



IBM Developer  
SKILLS NETWORK

# Winning Space Race with Data Science

Shane Michael  
February 15<sup>th</sup>, 2025



# Outline

---

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

# Executive Summary

---

- Space X data was gathered, organized and analyzed using machine learning models which were evaluated for best use case.
- From the data analyzed the models all performed at the same level of predictive ability. The models were able to predict with high confidence if a particular launch is going to land the first stage or not.

# Introduction

---

- The project's goal is to find out the factors that contribute most to the successful landing of the first stage of Falcon 9 rockets
- This data will help SpaceX to standardize their launches and minimize failures to land to ensure their costs remain competitive with SpaceX





Section 1

# Methodology

# Methodology

---

## Executive Summary

- Data collection methodology:
  - Data was collected through a Database API as well as from webscraping tables from wikipedia
- Perform data wrangling
  - The data was processed using Python dataframes to determine success rates.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Tested multiple Classification models and tuned to most impactful hyperparameters with GridSearchCV. Compared accuracy scores to find best model.

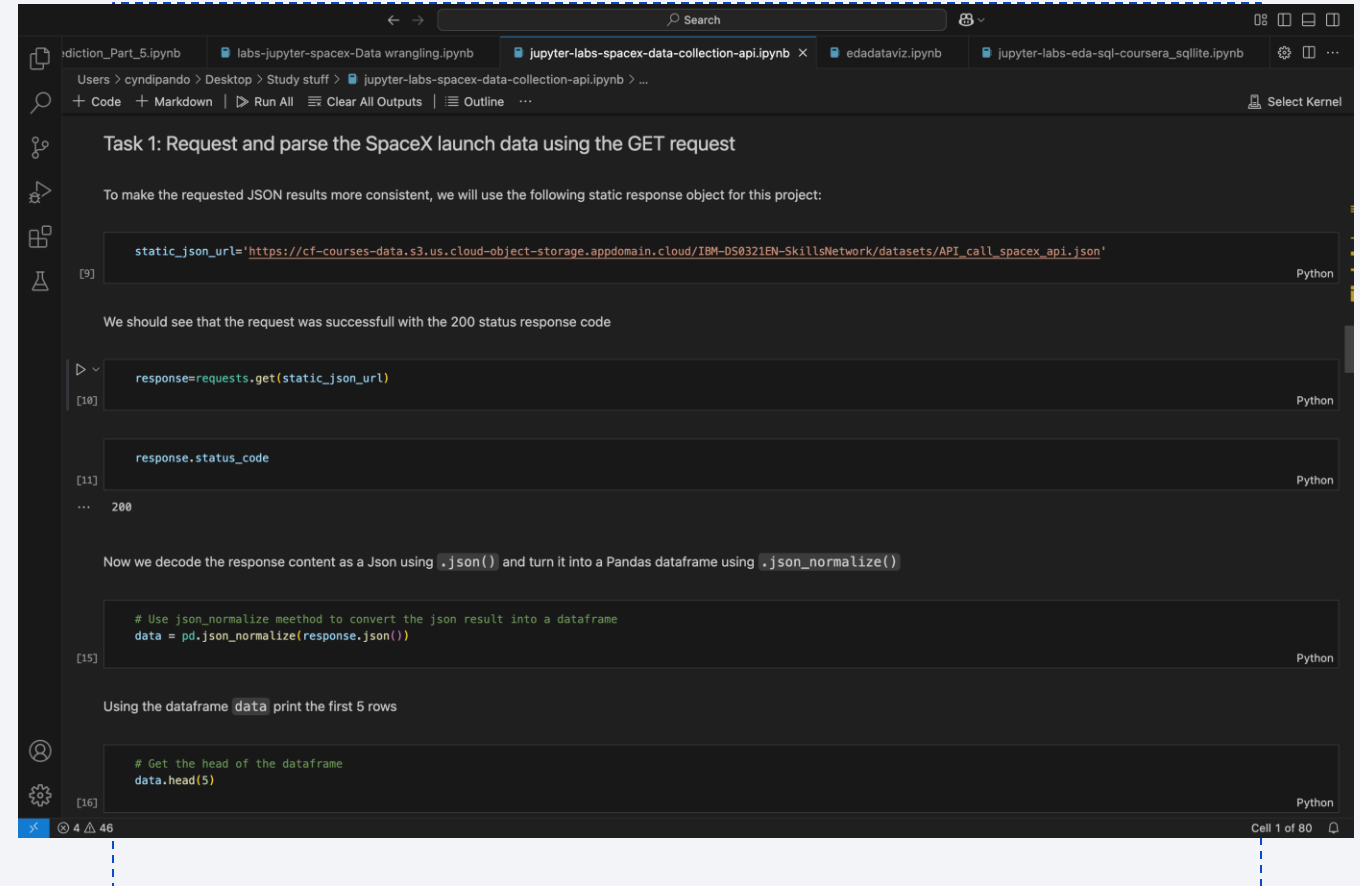
# Data Collection

---

- The data was collected from two main sources.
  - Through a REST api connecting to a database with preconstructed tables
  - Through Webscrapping of the Wikipedia page tables to find relevant data
- Data had to be normalized as it enters as a JSON file so using the `.json_normalize` method allowed it to be transformed into a dataframe.
- BeautifulSoup was used for webscraping the html tables of the Wikipedia article.
- After loading data it had to be cleaned, and missing values had to either be filled, dropped, or ignored depending on the data missing.

# Data Collection – SpaceX API

- To use the API we had to send a request to the server and await a response. This response was later transformed into the pandas dataframe.
- <https://github.com/Shane-Michael-git/IBMDataScienceCapstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb>



The screenshot displays a Jupyter Notebook interface with the following content:

- Task 1: Request and parse the SpaceX launch data using the GET request**
- Text: "To make the requested JSON results more consistent, we will use the following static response object for this project:"
- Code cell [9]:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DSE0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```
- Text: "We should see that the request was successful with the 200 status response code"
- Code cell [10]:

```
response=requests.get(static_json_url)
```
- Code cell [11]:

```
response.status_code
```

Output: 200
- Text: "Now we decode the response content as a json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`"
- Code cell [15]:

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```
- Text: "Using the dataframe `data` print the first 5 rows"
- Code cell [16]:

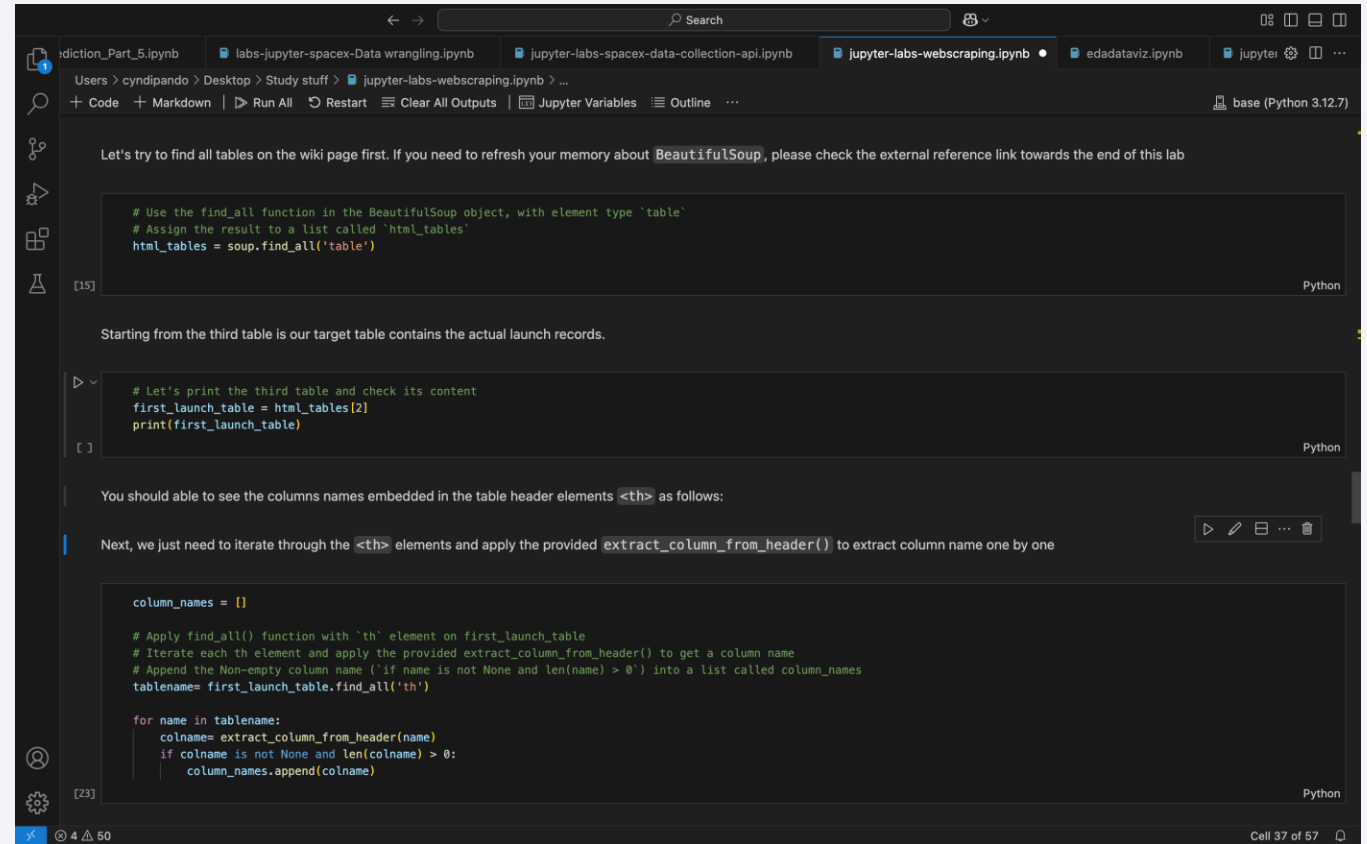
```
# Get the head of the dataframe
data.head(5)
```



