site LIKE 'CCA%' LIMIT 5
```

```
%sql SELECT SUM(payload_mass__kg_) FROM eda_with_sql
WHERE UCASE(customer)='NASA (CRS)'
```

```
%sql SELECT AVG(payload_mass__kg_) FROM eda_with_sql
WHERE booster_version like 'F9 v1.1%'
```

```
%sql SELECT MIN(date) FROM eda_with_sql
WHERE landing__outcome='Success (ground pad)'
```

```
%sql SELECT booster_version,landing__outcome,payload_mass__kg_ FROM eda_with_sql
WHERE landing__outcome='Success (drone ship)' AND payload_mass__kg_ BETWEEN 4000 AND 6000
```

```
%sql SELECT COUNT(mission_outcome) FROM eda_with_sql
WHERE UCASE(mission_outcome) LIKE 'SUCCESS%'
```

```
%sql SELECT booster_version,payload_mass__kg_ FROM eda_with_sql
WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM eda_with_sql)
```

```
%sql SELECT date,booster_version,launch_site,landing__outcome FROM eda_with_sql
WHERE landing__outcome='Failure (drone ship)' AND YEAR(date)=2015
```

```
%sql SELECT landing__outcome,COUNT(landing__outcome) FROM eda_with_sql
WHERE date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY landing__outcome ORDER BY COUNT(landing__outcome) DESC
```

EDA with Data Visualization & Feature Engineering

- First, the csv data file is loaded into a pandas DataFrame.
- Next, seaborn.catplot() is used to plot the relationship between variable pairs, e.g.
  - Flight Number and Launch Site or Orbit Type,
  - Payload Mass and Launch Site or Orbit Type.
- The success rate of each orbit type is visualized with a bar chart.
- The yearly trend of landing success rate is visualized with a line chart.
- Feature engineering is carried out by
  - i) selecting the useful features for building machine-learning models,
  - ii) one-hot encoding of categorical columns, and
  - iii) casting all numeric to float64 type.
- GitHub URL of the completed EDA with data visualization notebook:

https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module22_jupyter-labs-spacex-eda-dataviz-pandas-mine.ipynb

Build an Interactive Map with Folium

- First, a map of the lower 48 states of the USA is generated with Folium.
- Next, the following map objects are added to the map:
  - A circle and marker for each of the 4 launch sites. In general, they’re close to the equator.
  - A cluster of markers, each showing the launch site and status of one row or launch;
  - A MousePosition object for getting the coordinates of any point on the map by the mouse;
  - A text and line showing the distance of VAFB SLC-4E to the nearby coast, ~1.35 km. The railway along the coast is a little closer;
  - A text and line showing the distance of VAFB SLC-4E to the nearby parking lot, ~0.52 km.
- GitHub URL of the completed interactive map with Folium:

https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module31_jupyter-labs_spacex-folium-map-launch_site_location-mine.ipynb

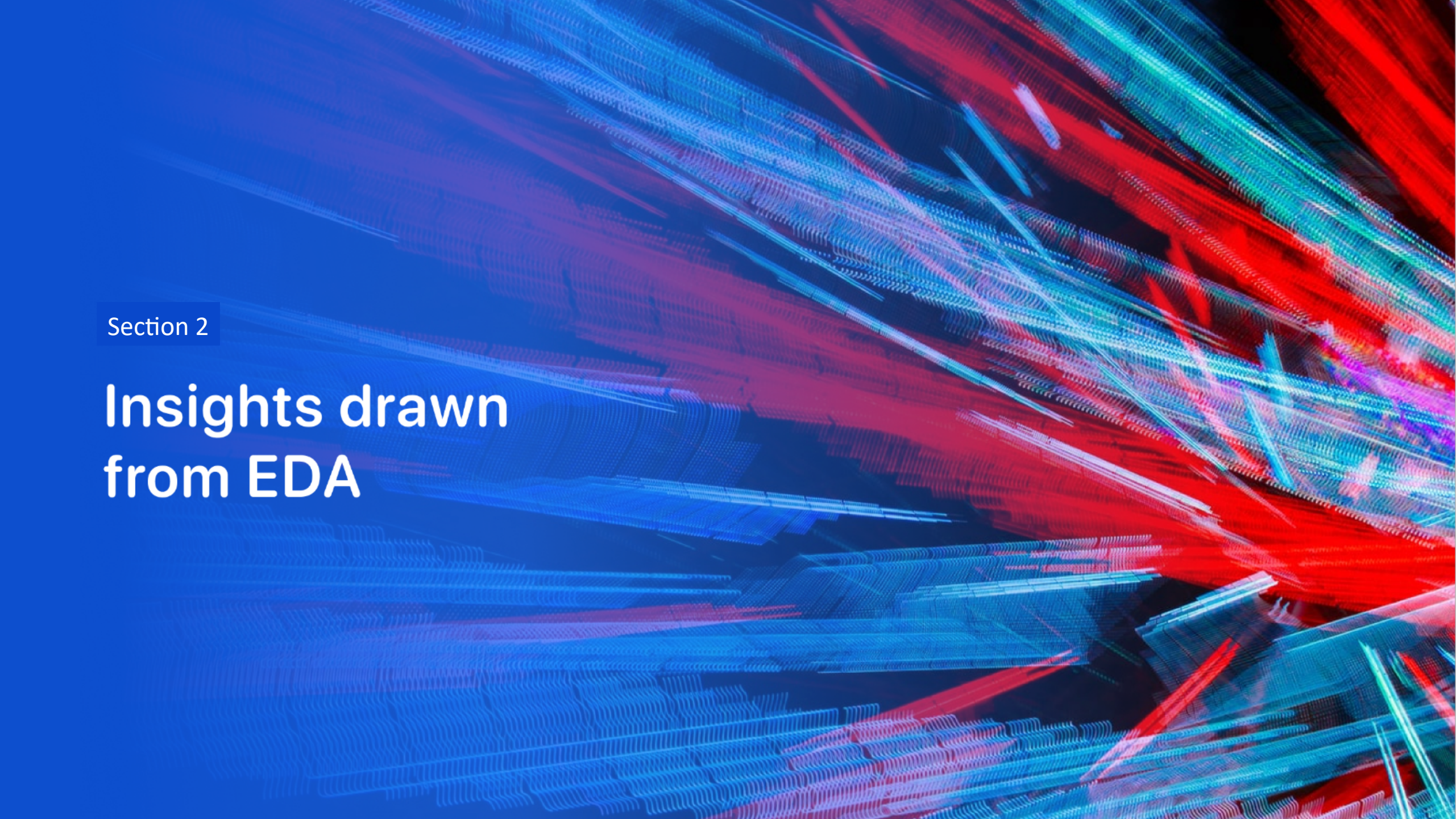
Build a Dashboard with Plotly Dash

- A dashboard is built with plotly and dash modules, with 2 inputs or interactions:
  - A dropdown list to select launch site, and
  - A slide to select payload mass range.
- The dashboard also have these 2 charts:
  - A pie chart:
    - When all launch sites are selected, the pie chart shows each site’ successful landings as a percentage of all successful landings.
    - When a specific launch site is selected, the pie chart show this site’s successful landings as a percentage of this site’s total landing attempts.
  - A scatter chart of landing outcome or class vs payload mass:
    - The launch site(s) is selected by the dropdown list, and the payload mass range by the slider.
- GitHub URL of the completed Plotly Dash lab:
- https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module32_plotly-dash-mine.py

Predictive Analysis (Classification)

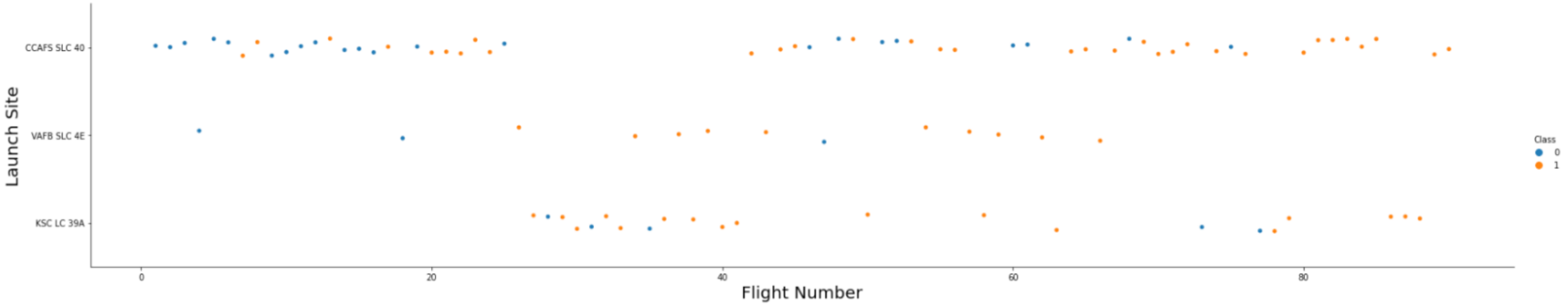
- First, matrices X and Y are prepared:
  - X is read from a CSV file into a DataFrame, and then transformed with sklearn.preprocessing.StandardScaler().
  - Y is read from another CSV file into a DataFrame, and then column ‘Class’ is cast into a NumPy ndarray.
- Second, sklearn.model_selection.train_test_split() is used to split both X and Y into the training X & Y set and the testing X & Y set.
- Next, 4 machine-learning models are built and evaluated with sklearn:
  - 4 ML Models: Logistic Regression, SVM, KNN, and Decision Tree.
  - Each model is built and evaluated with this process flow:
    1. Identify the parameters and initialize the model;
    2. Use sklearn.model_selection.GridSearchCV() to fit on the training X & Y set and obtain the best parameters and training score;
    3. Use the trained model’s score() function and the testing X & Y set to get the test accuracy;
    4. Obtain y_hat by calling the trained model’s predict(X_test) function; and
    5. Obtain the confusion matrix by calling sklearn.metrics.confusion_matrix(Y_test, y_hat) and visualize it.
- GitHub URL of the completed predictive analysis lab:

https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module41_jupyter-labs_spacex_machine-learning-prediction-mine.ipynb


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

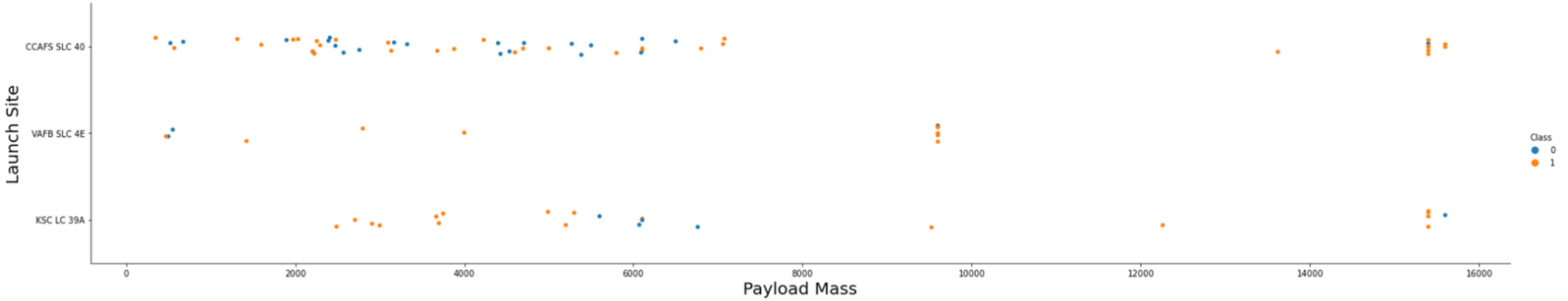
[6]:
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



- The above is a scatter plot of Flight Number vs. Launch Site.
- CCAFS SLC 40 is the main launch site, except for flight numbers 25-42, when KSC LC 39A carried out most launches. Site VAFB SLC 4E carried a minority of launches.

Payload vs. Launch Site

