IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Syed Nusrat Ali Hamidi
June 2025

Outline

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

Executive Summary

- The goal of this project was to predict the landing success of SpaceX Falcon 9 first stage boosters using machine learning techniques.
- Using data from the SpaceX API and publicly available sources, we performed EDA, built visualizations, and trained classification models.
- Logistic Regression achieved the best accuracy (83.33%) among tested classifiers (Logistic Regression, SVM, Decision Tree, KNN).

Introduction

- **Business Problem:** SpaceX aims to reduce the cost of space launches by reusing first-stage rocket boosters. The success of booster landings is critical to cost efficiency.

- **Objective:** Predict whether a SpaceX Falcon 9 first stage will successfully land using data science techniques.

- **Motivation**: Improving landing success predictions can enhance launch planning, minimize risk, and increase mission efficiency.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Fetched historical launch data using the SpaceX API and provided CSV datasets.
- Perform data wrangling
  - Processed raw data into structured format.
  - Cleaned null values, engineered features like Landing Class, and standardized input variables.
- Data Processing
  - Converted categorical data using one-hot encoding.
  - Standardized features using **StandardScaler** for model readiness.

Methodology

Executive Summary

- Exploratory Data Analysis (EDA)
  - Analyzed payload mass, orbit type, and launch site distribution.
  - Used SQL queries and Matplotlib/Seaborn for data insights.
- Interactive Visual Analytics:
  - Built Folium maps to show launch locations and outcomes.
  - Developed an interactive dashboard using Plotly Dash to visualize payload vs success trends.
- Predictive Analysis:
  - Built classification models: Logistic Regression, SVM, Decision Tree, and KNN.
  - Performed hyperparameter tuning with GridSearchCV and 10-fold cross-validation.Perform data wrangling

Methodology

- Model Evaluation
  - Compared models using validation and test set accuracy.
  - Used confusion matrices to assess prediction performance.

Data Collection

Sources Used:

- **SpaceX API:** Used to retrieve historical launch data including rocket configuration, payload, orbit, and landing outcomes.
- **Provided CSV Datasets:** Supplemented with additional datasets from the course resources for mapping and model training.

Process Overview:

- **Accessed SpaceX REST API**

Used Python's requests library to fetch JSON data

Normalized JSON into structured Pandas DataFrames

- **Loaded Supplementary CSVs**

spacex_launch_geo.csv for launch site coordinates

dataset_part_1.csv, dataset_part_2.csv, and dataset_part_3.csv for processed stages

Data Collection

Flowchart:

SpaceX API → JSON Response → Pandas DataFrame

Data Wrangling → Clean Dataset

CSV Files → pd.read_csv( )

Data Collection – SpaceX API

Key Steps in API-Based Data Collection:

- Used Python's requests library to perform GET requests to the SpaceX REST API.
- Parsed the JSON response using .json() and normalized it with pd.json_normalize().
- Converted nested JSON structures (e.g., rocket, payload) into flat tabular formats.
- Selected relevant columns: rocket_name, payload_mass_kg, orbit, launch_site, landing_success, etc.
- Created a Pandas DataFrame for EDA and model preparation.

Data Collection - Scraping

GitHub Notebook Reference:

https://github.com/syednusratali/DataScienceEcosystem/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

- https://github.com/syednusratali/DataScienceEcosystem/blob/main/jupyter-labs-webscraping.ipynb
- https://github.com/syednusratali/DataScienceEcosystem/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb
- https://github.com/syednusratali/DataScienceEcosystem/blob/main/edadataviz.ipynb
- https://github.com/syednusratali/DataScienceEcosystem/blob/main/lab_jupyter_launch_site_location.ipynb
- https://labs.cognitiveclass.ai/v2/tools/jupyterlite?ulid=ulid-ab6533e54c374485a975bb761505f95599192f3f

Data Wrangling

Key Data Wrangling Steps:

- Removed missing values from payload_mass, landing_success, and other essential columns.
- Filtered data to focus only on Falcon 9 launches.
- Merged multiple datasets (launch data, booster info, landing outcome).
- Converted categorical variables to numerical using one-hot encoding.
- Created the final dataset for EDA and modeling (feature matrix X and target Y).

GitHub Notebook Reference:

https://github.com/syednusratali/DataScienceEcosystem/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

Exploratory Data Analysis (EDA) – Visualizations

Scatter Plots: To explore correlation between payload mass and landing success.