# Data Collection - Scraping

- To Webscrape for information a BeautifulSoup object is created to parse the data of Falcon9 launches and eventually add the desired data to the dataframe.

- <https://github.com/Shane-Michael-git/IBMDataScienceCapstone/blob/main/jupyter-labs-webscraping.ipynb>



The screenshot shows a Jupyter Notebook with the following content:

Let's try to find all tables on the wiki page first. If you need to refresh your memory about BeautifulSoup, please check the external reference link towards the end of this lab

```
# Use the find_all function in the BeautifulSoup object, with element type 'table'
# Assign the result to a list called 'html_tables'
html_tables = soup.find_all('table')
```

[15]

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

[ ]

You should be able to see the columns names embedded in the table header elements `<th>` as follows:

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

```
column_names = []

# Apply find_all() function with 'th' element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name ('if name is not None and len(name) > 0') into a list called column_names
tablename= first_launch_table.find_all('th')

for name in tablename:
    colname= extract_column_from_header(name)
    if colname is not None and len(colname) > 0:
        column_names.append(colname)
```

[23]

Cell 37 of 57

# Data Wrangling

- To process this data further various metrics were calculated such as orbit type, launch site, and even the number of the launch itself to try to find relationships between these variables and the launch outcome.
- A landing outcome label was created to simplify the data as a number, 1 for successful landing, 0 for a failure.
- <https://github.com/Shane-Michael-git/IBMDDataScienceCapstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb>

The screenshot shows a Jupyter Notebook interface with the following content:

**TASK 4: Create a landing outcome label from Outcome column**

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

```
[11]: # landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcome:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

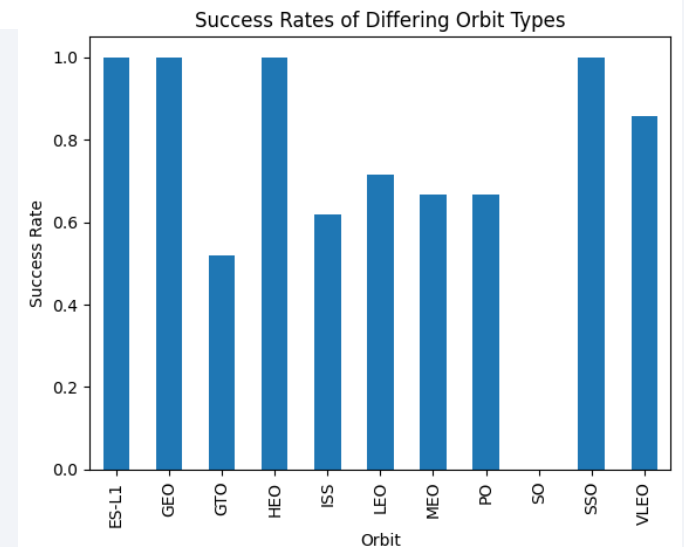
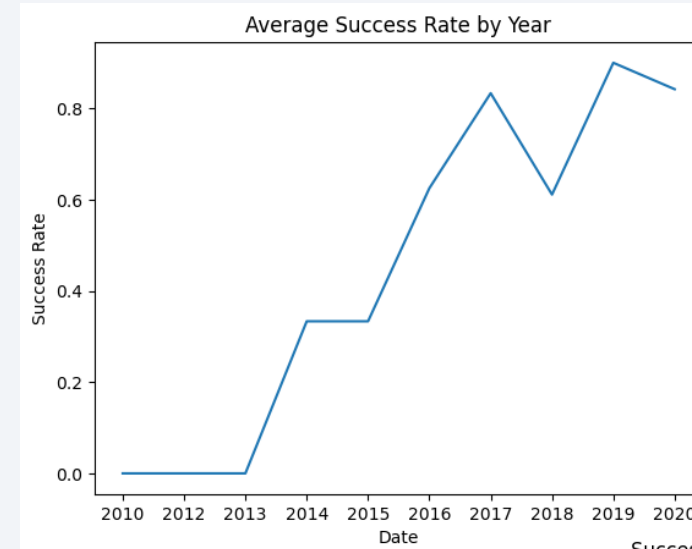
```
[12]: df['Class'] = landing_class
df[['Class']].head(8)
```

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

```
df.head(5)
```

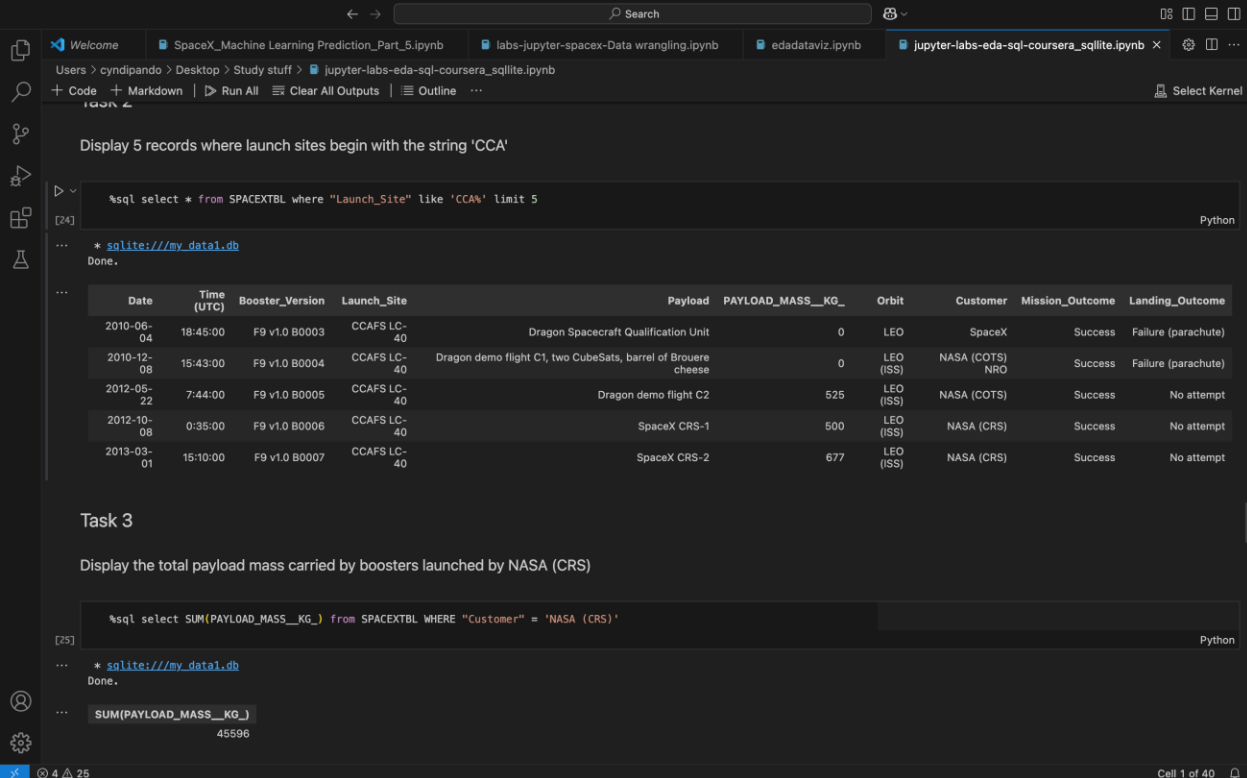
# EDA with Data Visualization

- During EDA relationships between variables were analyzed, flight number vs launch site, payload vs launch site, success rate of orbit types, flight number vs orbit type, and launch success rate by year. These charts help to visualize these relationships and determine the relevance of certain features.
- <https://github.com/Shane-Michael-git/IBMDataScienceCapstone/blob/main/edadataviz.ipynb>



# EDA with SQL

- Using SQL magics in jupyter notebooks we were able to take a look at specific data with SQL queries.
  - SQL queries are detailed further in the report but they allowed us to look at specific data in a more efficient manner than using pandas
- [https://github.com/Shane-Michael-git/IBMDataScienceCapstone/blob/main/jupyter-labs-eda-sql-coursera\\_sqlite.ipynb](https://github.com/Shane-Michael-git/IBMDataScienceCapstone/blob/main/jupyter-labs-eda-sql-coursera_sqlite.ipynb)



The screenshot shows a Jupyter Notebook interface with two code cells. The first cell contains a SQL query to display 5 records where launch sites begin with the string 'CCA'. The output shows a table with columns: Date, Time (UTC), Booster\_Version, Launch\_Site, Payload, PAYLOAD\_MASS\_KG\_, Orbit, Customer, Mission\_Outcome, and Landing\_Outcome. The second cell contains a SQL query to display the total payload mass carried by boosters launched by NASA (CRS). The output shows the sum of payload mass as 45596.