```
[7]: # 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 = 5)
plt.xlabel("Payload Mass",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



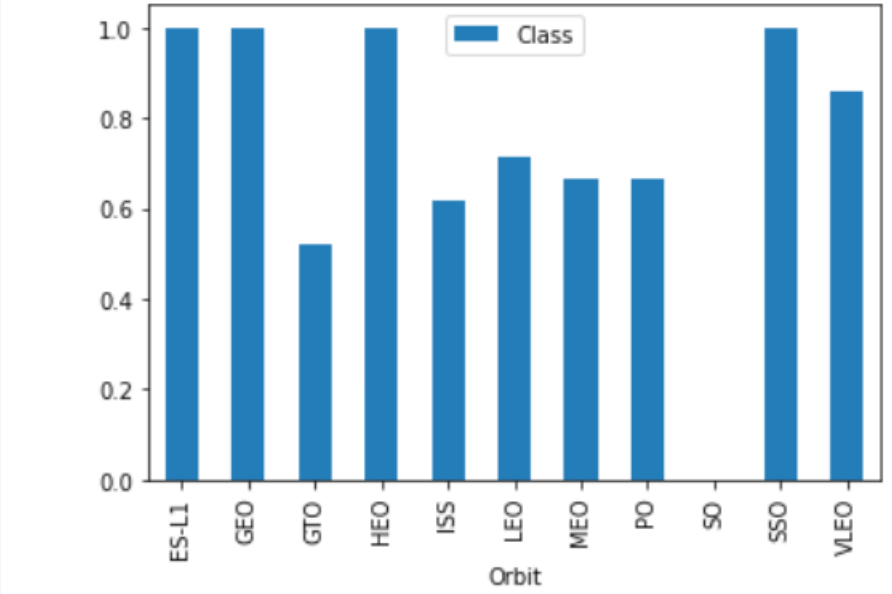
- The above is a scatter plot of Payload vs. Launch Site
- Both CCAFS SLC 40 and KSC LC 39A had launches with max payload mass close to 16,000 kg, whereas VAFB SLC 4E had launches with max payload mass close to 10,000 kg only.

Success Rate vs. Orbit Type

- On the right is a bar chart for the success rate of each orbit type.
- Orbit types ES-L1, GEO, HEO, and SSO have 100% success rate.
  - Note ES-L1, GEO, and HEO has only 1 launch each, while SSO has 5 launches.
- Orbit types GTO, ISS, LEO, MEO, PO, and VLEO have success rates between 50% and 90%.
- Orbit type SO has no computed success rate or missing data.

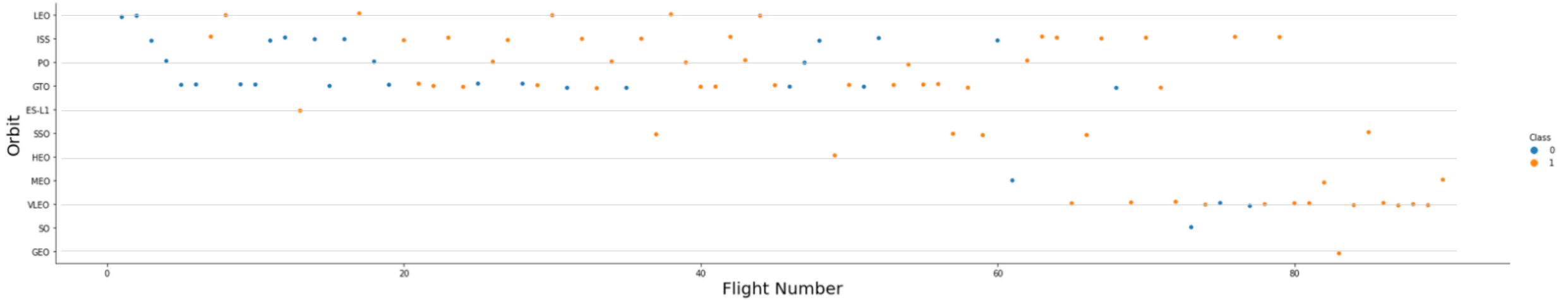
```
[8]: # HINT use groupby method on Orbit column and get the mean of Class column
     df[['Orbit', 'Class']].groupby('Orbit').mean().plot(kind='bar')
```

```
[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7845b27190>
```



Flight Number vs. Orbit Type

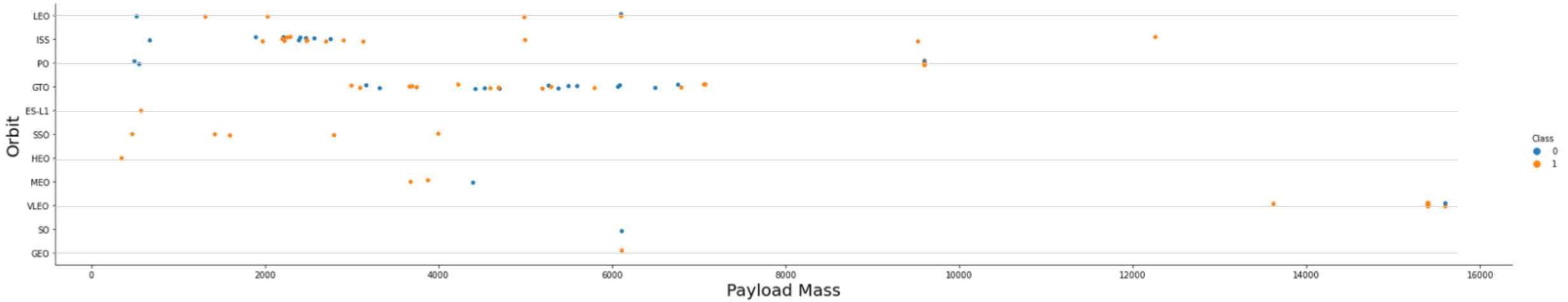
```
[9]: # Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



- The above is a scatter point of Flight number vs. Orbit type.
- Flights number 1 to 55 are mostly these 4 orbit types: LEO, ISS, PO, and GTO.
- Since flight number 56, the orbit types are more varied, with VLEO becoming one of the main orbit types.

Payload vs. Orbit Type

```
[10]: # 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 = 5)
plt.xlabel("Payload Mass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

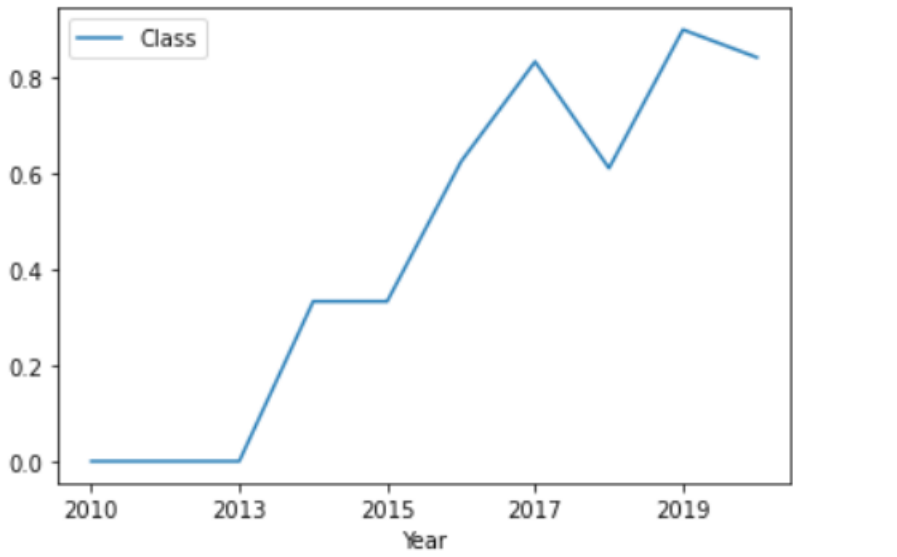


- The above shows a scatter plot of payload vs. orbit type
- We can see launches for dfferent orbits tend to have different Payload Mass values:
  - Launches for VLEO has the greatest Payload Mass, all >13,500 kg.
  - Launches for ISS has the greatest range of Payload Mass, from <1,000 kg to >12,000 kg.
  - Launches for GTO has Payload Mass in the range of ~3,000 kg to ~7,000 kg.