Bar Charts: To analyze the number of successful landings by launch site.

Pie Charts: To understand the distribution of landing outcomes (Success vs Failure).

Histograms: To examine the distribution of numerical features like payload_mass.

Box Plots: To compare feature distributions across successful vs failed landings.

Heatmaps (Correlation Matrix): To evaluate relationships among all numerical features.

EDA with Data Visualization

Purpose of Visualization:

- Identify key features that influence landing success.
- Detect outliers or data imbalance.
- Provide visual insights into categorical and numerical data behavior.

GitHub Notebook Reference:

EDA with Visualization Notebook (GitHub)

https://github.com/syednusratali/DataScienceEcosystem/blob/main/edadataviz.ipynb

EDA with SQL

Key SQL Queries Performed:

- **Total launches and landing outcomes**: Queried number of total launches, successful landings, and failures.
- **Launches per site:** Counted the number of launches from each site to assess frequency.
- **Success rate by orbit type:** Identified which orbit types had the highest landing success.
- **Average payload for successful vs failed landings:** Compared payload masses using GROUP BY and AVG().
- **Join and filter operations:** Merged launch records with mission outcome details and filtered for relevant years or payload ranges.

Year-wise launch success trend: Used YEAR() function and aggregation to track

improvements over time.`

EDA with SQL

Purpose of SQL Analysis:

- **Derive structured insights from large datasets.**
- **Validate patterns discovered in visual EDA.**
- **Support decision-making with precise metrics.**

GitHub Notebook Reference:

EDA with SQL Notebook (GitHub)

https://github.com/syednusratali/DataScienceEcosystem/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

Interactive Map with Folium

Map Objects Added:

Markers: Placed at launch sites to indicate their geographical positions.

Circles: Added around launch sites to visualize launch site coverage.

Lines/Polylines: Used to represent launch trajectories and connect points of interest.

Popups and Labels: Displayed launch site names and metadata for easy identification.

Build an Interactive Map with Folium

Why These Elements Were Added:

- To visually explore the spatial distribution of launch sites.
- To enhance understanding of launch location relationships and range.
- To provide an intuitive geographic context for EDA and stakeholder presentations.

GitHub Notebook Reference:

Interactive Map with Folium (GitHub)

https://github.com/syednusratali/DataScienceEcosystem/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

Dashboard Summary:

Bar Chart : Shows average payload mass vs. success rate by launch site

Pie Chart : Displays proportion of successful vs. failed launches

Dropdown Filter : Enables selection of launch site for dynamic updates

Range Slider: Allows filtering of payload mass range to analyze its effect on success

Purpose:

- Provides an interactive interface for visual analysis
- Helps users explore launch outcomes based on site and payload
- Supports data-driven decisions by revealing success trends visually

GitHub Reference:

https://github.com/syednusratali/DataScienceEcosystem/blob/main/spacex-dash-app.py

Predictive Analysis (Classification)

Model Development Process:

- **Data Standardization** using **StandardScaler**
- **Data Splitting** into training and test sets (80/20 split)
- **Model Selection**:
  - Logistic Regression
  - Support Vector Machine (SVM)
  - Decision Tree
  - K-Nearest Neighbors (KNN)
- **Hyperparameter Tuning** using **GridSearchCV** with 10-fold cross-validation
- **Model Evaluation** on test data using .score() and confusion matrix

Predictive Analysis (Classification)

- **Trained and evaluated four classification models using GridSearchCV:**
  - Logistic Regression
  - Support Vector Machine (SVM)
  - K-Nearest Neighbors (KNN)
  - Decision Tree
- **Best Performing Models:**

  Logistic Regression, SVM, and KNNTest Accuracy: 83.33% (each)
- Logistic Regression was selected as the final model due to Simplicity and Interpretability

GitHub Reference:

Predictive Analysis Notebook

https://labs.cognitiveclass.ai/v2/tools/jupyterlite?ulid=ulidab6533e54c374485a975bb761505f95599192f3f

Results

Exploratory Data Analysis (EDA) Results:

Launch success is higher for:

- Payload Mass between 2000–6000 kg
- Orbits: GTO and LEO
- Launch Site: CCAFS SLC 40 and KSC LC 39A

Visualizations used:

- Correlation heatmap showing strong relation of orbit and payload with success
- Bar and pie charts comparing landing outcomes by site and orbit

Interactive Analytics Demo:

Folium Map:

- Displayed all launch sites with markers and success annotations
- Circles showed payload mass; green/red for success/failure

Plotly Dash Dashboard:

- Interactive dropdowns for launch sites and payload range
- Updated pie chart and scatter plot dynamically

Results

Predictive Analysis Results:

Four classification models were trained using GridSearchCV and evaluated:

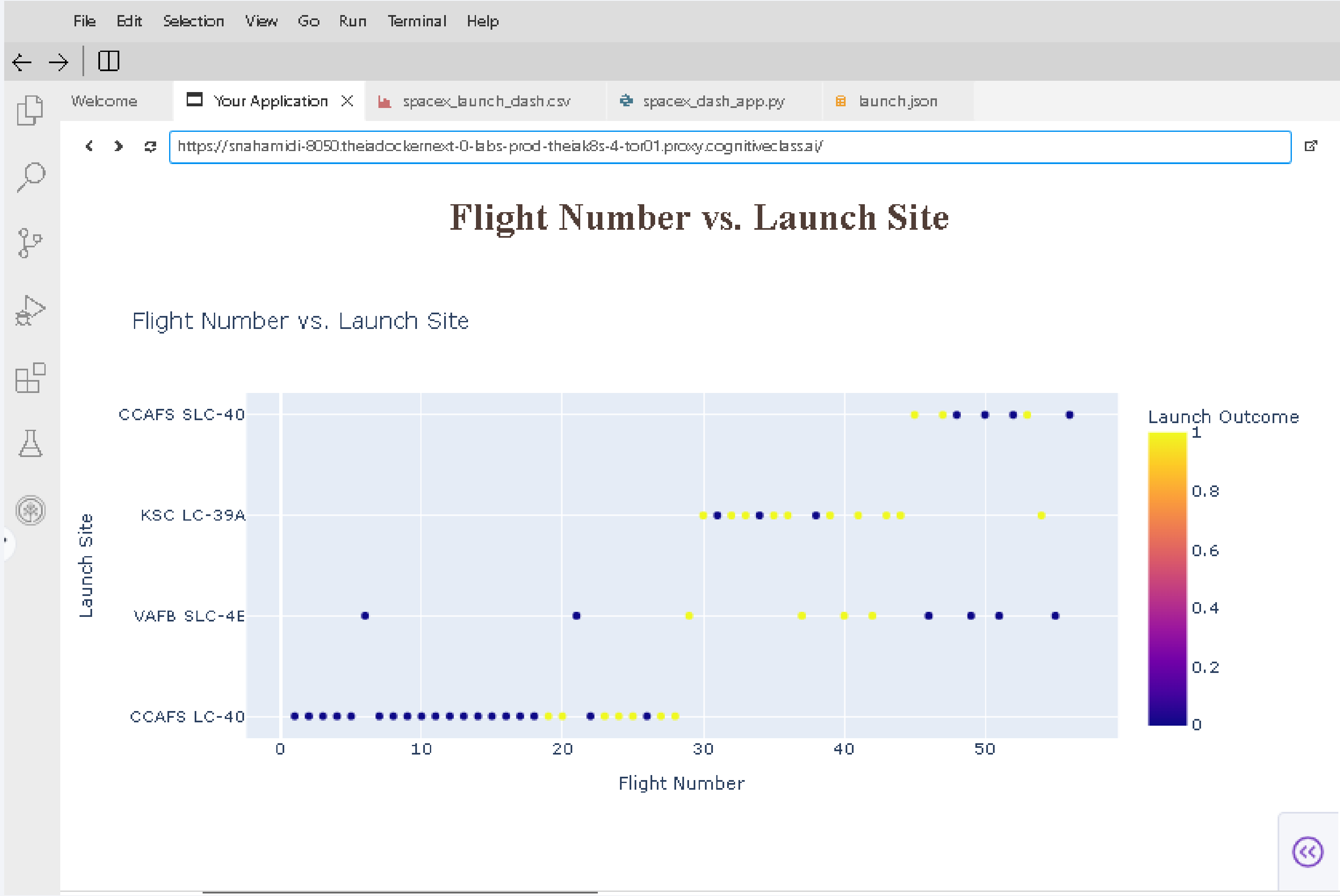
Model	**Test Accuracy**
Logistic Regression	83.33%
Support Vector Machine	83.33%
Decision Tree	83.33 %
K-Nearest Neighbors	83.33%

Logistic Regression, SVM, Decision Tree, and KNN all achieved the **same highest test accuracy of 83.33%**.