```
%sql select * from SPACEXTBL where "Launch_Site" like 'CCA%' limit 5
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-05-04	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

```
%sql select SUM(PAYLOAD_MASS_KG_) from SPACEXTBL WHERE "Customer" = 'NASA (CRS)'
```

```
SUM(PAYLOAD_MASS_KG_)
45596
```

# Build an Interactive Map with Folium

---

- Folium was used to visualize the launch site locations initially
- From there markers were added to show the success/fails of each launch site in a visual manner.
- Distance markers and lines were added to show the relative distance from a launch site to landmarks.
  - Coast
  - Highway
  - Railroad
  - City
- [https://github.com/Shane-Michael-git/IBMDataScienceCapstone/blob/main/lab\\_jupyter\\_launch\\_site\\_location.ipynb](https://github.com/Shane-Michael-git/IBMDataScienceCapstone/blob/main/lab_jupyter_launch_site_location.ipynb)



# Build a Dashboard with Plotly Dash

---

- Dash was used to build an online dashboard app using Plotly graphs as visual displays
- A Pie chart was chosen to display success rate of all sites and then success/fail of each individual site
- A payload slider was added to show a scatter plot of payload vs success
  - Booster version was included in the marker color to distinguish boosters
- [https://github.com/Shane-Michael-git/IBMDataScienceCapstone/blob/main/spacex\\_dash\\_app\(1\).py](https://github.com/Shane-Michael-git/IBMDataScienceCapstone/blob/main/spacex_dash_app(1).py)

# Predictive Analysis (Classification)

---

- The data was loaded into two dataframes and transformed by StandardScaler. It was then split with train\_test\_split into training and testing datasets
- Models for Logistic Regression, SVM, Decision Tree, and KNN were created and inserted into a GridSearchCV object. The training data was fit to that object and the ideal hyperparameters were found.
- Confusion matrices and accuracy scores were taken for each classifier to compare and contrast results.
- [https://github.com/Shane-Michael-git/IBMDataScienceCapstone/blob/main/SpaceX\\_Machine%20Learning%20Prediction\\_Part\\_5.ipynb](https://github.com/Shane-Michael-git/IBMDataScienceCapstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb)

# Results

---

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



The background of the slide is an abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks in shades of blue and red, creating a sense of motion or data flow. A faint, light blue grid pattern is also visible, particularly in the lower-left quadrant. The overall effect is high-tech and digital.

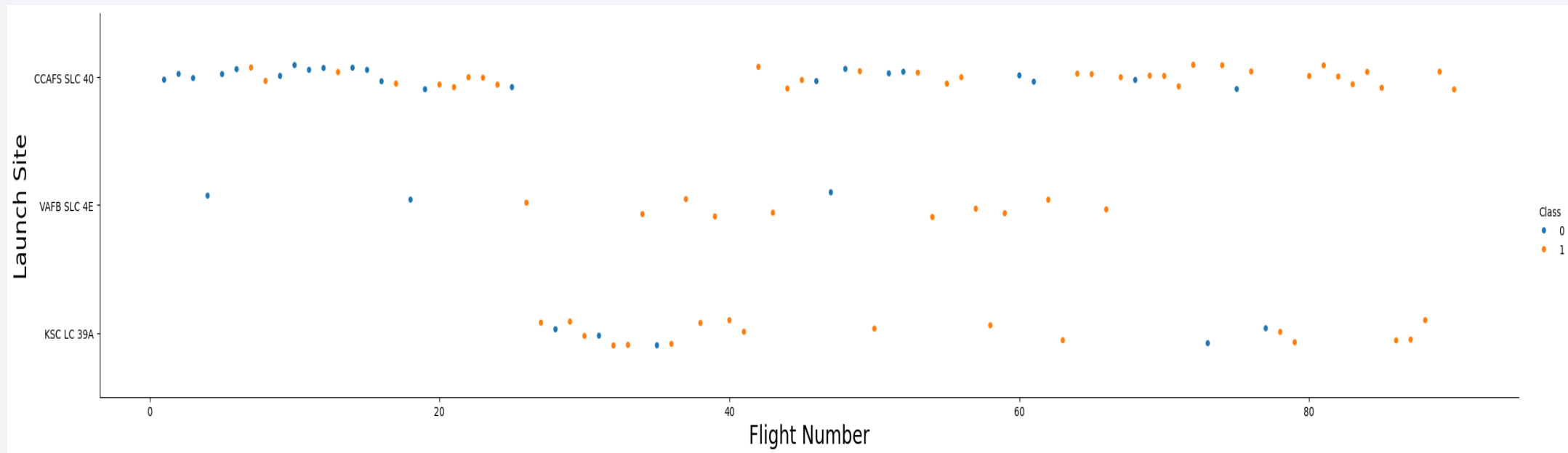
Section 2

# Insights drawn from EDA



# Flight Number vs. Launch Site

- The Scatter plot below shows that while it might be a very small difference there were more failures when there had been less flights. However there are strong diminishing returns.