Launch Success Yearly Trend

- On the right is a line chart of yearly average success rate.
- From the chart we can see the following:
  - SpaceX did not manage to land any first stage successfully between 2010 and 2013.
  - Since their first successful landing in 2014, SpaceX managed to incrase the success rate to >80% in 2019 and 2020.

```
[15]: df[['Year', 'Class']].groupby('Year').mean().plot()

[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7844f0f350>
```



All Launch Site Names

Task 1

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

```
[17]: %sql SELECT UNIQUE launch_site FROM eda_with_sql
```

```
 * ibm_db_sa://ytj70292:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

[17]:

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

- The names of the 4 unique launch sites are shown above.
- SQL keyword “UNIQUE” helps to remove repeating values from the query result.

Launch Site Names Begin with 'CCA'

Task 2

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

[28]:
```
%sql SELECT * FROM eda_with_sql WHERE launch_site LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://ytj70292:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

[28]:

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

- The above query finds 5 records where launch sites begin with `CCA`
- SQL keyword “LIKE” helps match the launch_site name with the required string pattern, and “LIMIT 5” establishes the max number of results to be returned.

Total Payload Mass

Task 3

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

```
[31]: %sql SELECT SUM(payload_mass__kg_) FROM eda_with_sql WHERE UCASE(customer)='NASA (CRS)'
 * ibm_db_sa://ytj70292:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
[31]:    1
45596
```

- The above query calculates the total payload carried by boosters from NASA.
- SQL function SUM() specifies the total payload mass to be returned, and “WHERE UCASE(customer)=’NASA (CRS)’” identifies the customer of interest and makes it insensitive to case.

Average Payload Mass by F9 v1.1

Task 4

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

```
[34]: %sql SELECT AVG(payload_mass__kg_) FROM eda_with_sql WHERE booster_version like 'F9 v1.1%'
```

```
 * ibm_db_sa://ytj70292:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

```
[34]:    1
```

```
2534
```

- The above query calculates the average payload mass carried by booster version F9 v1.1
- SQL function AVG() specifies the average payload mass to be returned, and “WHERE booster_version LIKE ‘F9 v1.1%’” pattern-matches to include all F9 v1.1 boosters.

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

[37]:
```
%sql SELECT MIN(date) FROM eda_with_sql WHERE landing__outcome='Success (ground pad)'
```

```
 * ibm_db_sa://ytj70292:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

[37]:

1
2015-12-22

- The above query finds the date of the first successful landing outcome on ground pad.
- SQL function MIN() ensures the first or smallest date is to be returned, and “WHERE landing_outcome=’’Success (ground pad)” narrows down the landing target and outcome.

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

[44]:
```
%sql SELECT booster_version,landing__outcome,payload_mass__kg_ FROM eda_with_sql WHERE landing__outcome='Success (drone ship)' AND payload_mass__kg_ BETWEEN 4000 AND 6000
```

```
 * ibm_db_sa://ytj70292:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

[44]:

booster_version	landing__outcome	payload_mass__kg_
F9 FT B1022	Success (drone ship)	4696
F9 FT B1026	Success (drone ship)	4600
F9 FT B1021.2	Success (drone ship)	5300
F9 FT B1031.2	Success (drone ship)	5200

- The above query lists the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 kg but less than 6000 kg.
- SQL keyword “AND” is used to combine 2 conditions into a logical “and”.

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

```
[45]: %sql SELECT COUNT(mission_outcome) FROM eda_with_sql WHERE UCASE(mission_outcome) LIKE 'SUCCESS%'
```

```
 * ibm_db_sa://ytj70292:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

[45]:

1
100

```
[46]: %sql SELECT COUNT(mission_outcome) FROM eda_with_sql WHERE UCASE(mission_outcome) LIKE 'FAILURE%'
```

```
 * ibm_db_sa://ytj70292:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

[46]:

1
1

- The above 2 queries calculate the total number of successful and failure mission outcomes.
- To combine them into 1 query, I would consider the following:
- %sql SELECT COUNT(mission_outcome) FROM eda_with_sql GROUP BY UCASE(mission_outcome)

Boosters Carried Maximum Payload

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
[50]: %sql SELECT booster_version,payload_mass__kg_ FROM eda_with_sql WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM eda_with_sql)
```

```
 * ibm_db_sa://ytj70292:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

[50]:

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

- The above query lists the names of the booster which have carried the maximum payload mass.
- The subquery first establishes MAX(payload_mass__kg_) as 15600 kg, which is then used to identify the booster_versions carrying this max payload mass.

2015 Launch Records

Task 9

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

[53]:
```
%sql SELECT date,booster_version,launch_site,landing__outcome FROM eda_with_sql WHERE landing__outcome='Failure (drone ship)' AND YEAR(date)=2015
```

```
 * ibm_db_sa://ytj70292:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

[53]:

DATE	booster_version	launch_site	landing__outcome
2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

- The above query lists the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015.
- The required columns are identified after keyword SELECT and the 2 conditons are combined in the WHERE clause with keyward AND. Note also the usage of SQL function YEAR().

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

```
[66]: %sql SELECT landing__outcome,COUNT(landing__outcome) FROM eda_with_sql WHERE date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY landing__outcome ORDER BY COUNT(landing__outcome) DESC
 * ibm_db_sa://ytj70292:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/BLUDB
Done.
```

[66]:

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

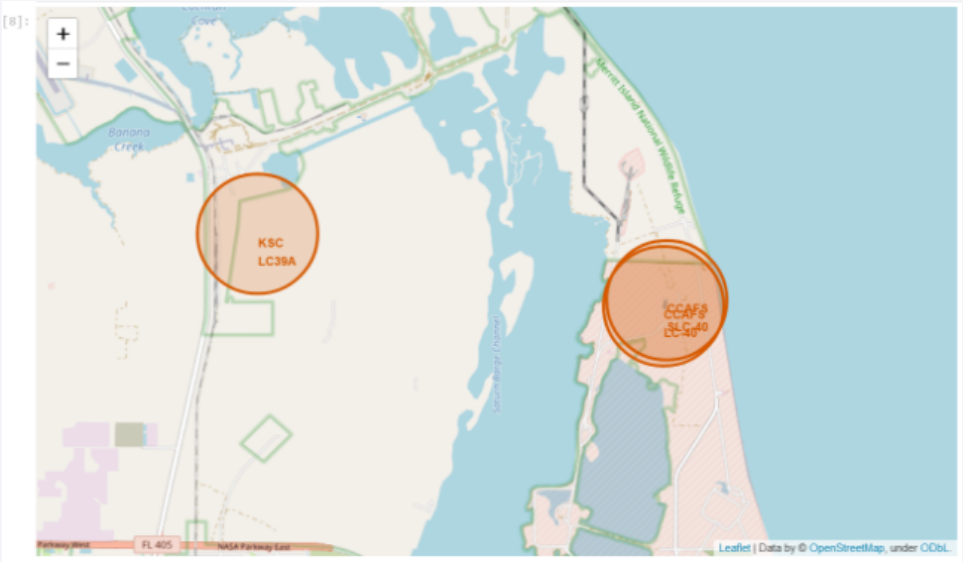
- The above query ranks the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.
- Note the use of the following SQL function or keywards:
  - COUNT(), BETWEEN, GROUP BY, ORDER BY, and DESC.