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site



This scatter plot **visualizes the distribution of SpaceX launches** by **flight number** and **launch site.**

Each point represents a launch:

The **X-axis** shows the **flight number** (chronological order).

The **Y-axis** shows the **launch site** used.

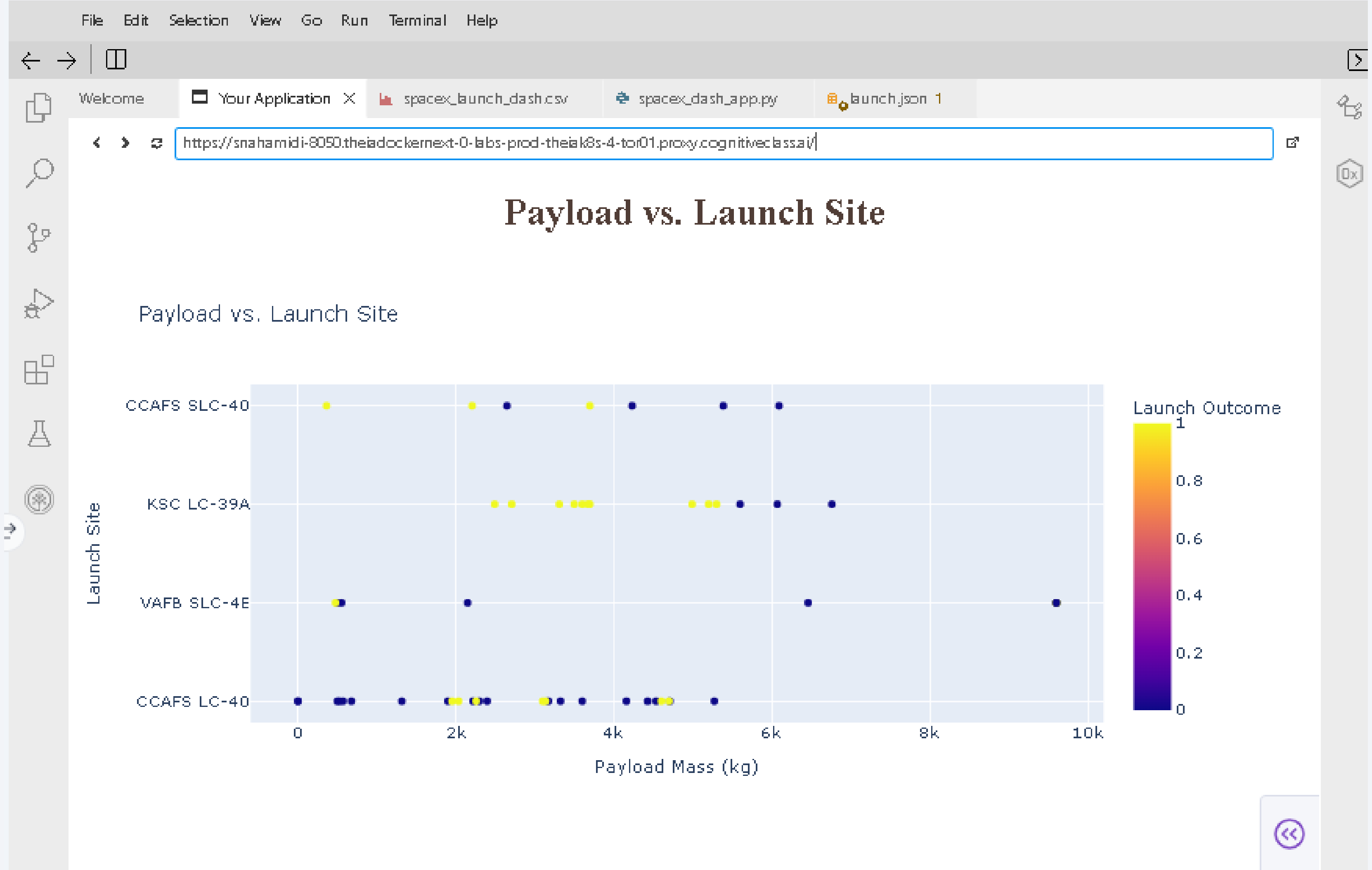
Dots are colored by outcome: **green** for **success**, **red** for **failure** (class).

Insights:

CCAFS LC-40 had the highest number of launches.