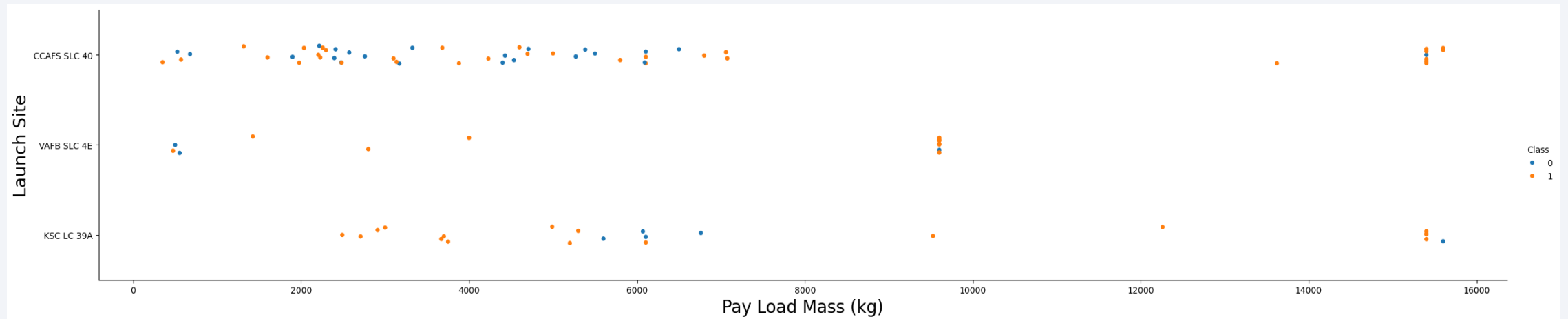




# Payload vs. Launch Site

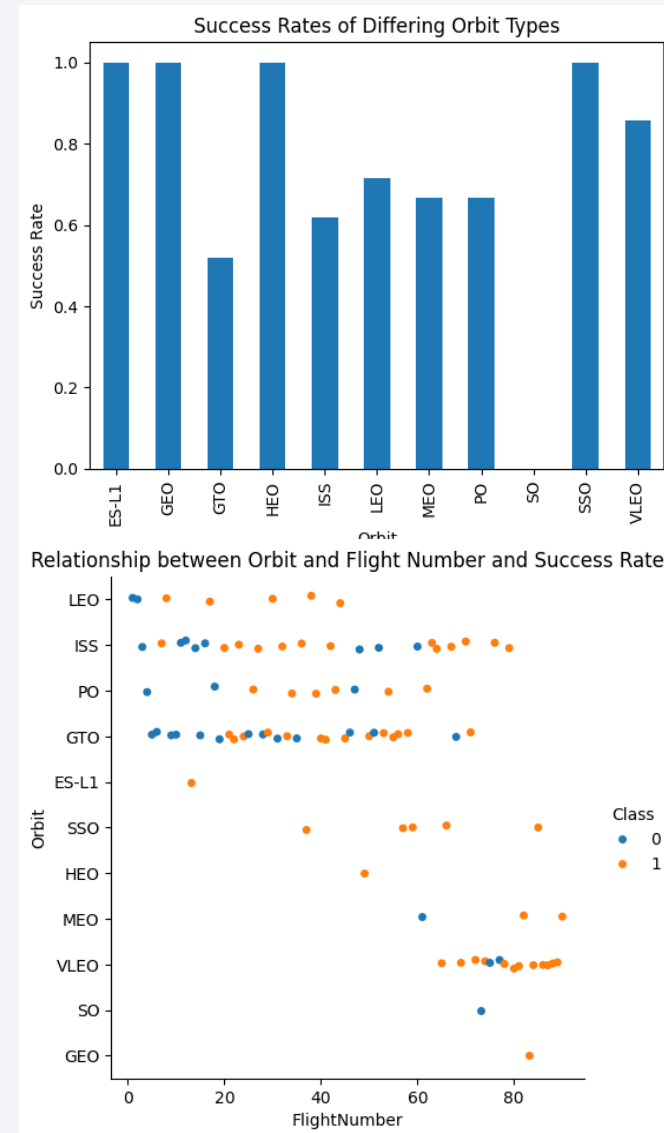
---

- From this data it can be surmised that the launch site VAFB SLC 4E does not launch rockets with a “heavy”(over 10,000kg) payloads.



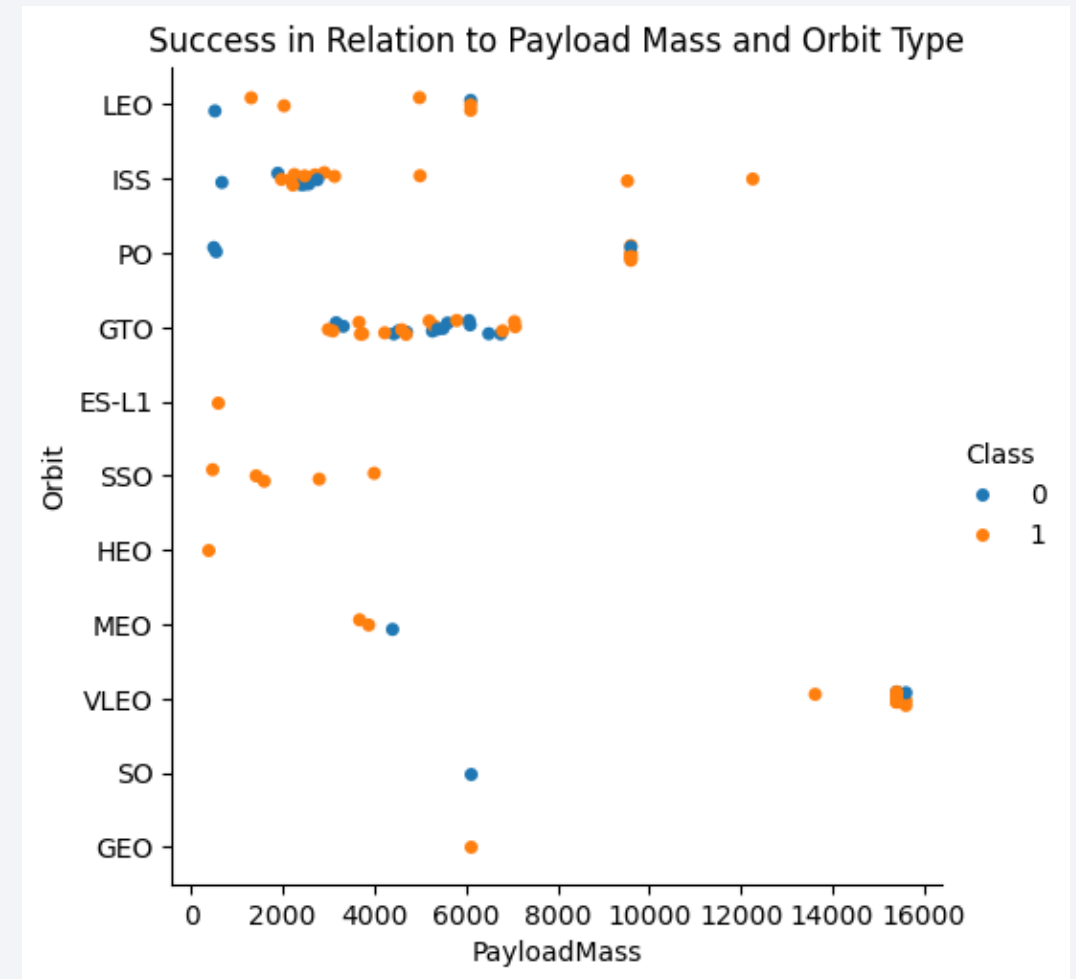
# Success Rate vs. Orbit Type

- Overall the riskiest orbit is GTO, although SO has a lower success rate there is simply not enough data to compare the two properly.
- ES-L1, GEO and HEO are poor examples of success as they are also singular instances.



# Payload vs. Orbit Type

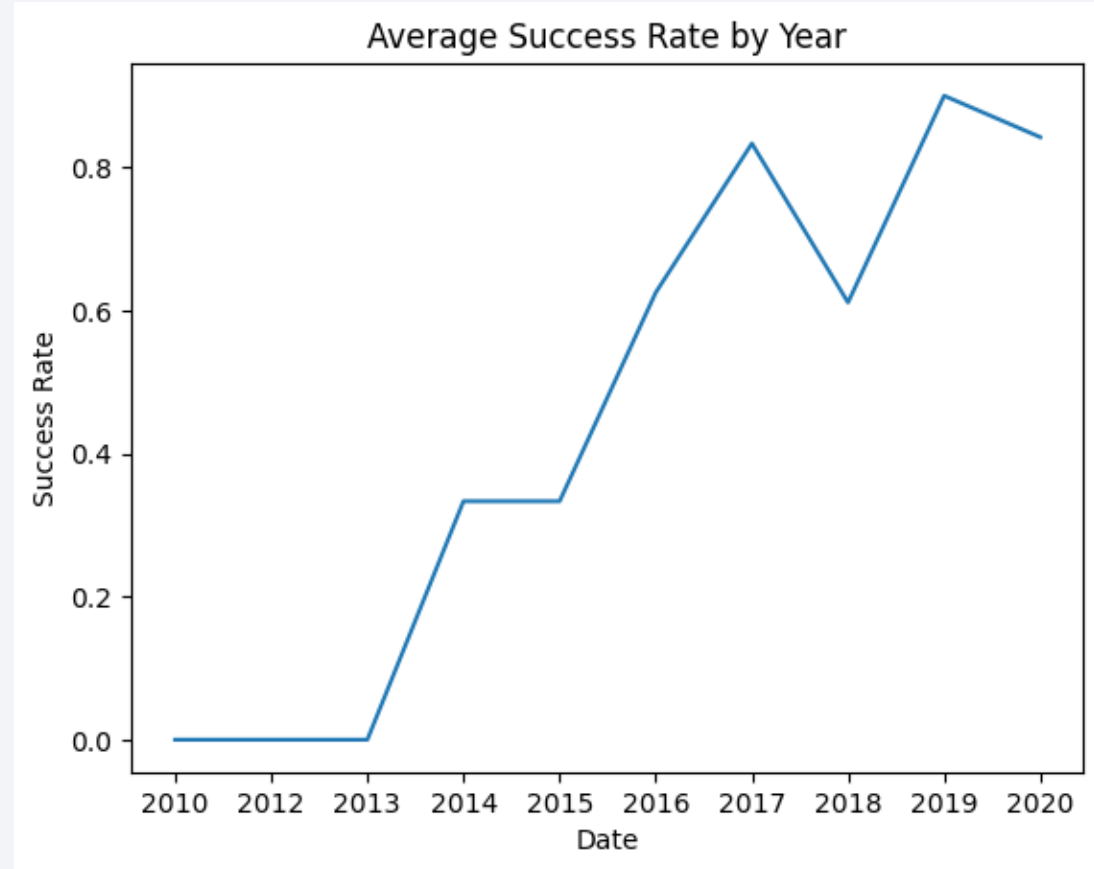
- The graph shows that payload mass is on average below 10,000 kg. This means very few orbit types carry heavy payloads
- Of the orbit types that carry heavy payloads VLEO has the highest number of successes while ISS and PO are not far behind.