Section 3

Launch Sites Proximities Analysis

4 Launch Sites: 3 in Florida and 1 in California

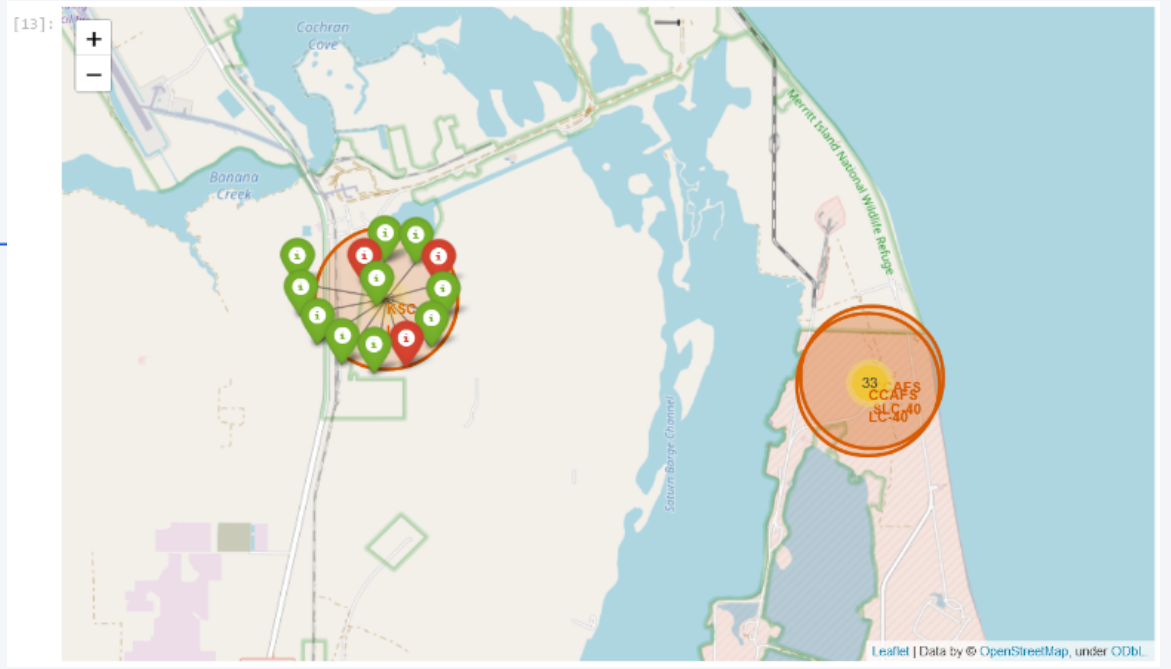
- VAFB SLC4E is in California.
- The rest 3 are in Florida, including KSC LC39A, CCAFS LC40, and CCAFS SLC40.
- CCAFS LC40 and CCAFS SLC40 are very close together.

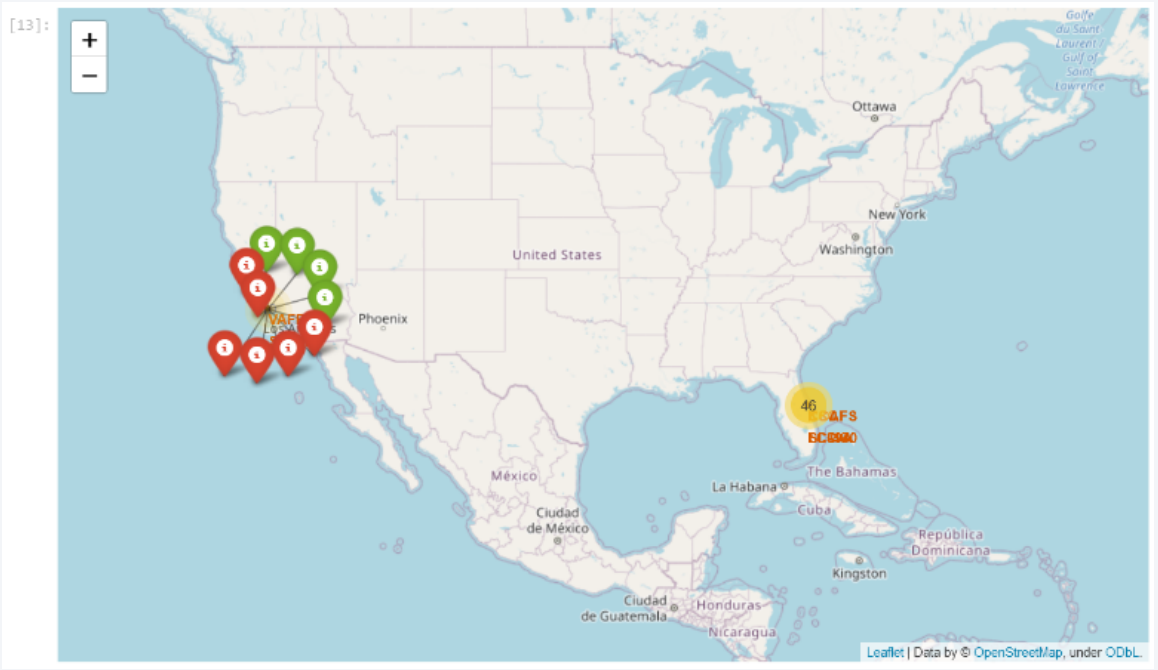


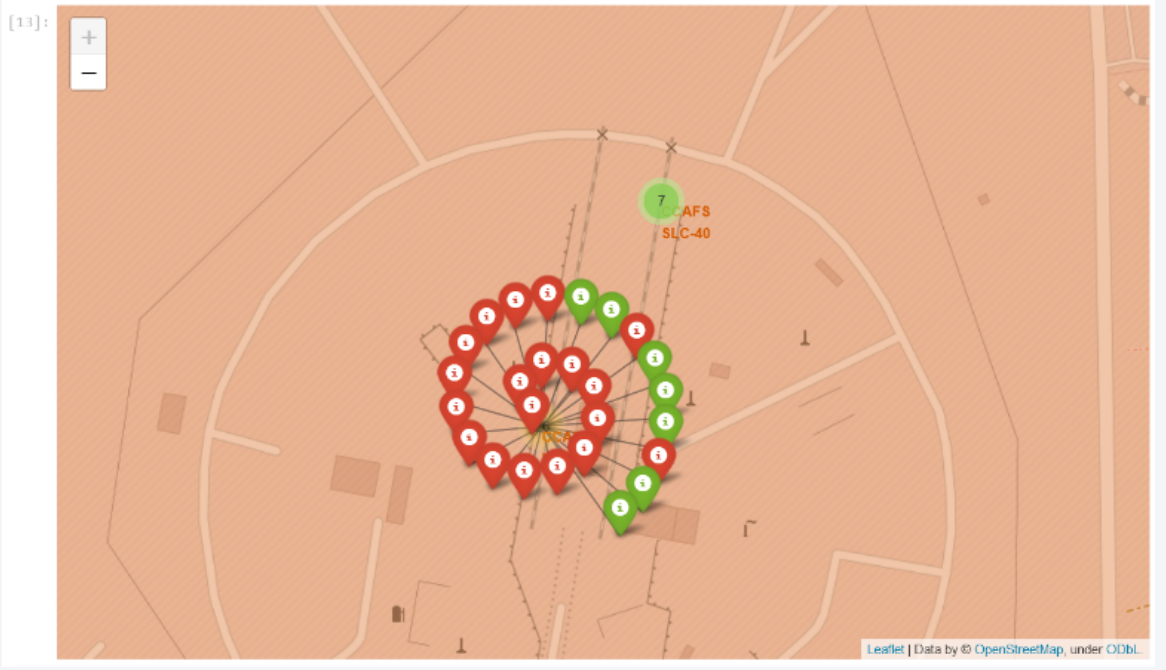

```
vafb_slc4e_coord = [34.632834, -120.610746]
circle_vafb_slc4e = folium.Circle(vafb_slc4e_coord, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('VAFB SLC4E'))
marker_vafb_slc4e = folium.Marker(vafb_slc4e_coord, icon=DivIcon(icon_size=(20,20), icon_anchor=(0,0),
                                                                 html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>'
site_map.add_child(circle_vafb_slc4e)
site_map.add_child(marker_vafb_slc4e)

site_map
```

Landing outcomes per site

- 4 out of 10 (4/10) launches at VAFB SLC4E had the first-stages landed successfully.
- 10/13 launches at KSC LC39A landed successfully.
- 7/26 launches at CAFS LC40 landed successfully.
- 3/7 launches at CAFS SLC40 landed successfully.