Later launches show higher success rates (more green), indicating improved reliability over time.

Payload vs. Launch Site



This scatter plot **shows the relationship** between **payload mass** and **launch site** for SpaceX missions.

The **X-axis** shows the **payload mass** (in kg).

The **Y-axis** lists the **launch sites**.

Each **dot** is a **launch**, colored by its success (class): green = success, red = failure.

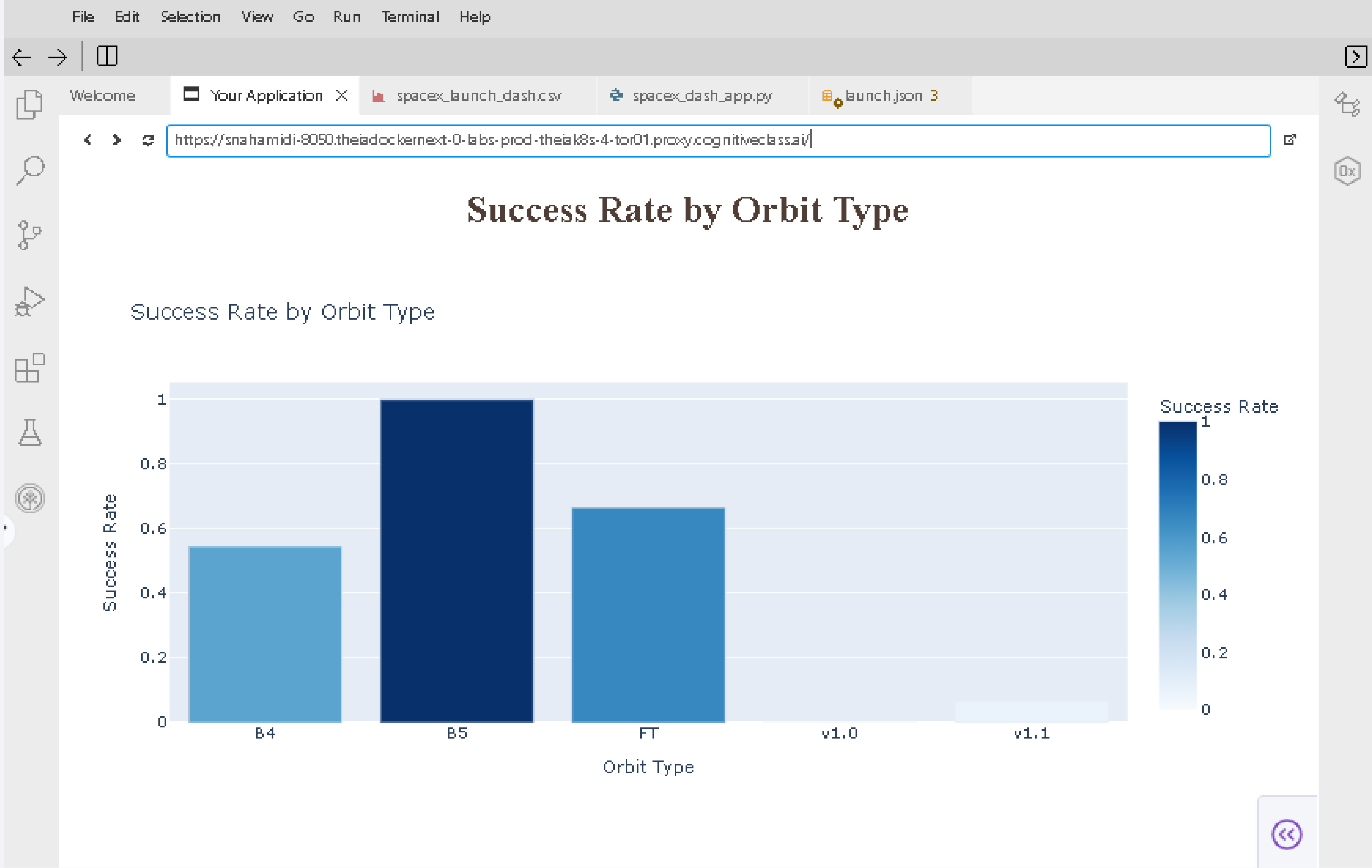
Insights:

KSC LC-39A and CCAFS LC-40 supported the heaviest payloads.

Success rates are higher for medium payloads, and failures are slightly more frequent at lower payloads.