# Launch Success Yearly Trend

---

- The chart shows that while there are areas where things level off or even dip, in general success rates of the first stage landing have increased with time. This can be assumed to be an effect of fine tuning over time.



# All Launch Site Names

---

- Four unique launch sites were found, their names are as follows; CCAFS LC-40, VAFB SLC-4E, KSC LC-39A, CCAFS SLC-40
- I was able to isolate these using the SQL query; select distinct "Launch\_Site" from SPACEXTBL
- This query only returns the unique names found in the "Launch\_Site" column.



# Launch Site Names Begin with 'CCA'

---

- Using the SQL query; select \* from SPACEXTBL where "Launch\_Site" like 'CCA%' limit 5
- This query shows the first 5 results it finds where the first letters of the value of "Launch\_Site" will be "CCA"

# Total Payload Mass

---

- The Total Payload Mass can be found with the query; select SUM(PAYLOAD\_MASS\_\_KG\_) from SPACEXTBL WHERE "Customer" = 'NASA (CRS)'
- This query gives the answer “45,596” meaning the Total of all payload masses is 45,596kg.

# Average Payload Mass by F9 v1.1

---

- The average Payload Mass for the F9 v1.1 booster can be found using the query; `select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where "Booster_Version" = 'F9 v1.1'`
- This query only returns the average of 2928.4 kg.

# First Successful Ground Landing Date

---

- To find the first ground landing date use the following query; `select min(date) from SPACEXTBL where "Landing_Outcome" = 'Success (ground pad)'`
- This filter the date to the lowest and ensures its looking only for successful ground pad landing outcomes. This gives a single value of 2015-12-22

## Successful Drone Ship Landing with Payload between 4000 and 6000

---

- To find successful drone ship landings with payloads over between 4-6 thousand kg use the query; select "Booster\_Version" from SPACEXTBL where "Landing\_Outcome" = 'Success (drone ship)' and PAYLOAD\_MASS\_\_KG\_ between 4000 and 6000
- This query returns the names of boosters that have successfully landed on a drone ship while specifically carrying between 4000 and 6000 kg of payload. Their names are;
  - F9 FT B1022
  - F9 FT B1026
  - F9 FT B1021.2
  - F9 FT B1031.2



# Total Number of Successful and Failure Mission Outcomes

---

- To calculate the total count of mission outcomes use the query; select count("Mission\_Outcome") from SPACEXTBL
- This query counts the total non empty rows in the "Mission\_Outcome" column and returns the number 101 which is a combined total between successes and failures.

# Boosters Carried Maximum Payload

---

- To find boosters that have carried the maximum payload use the query; select "Booster\_Version" from SPACEXTBL where PAYLOAD\_MASS\_\_KG\_ = (SELECT MAX(PAYLOAD\_MASS\_\_KG\_) FROM SPACEXTBL)
- This query uses a subquery to specifically find the values that carried the highest payload. The booster versions are as follows: 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

---

- To get the failed landings on a drone ship for 2015 and display the month you use the query; select substr(Date, 6,2), "Landing\_Outcome", "Booster\_Version", "Launch\_Site" from SPACEXTBL where "Landing\_Outcome" = 'Failure (drone ship)' and substr(Date,0,5)='2015'
- This query specifies that the only rows we are interested in include the landing outcome of “Failure (drone ship)” and from the year 2015. There are two rows displayed.

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

---

- To rank landing outcomes in a specific date range use the query; select count("Landing\_Outcome"), "Landing\_Outcome" from SPACEXTBL where "Date" between '2010-06-04' and '2017-03-20' group by "Landing\_Outcome" order by count("Landing\_Outcome") DESC
- This query will order the results from highest number of instances to lowest. It shows that no attempts are most common, followed by a tie in successes and failures to land on drone ships, followed again by a tie of success on a ground pad and controlled into the ocean. Next is uncontrolled into the ocean which ties with failure with parachute. Last is precluded drone ship.

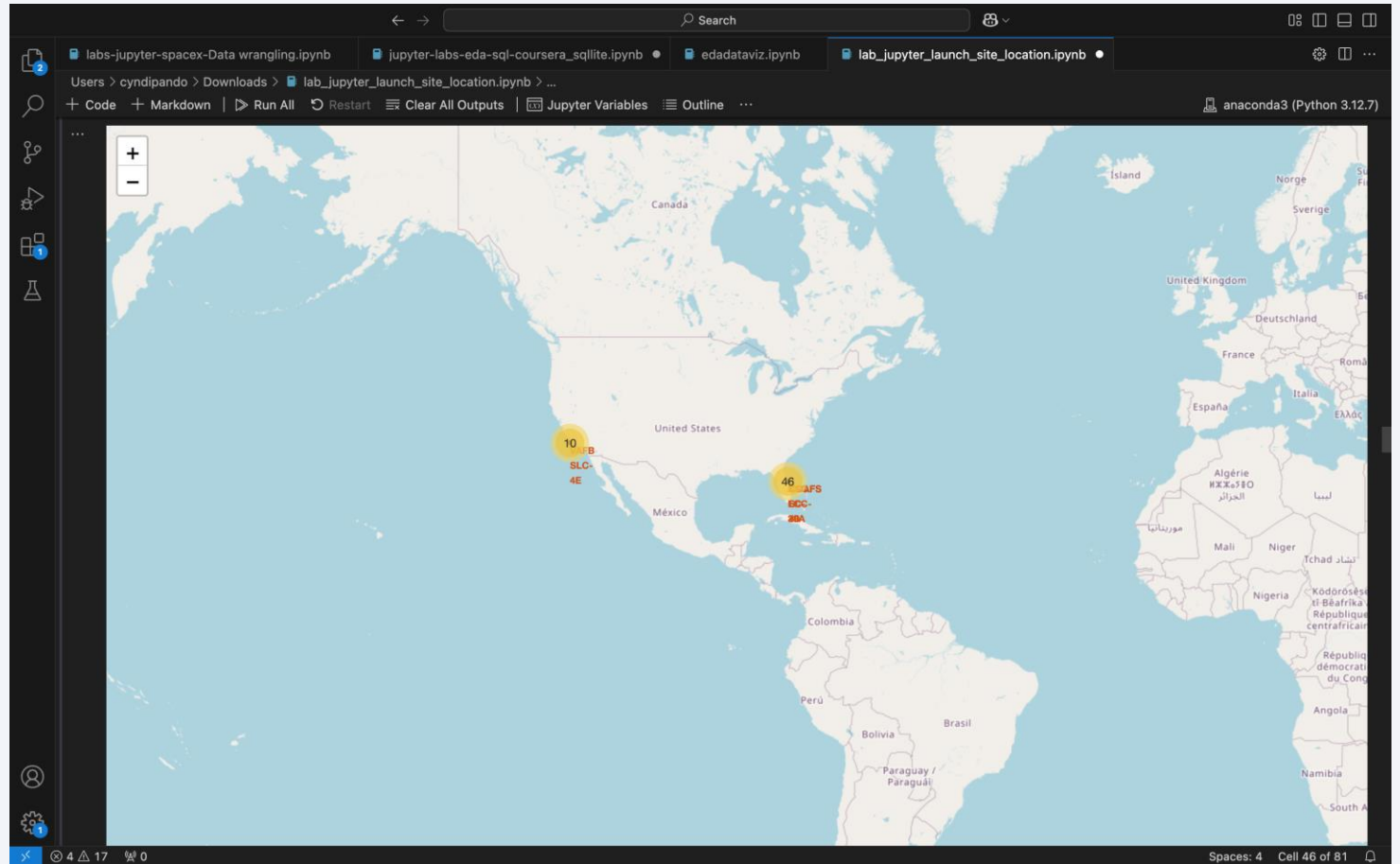
A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The image is a composite of a solid blue background on the left and a satellite photograph of Earth on the right. The Earth's surface is dark, with numerous bright yellow and orange lights representing cities and urban areas. The horizon of the Earth is visible as a thin, curved line separating the dark surface from the deep blue of space.