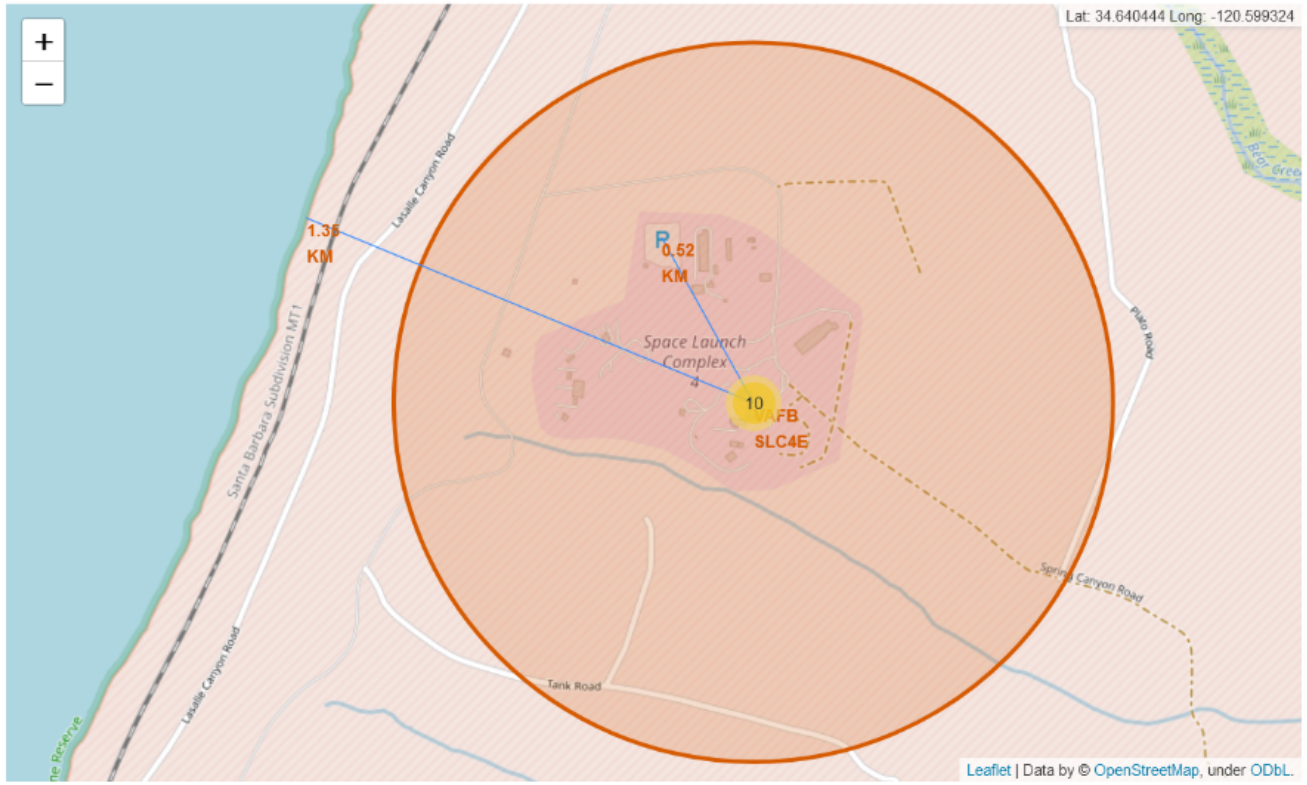




Proximity analysis of site VAFB SLC4E

```
[28]: # Create a marker with distance to a closest city, railway, highway, etc.
      # Draw a line between the marker to the launch site
      parking_coord = [34.636942, -120.613527]
      distance_parking = calculate_distance(vafb_slc4e_coord[0], vafb_slc4e_coord[1], parking_coord[0], parking_coord[1])
      parking_dist_marker = folium.Marker(parking_coord, icon=DivIcon(icon_size=(20,20), icon_anchor=(0,0),
                                                                      html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' %
      parking_line=folium.PolyLine(locations=[parking_coord, vafb_slc4e_coord], weight=1)
      site_map.add_child(parking_dist_marker)
      site_map.add_child(parking_line)
      site_map
```

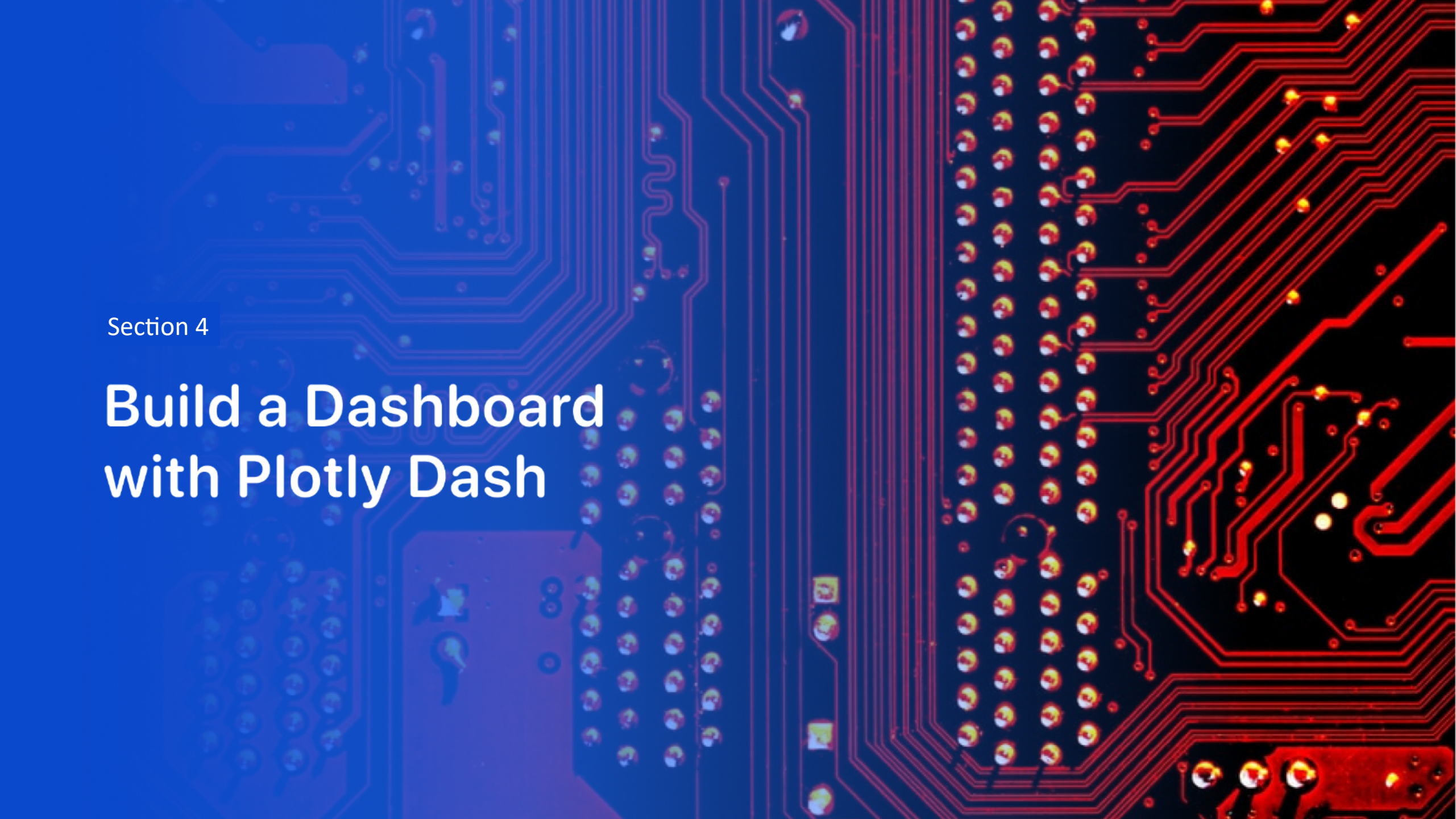
[28]:



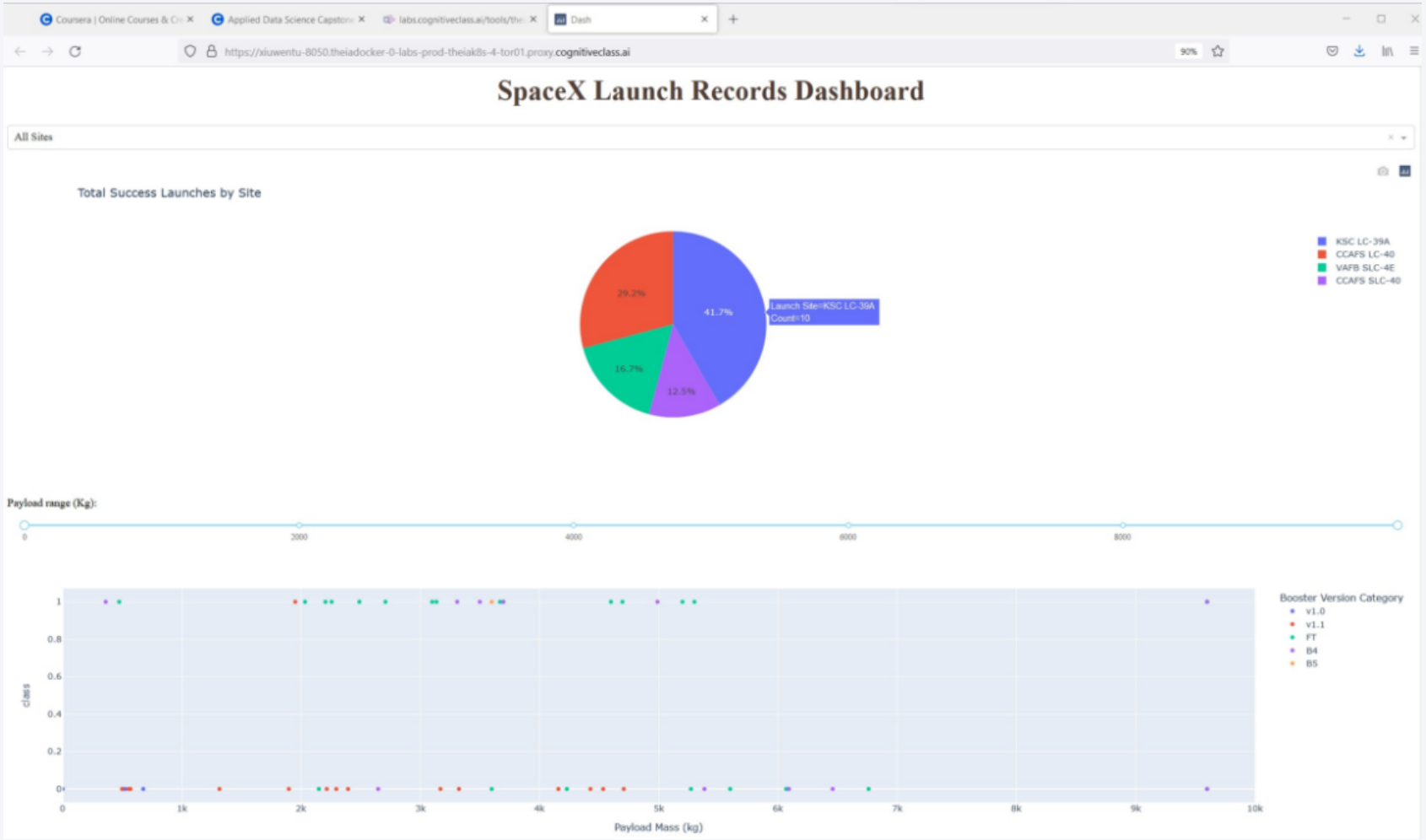
- Site VAFB SLC4E is 1.35 kilometer (km) away from the coast, and a railway is a little closer.
- It is 0.52 km away from its parking lot.
- It makes sense for a launch site to be close to the ocean. In case of a failed launch, the rocket and payload can be directed into the ocean for safety.
- Being close to railway can help with transportation of rocket and payload components.