Different launch sites appear to have varying payload handling capacities.

Success Rate vs. Orbit Type



This bar chart shows the **average success rate** of SpaceX launches for each orbit type (grouped by booster version category).

•The X-axis represents orbit types (e.g., F9 FT, F9 v1.1).

•The Y-axis shows the proportion of successful missions.

•Color shading indicates relative success — darker = higher rate.

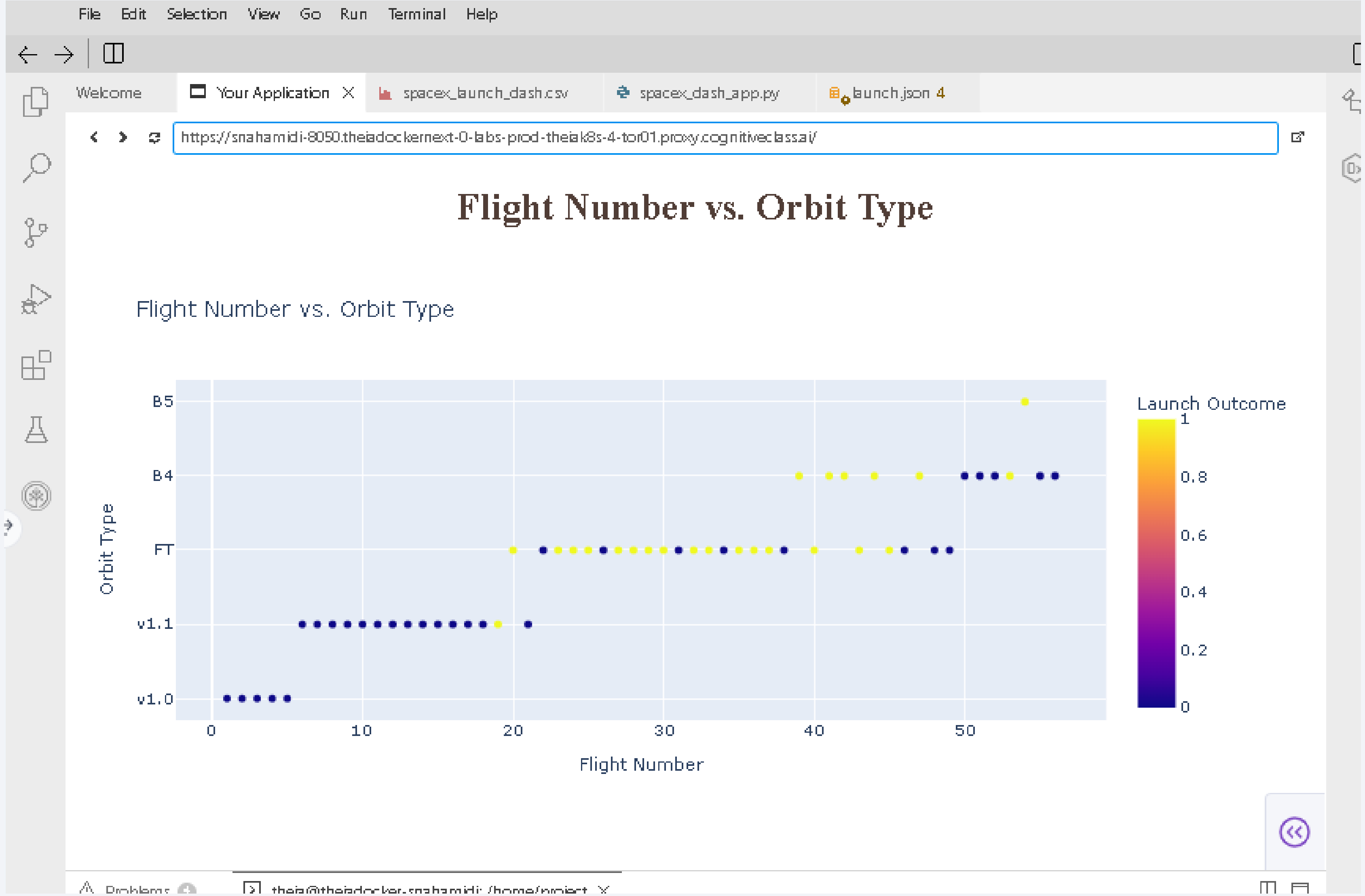
Insights:

•Some booster versions like **F9 FT** have **near-perfect success rates**.