Section 3

# Launch Sites Proximities Analysis

# Global Position of launch sites

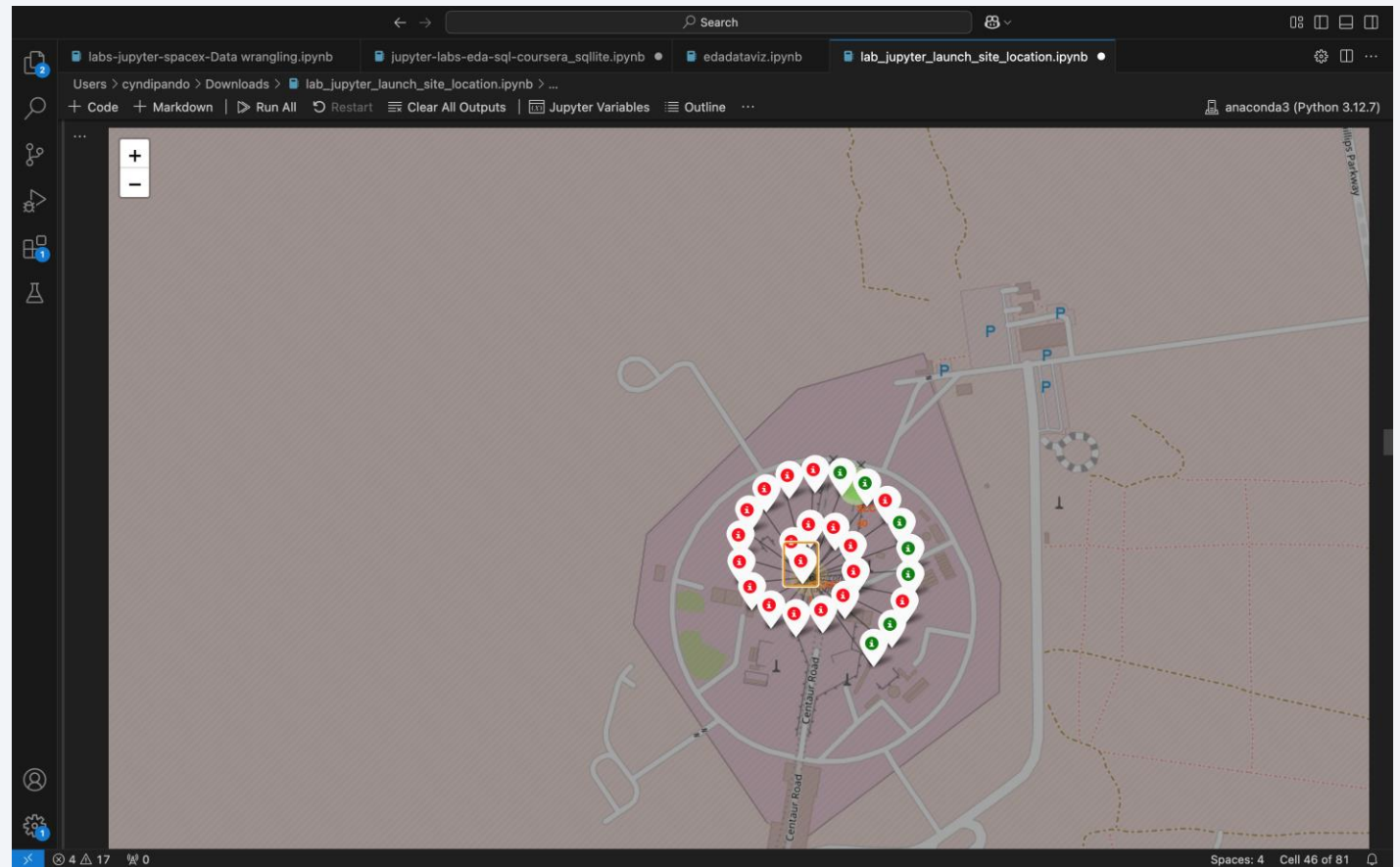
- This map displays the launch sites, which can be found closer to the coast, and their number of launches. By further hovering and zooming in more info can be discovered.



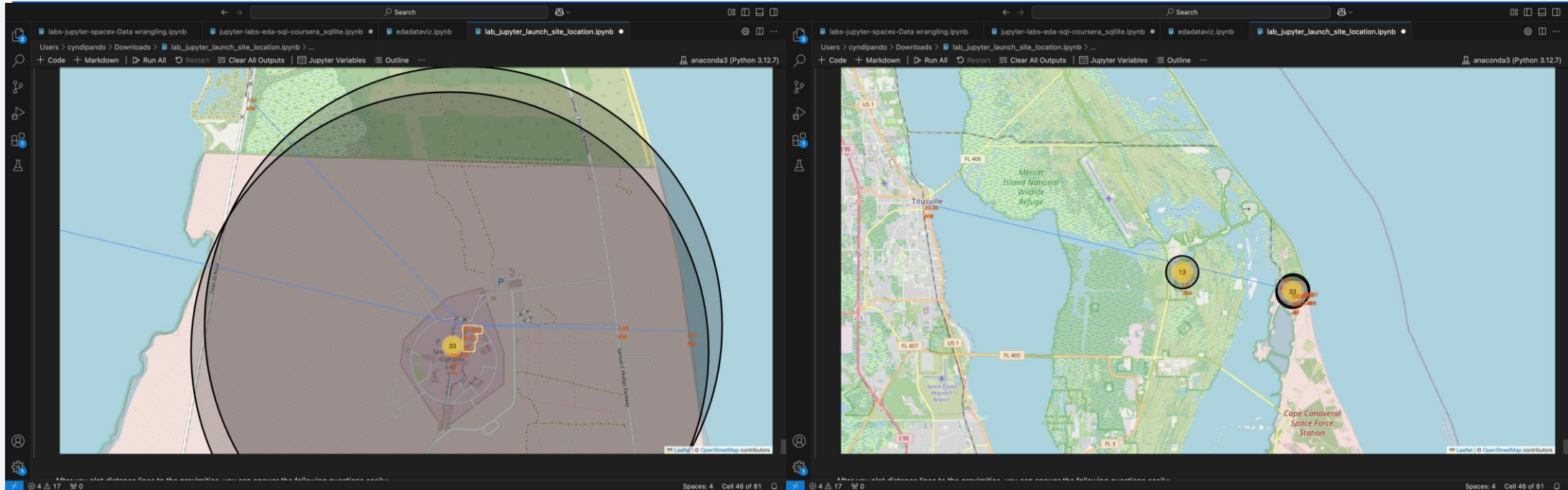


# Success/Failure markers

- This screenshot shows a marker cluster for a particular launch site, the successful stage 1 landing marked in green and the failures marked in red. This gives a visual representation of the success/failure ratio of each launch site.



# Proximity to infrastructure



- The above screenshots show the proximity of one launch site to the coast, nearby highways, nearby railroads, and in the second screenshot, nearby cities. As shown by the maps launch sites are in close proximity to railroads, coasts, and highways, but not very close to cities. This should be accounted for when choosing launch site locations.