Section 4

Build a Dashboard with Plotly Dash

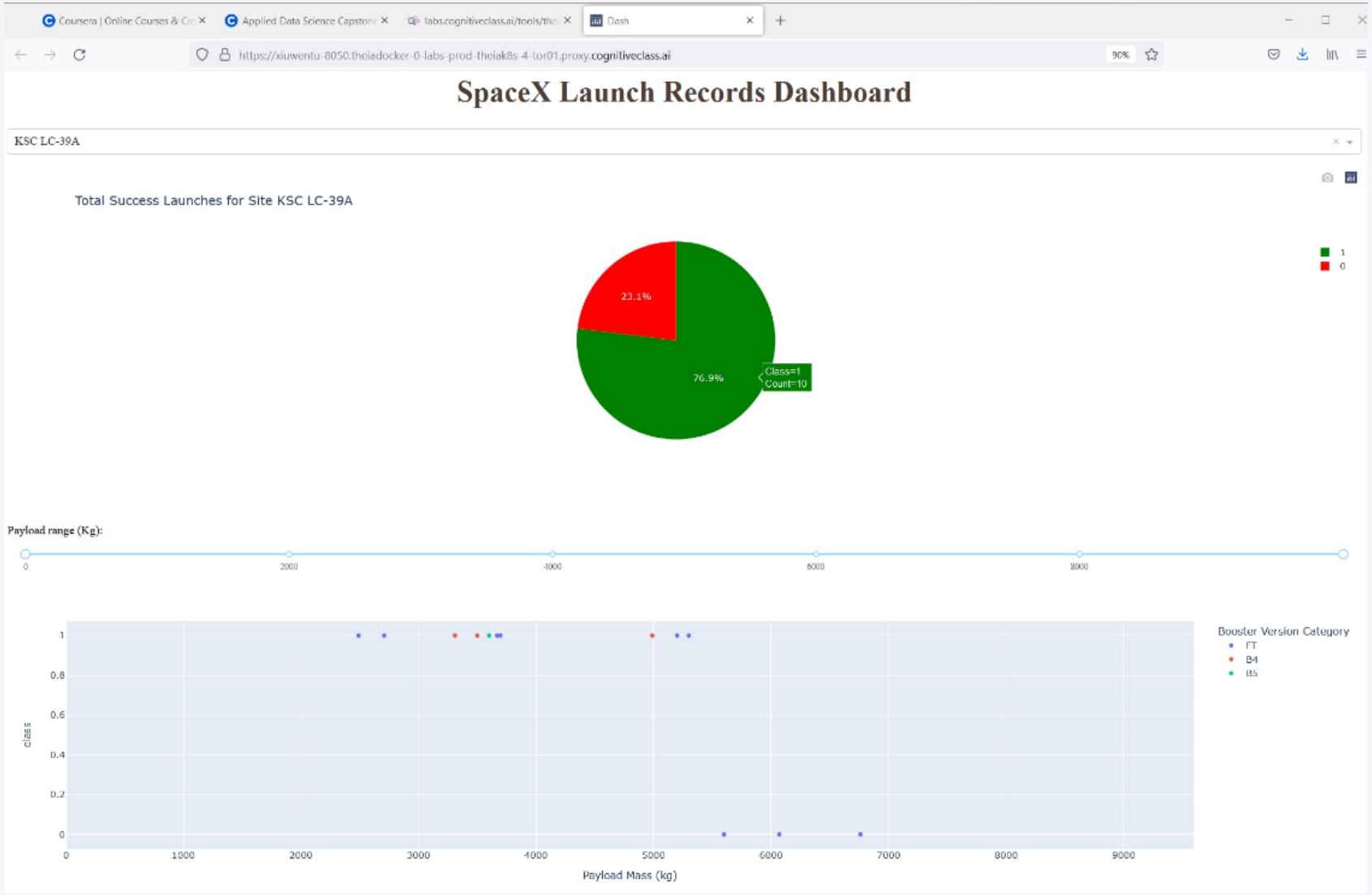


Landing success counts for all sites



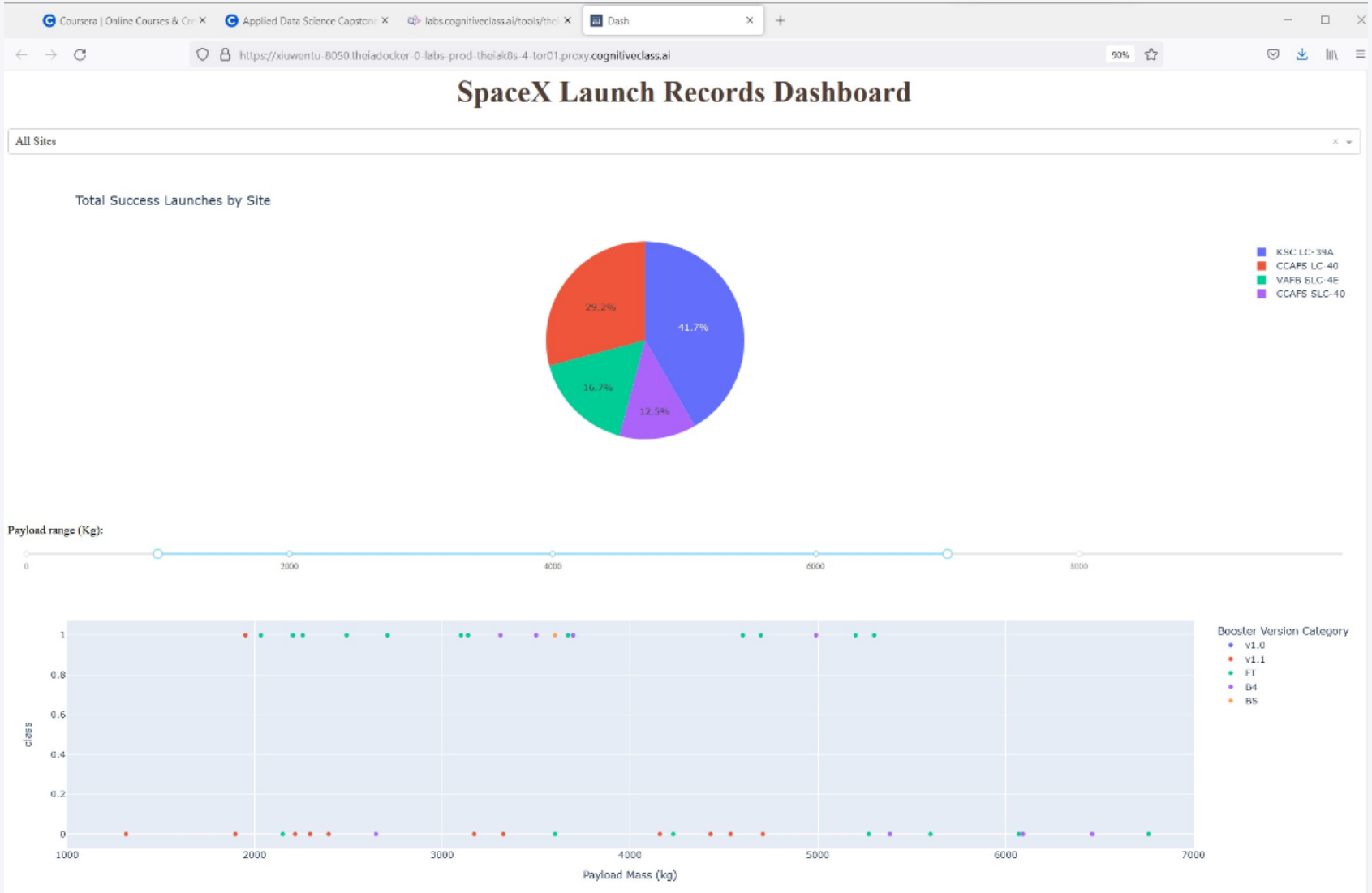
- The left shows the screenshot of launch success count for all sites.
- In this dataset, there are 24 success landing cases:
  - 41.7% (= 10/24) at KSC LC-39A
  - 29.2% (= 7/24) at CCAFS LC-40
  - 16.7% (= 4/24) at VAFB SLC-4E
  - 12.5% (= 3/24) at CCAFS SLC-40

KSC LC-39A has the highest landing success rate



- The left shows the screenshot of the piechart for KSC LC-39A, which has the highest launch success ratio of 76.9%, or 10/13.
- The success ratios of the rest 3 launch sites are lower:
  - CCAFS SLC-40 at 42.9% (= 3/7);
  - VAFB SLC-4E at 40.0% (= 4/10);
  - CCAFS LC-40 at 26.9% (= 7/26).