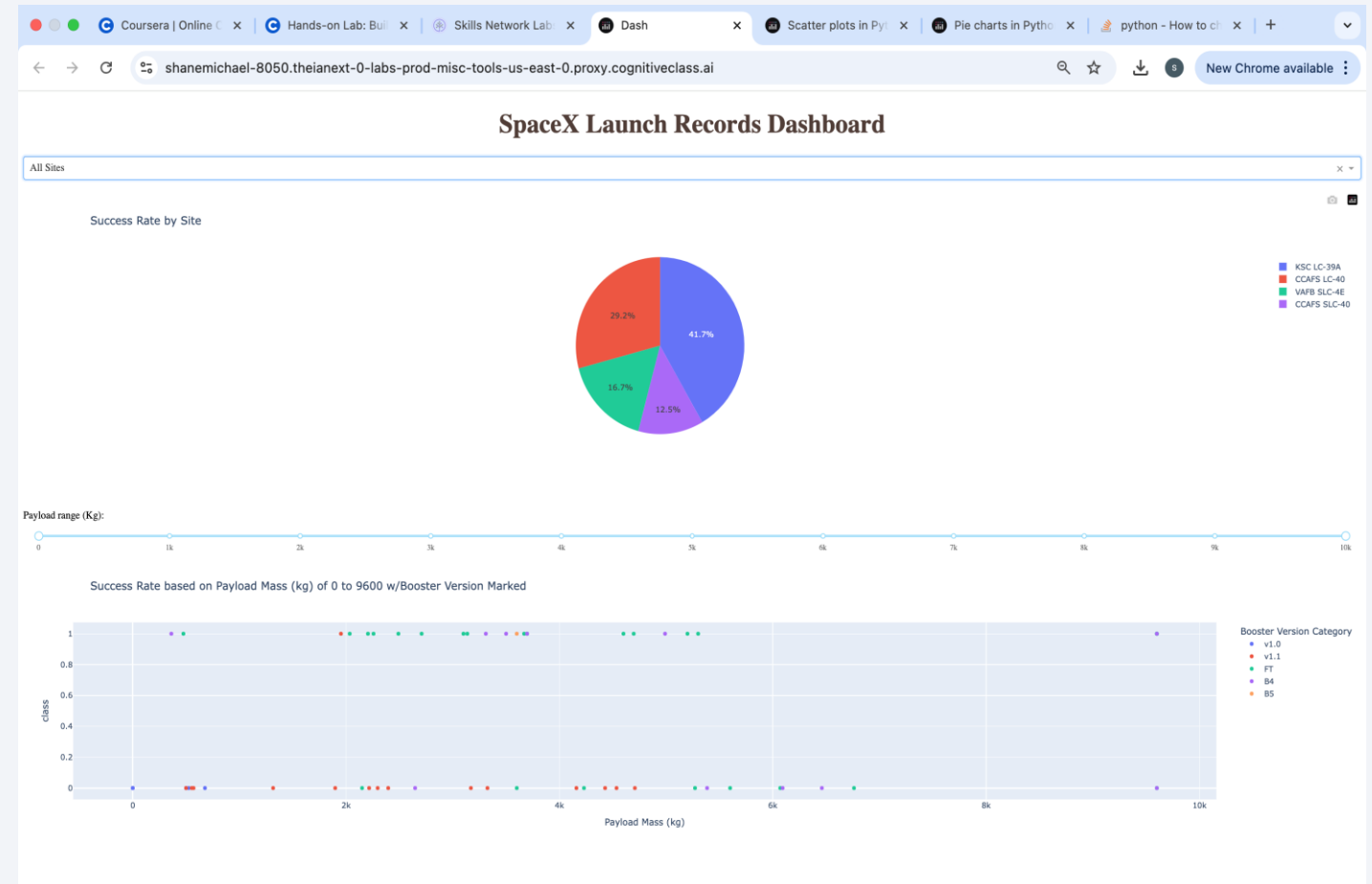


Section 4

# Build a Dashboard with Plotly Dash

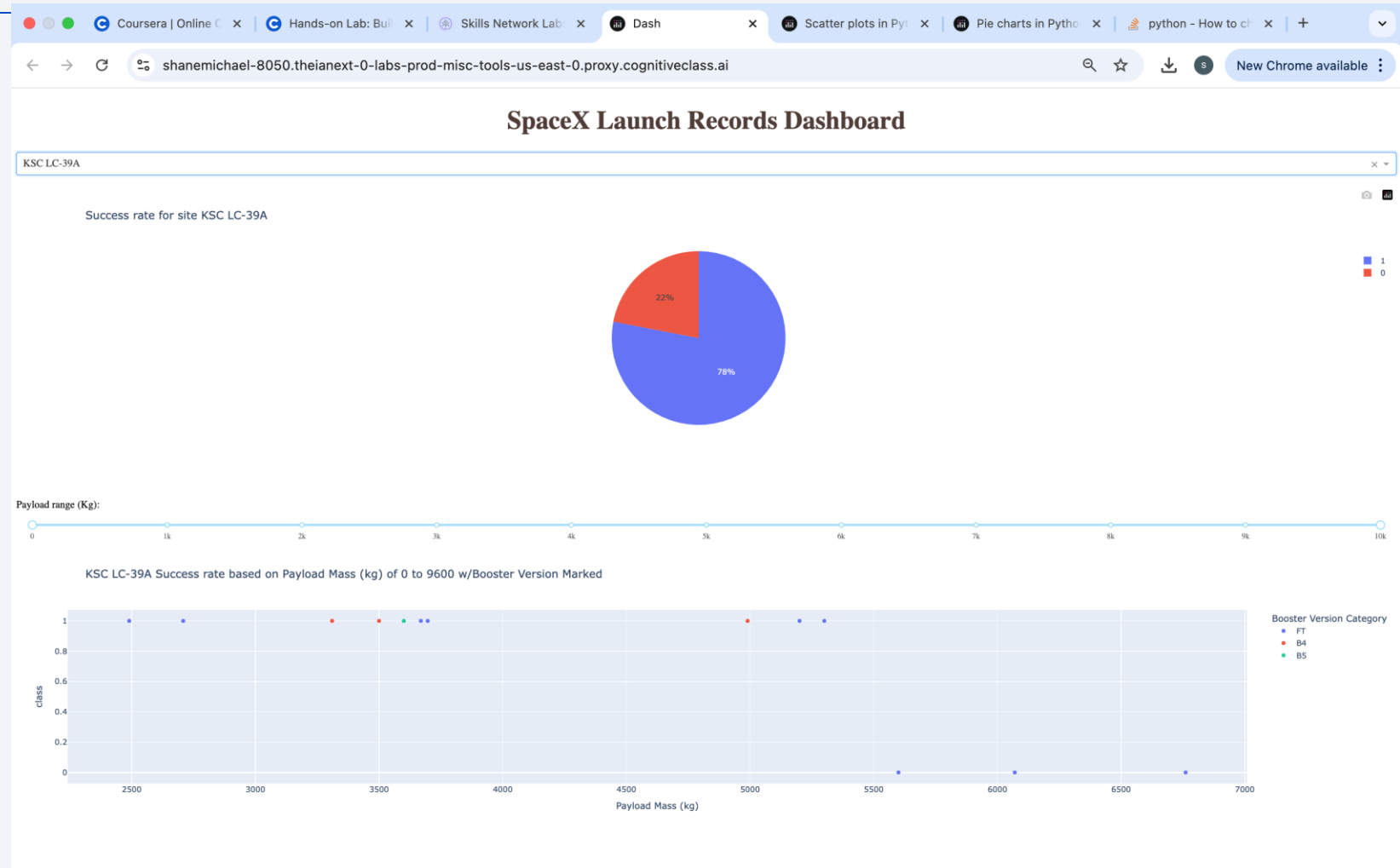
# Launch Sites proportion of success

- The accompanying pie chart shows the successes by site. Each slice is a proportion of the total successes that can be attributed to a specific launch site.



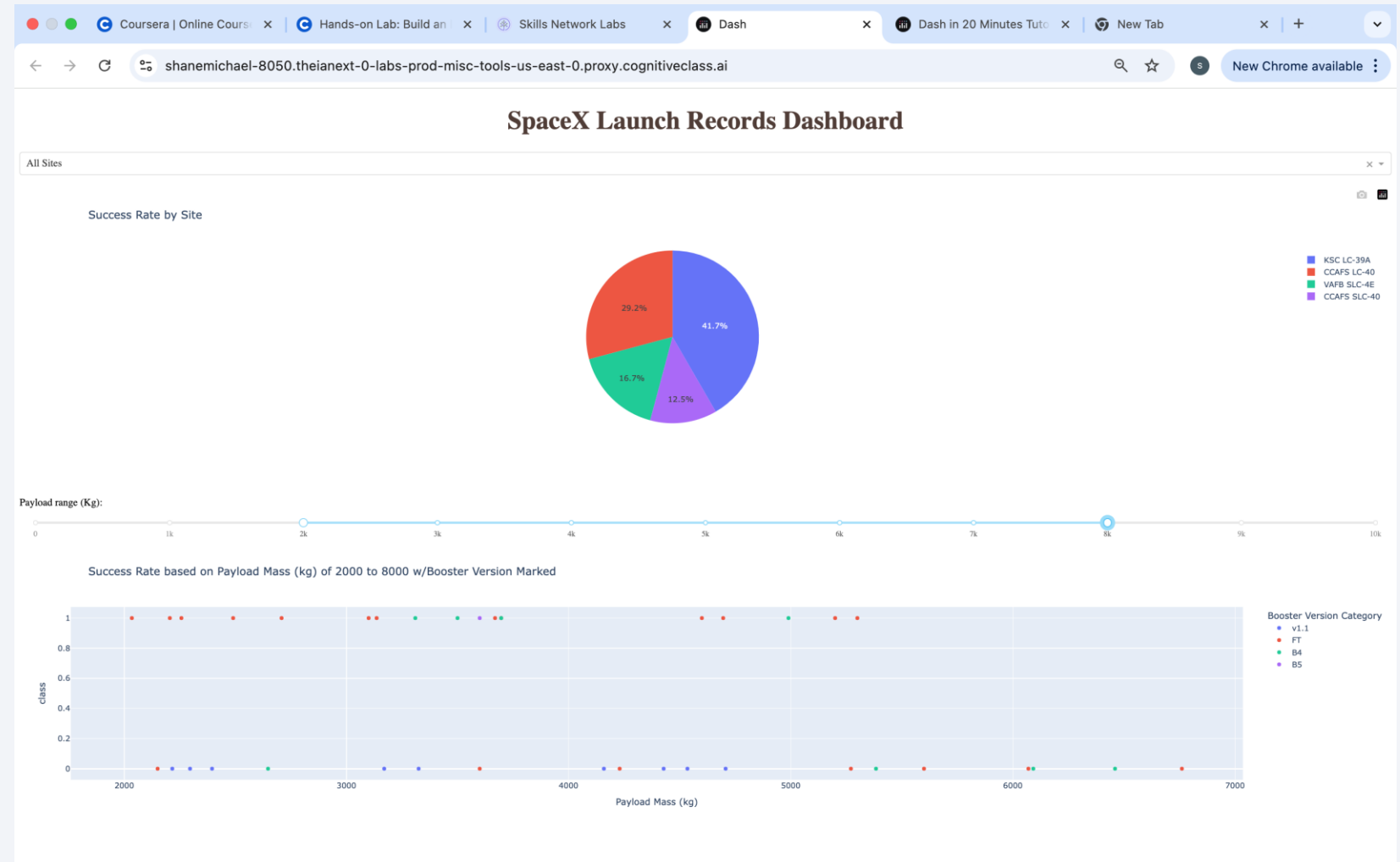
# Highest success rate Launch Site

- Shown in the accompanying graphic is the highest success rate launch site. Site KSC LC-39A had the highest rate of successful launches at 78%.



# Payload effect on Success rate

- The Screenshot included shows a range slider for payload adjusted to different ranges to hone in on differing payload success rates. It seems that as payload increases there is some decrease in successes. Some booster versions perform better than others at differing payloads. Such as FT performing very well with low to mid range payloads compared to the 1.1 which fails more often.





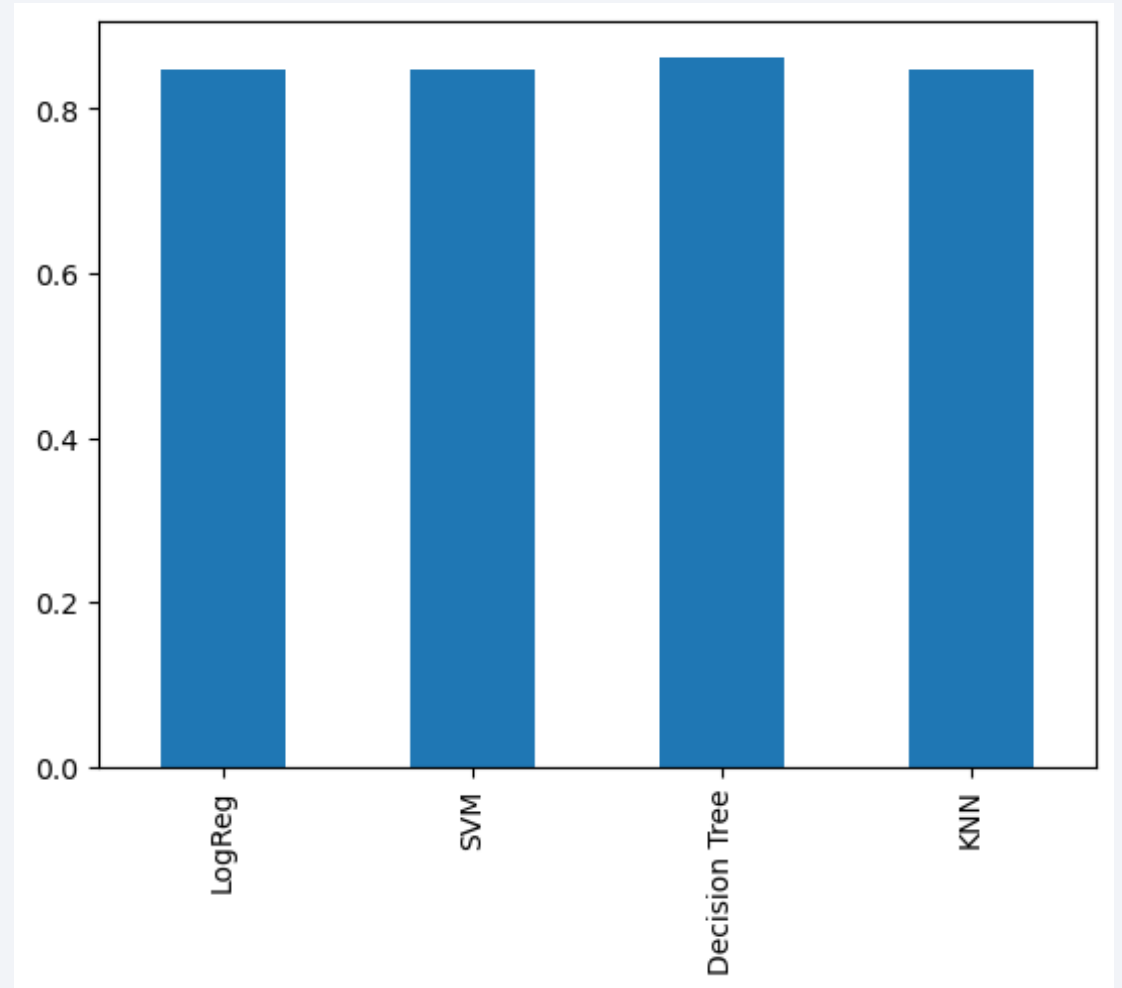
Section 5

# Predictive Analysis (Classification)



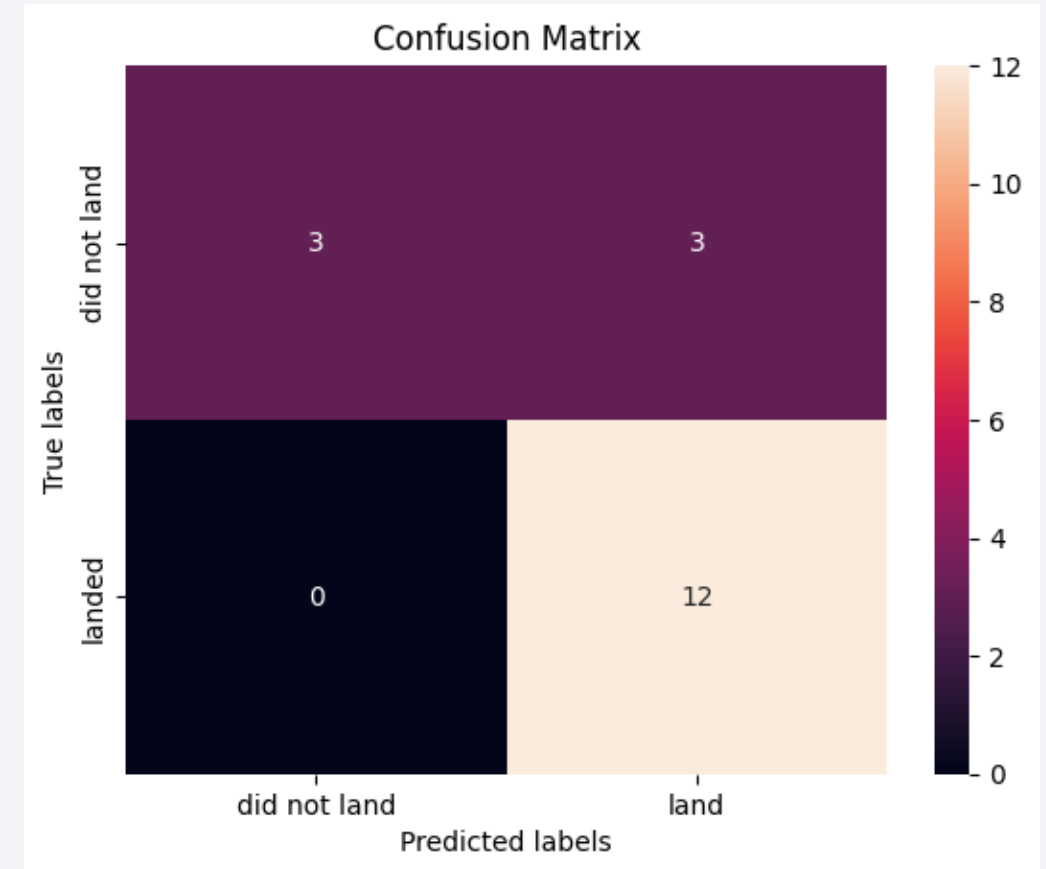
# Classification Accuracy

- From the predictive analysis four methods were chosen to attempt to find the best method for classification.
- The best score of the four models came from the decision tree which slightly outperformed the other three trees, however their .score was identical for each.



# Confusion Matrix

- All the confusion matrices are identical so I provided one. Overall they correctly predict 15 of the result and only mislabel 3 that did not land as would land.
- Although their confusion matrices are identical the decision tree ranked highest on the `.best_score_` compared to the other three.



# Conclusions

---

- The data showed many things, the Decision tree seems to be the most accurate although all four confusion matrices are identical.
- The locations proximity is important to keep in mind as it provides criteria for choosing potential launch site locations.
- Of the orbit types SSO has the highest confidence for success and GTO the lowest.

# Appendix

---

- To examine anything else please refer to my GITHUB repository
- <https://github.com/Shane-Michael-git/IBMDDataScienceCapstone.git>

Thank you!