Payload mass vs launch outcome plot for all sites



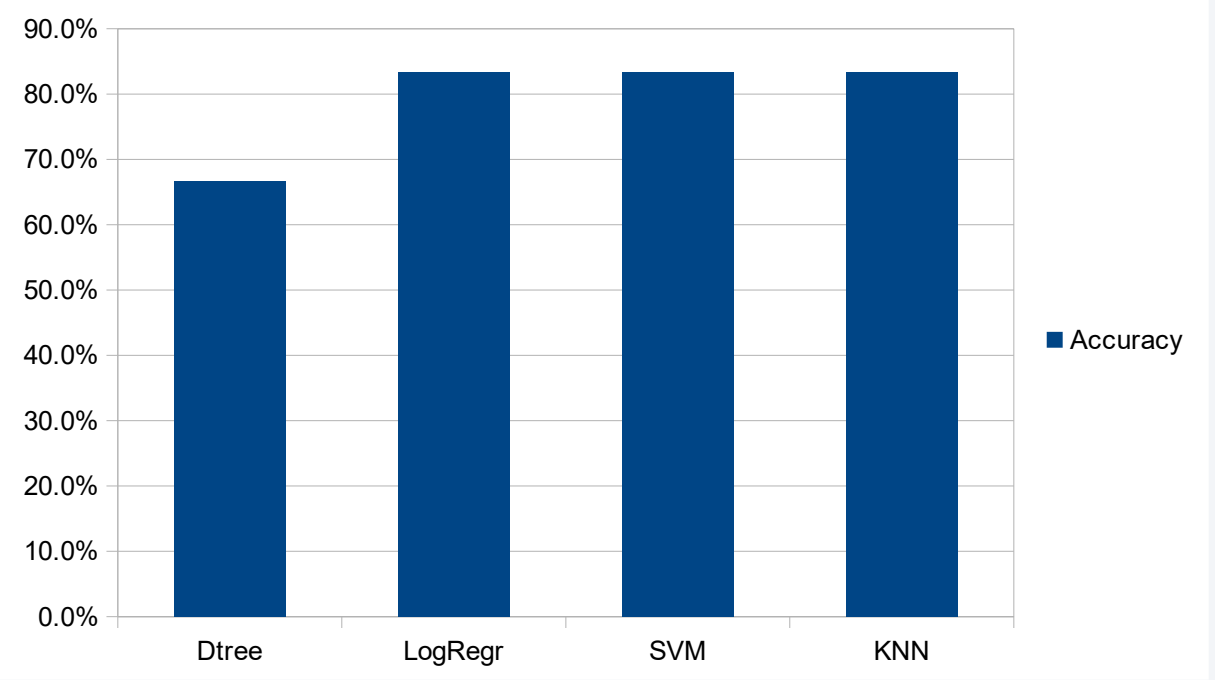
- The left shows screenshot of Payload vs. Launch Outcome scatter plot for all sites, with reduced payload mass range 1000-7000 kg selected.
- Within this reduced payload mass range 1000-7000 kg, successful landings are found only in
  - 1800-3800 kg and
  - 4500-5400 kg.

Section 5

Predictive Analysis (Classification)

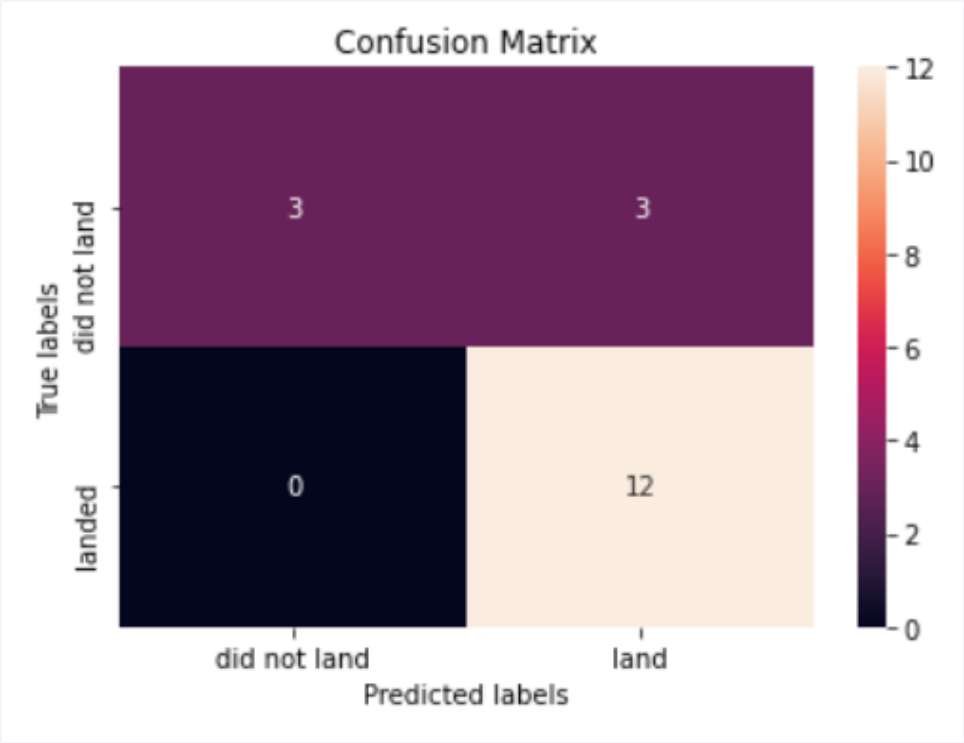
Classification Accuracy

- The right chart shows the test prediction accuracy for all 4 built classification models.
- The way I implemented them, three models have equal classification accuracy of 83.3%: Logistic Regression, SVM, and KNN;
- In comparison, Decision Tree has lower accuracy of 66.7%.



Confusion Matrix

- The way I implemented them, three models have equal classification accuracy and identical confusion matrix:
  1) Logistic Regression,
  2) SVM, and
  3) KNN.
- As can be seen on the right, out of the 18 test cases:
  - The 12 launches that landed successfully are all predicted correctly;
  - The 6 launches that did not land successfully are predicted correctly for 3 cases and incorrectly for the rest 3 cases.



Conclusions

- The test prediction accuracy for the landing outcome is 83.3% for 3 machine learning models:
  - Logistic Regression,
  - SVM, and
  - KNN.
- The accuracy of Decision Tree is lower at 66.7%.

Appendix – Links to my GitHub repository for this course

https://github.com/xiuwen-tu/applied_ds_capstone

https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module11_jupyter-labs-spacex-data-collection-api-mine.ipynb

https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module12-jupyter-labs-spacex-data-collection-webscraping-mine.ipynb

https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module13_jupyter-labs-spacex-data-wrangling-mine.ipynb

https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module21_jupyter-labs_spacex-eda_with_sql.ipynb

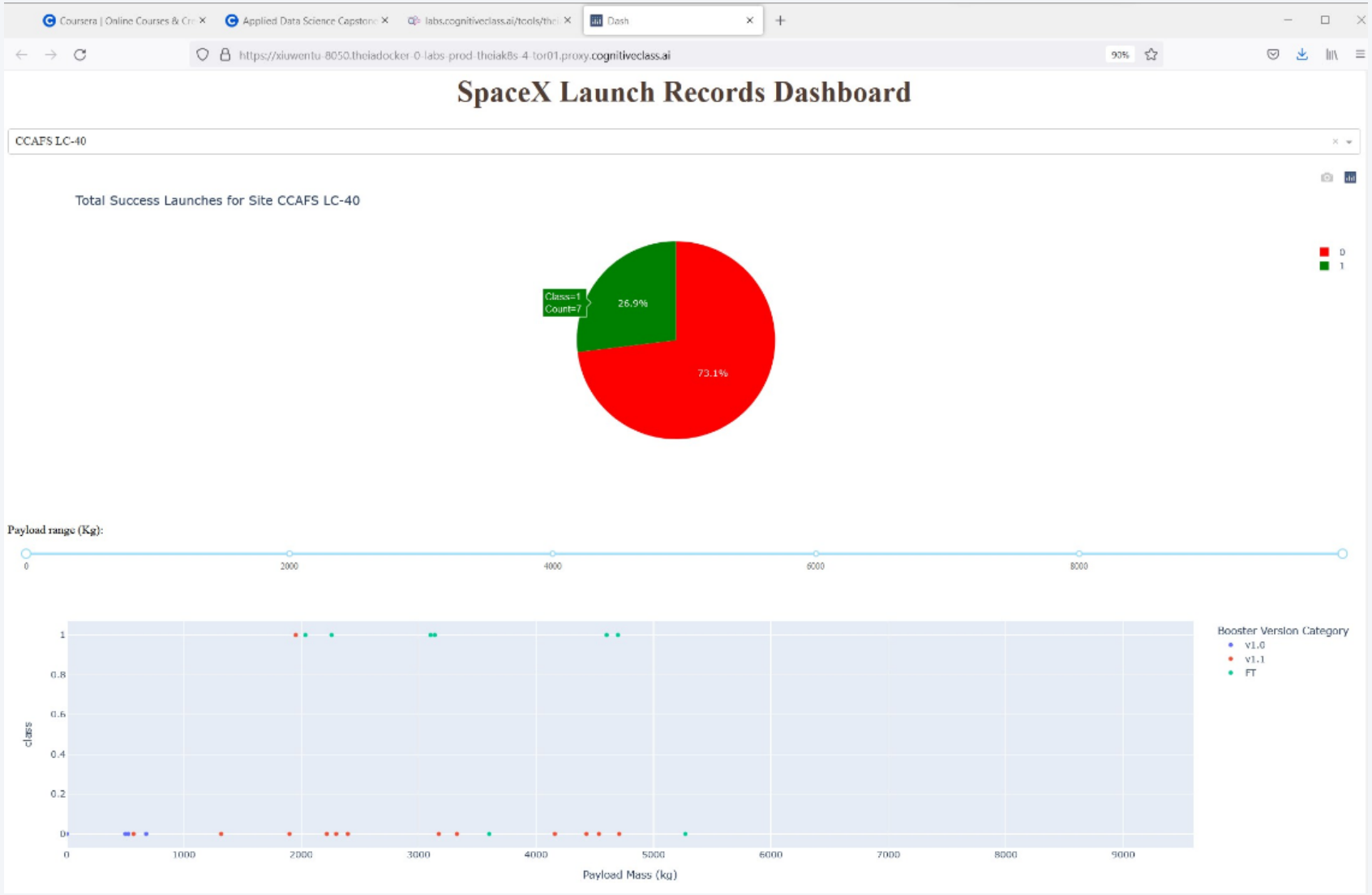
https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module22_jupyter-labs-spacex-eda-dataviz-pandas-mine.ipynb

https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module31_jupyter-labs_spacex-folium-map-launch_site_location-mine.ipynb

https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module32_plotly-dash-mine.py

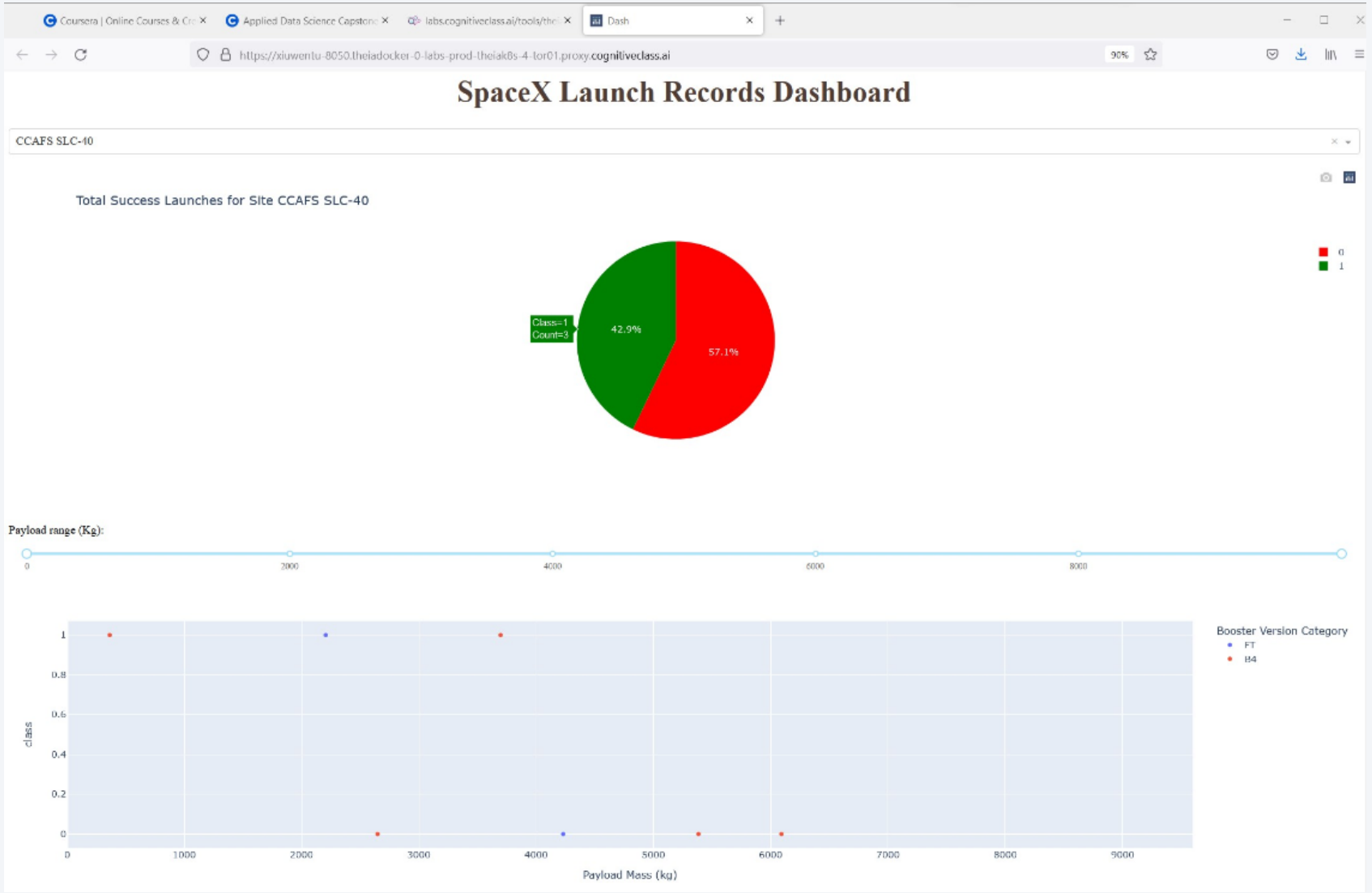
https://github.com/xiuwen-tu/applied_ds_capstone/blob/main/module41_jupyter-labs_spacex_machine-learning-prediction-mine.ipynb

Appendix – CCAFS LC-40 has landing success rate of 26.9%



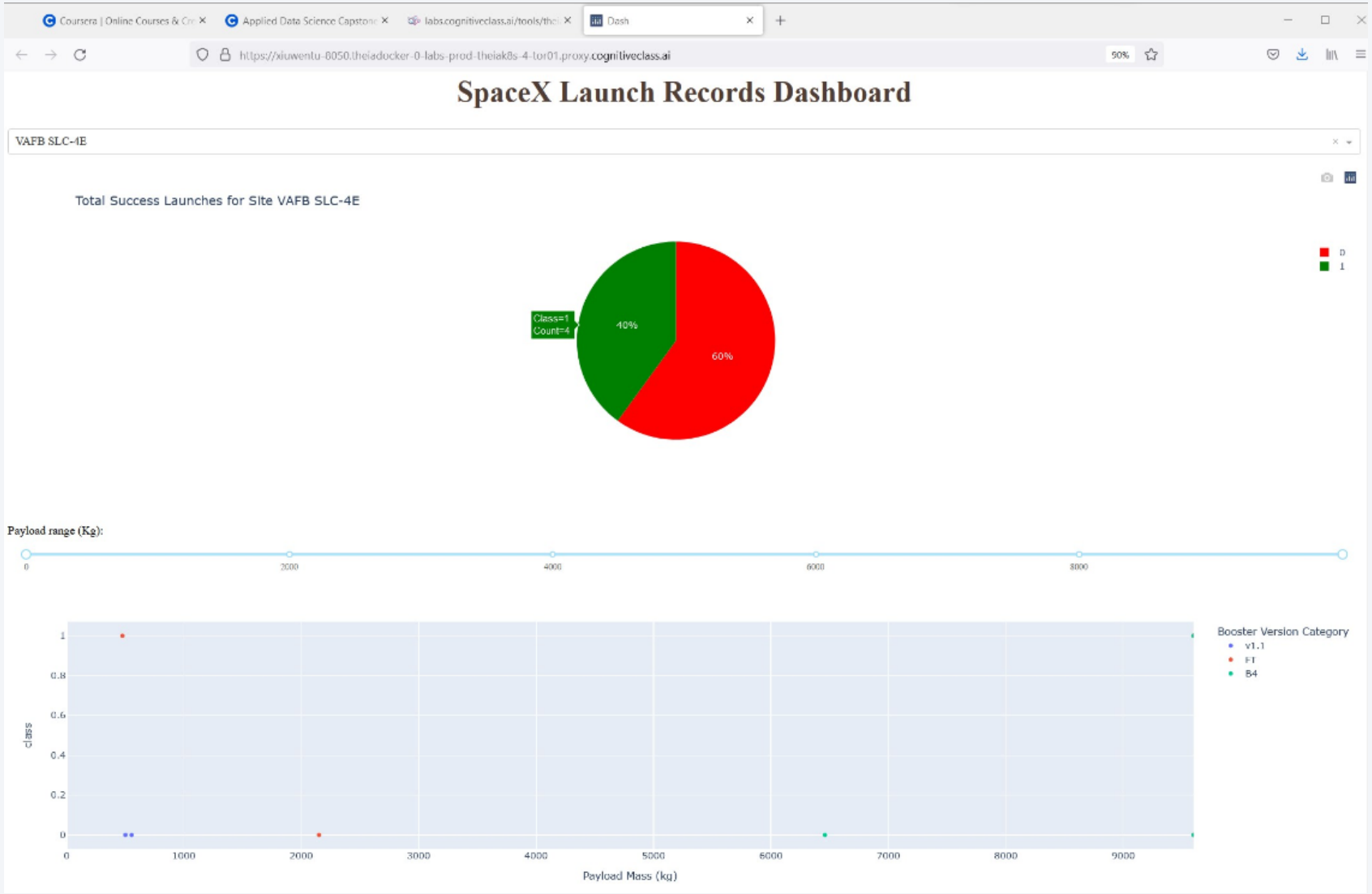
- The left shows the screenshot of the piechart for CCAFS LC-40, which has the launch success ratio of 26.9%, or 7/26.

Appendix – CCAFS SLC-40 has landing success rate of 42.9%



- The left shows the screenshot of the piechart for CCAFS SLC-40, which has the launch success ratio of 42.9%, or 3/7.

Appendix – VAFB SLC-4E has landing success rate of 40.0%



- The left shows the screenshot of the piechart for VAFB SLC-4E, which has the launch success ratio of 40.0%, or 4/10.

Thank you!