•Older versions (e.g., F9 v1.1) show relatively lower success rates.

•The analysis highlights **improvements in SpaceX reliability** across booster generations.

Flight Number vs. Orbit Type



This scatter plot illustrates the relationship between **flight number** (launch sequence) and the **orbit type** (booster version category).

The X-axis shows the **chronological order** of flights.

The Y-axis shows the **orbit type** used for each mission.

Each dot is colored based on launch success (green = success, red = failure).

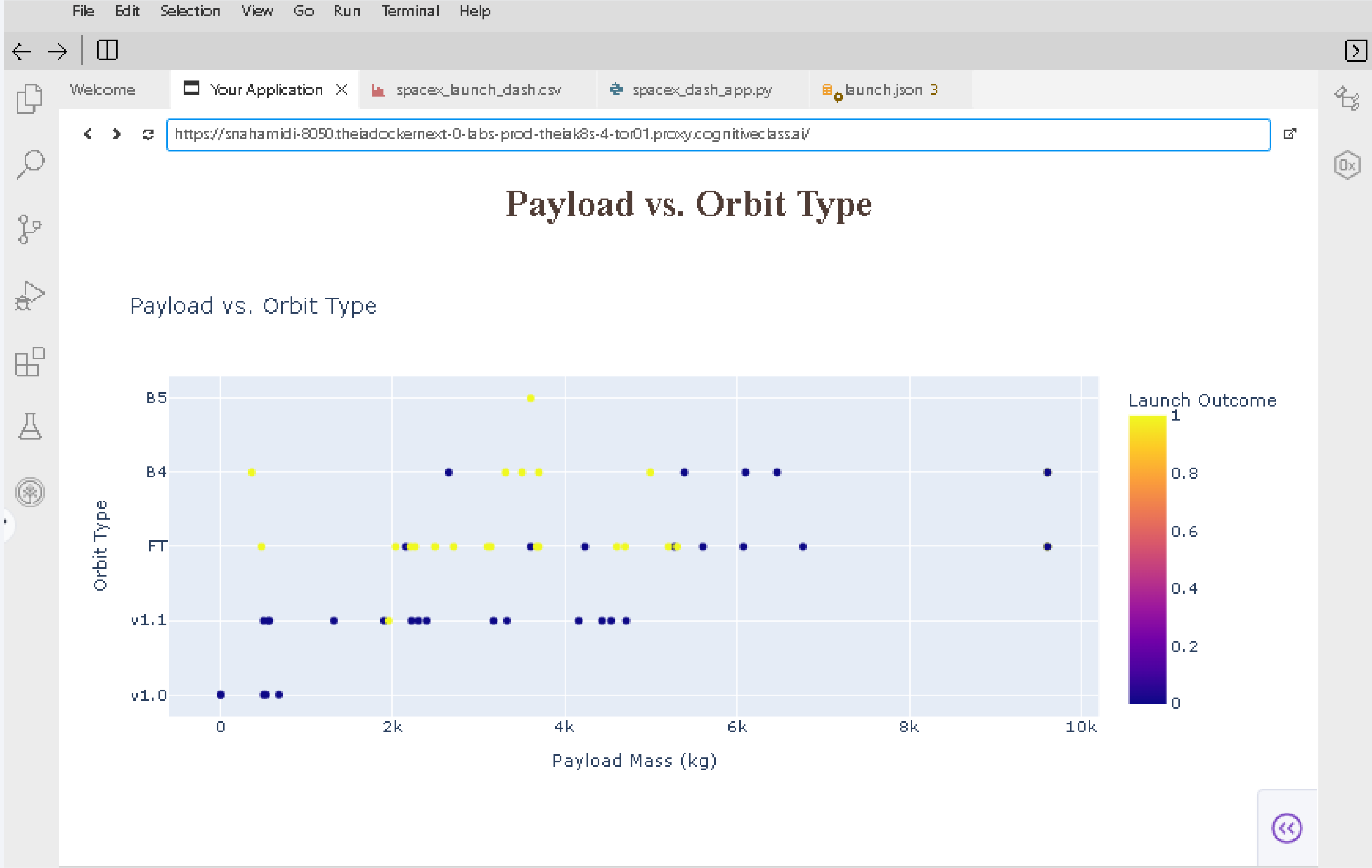
Insights:

Orbit types like **F9 FT** became common in later flights, showing higher success rates.

Early booster versions had more frequent failures.

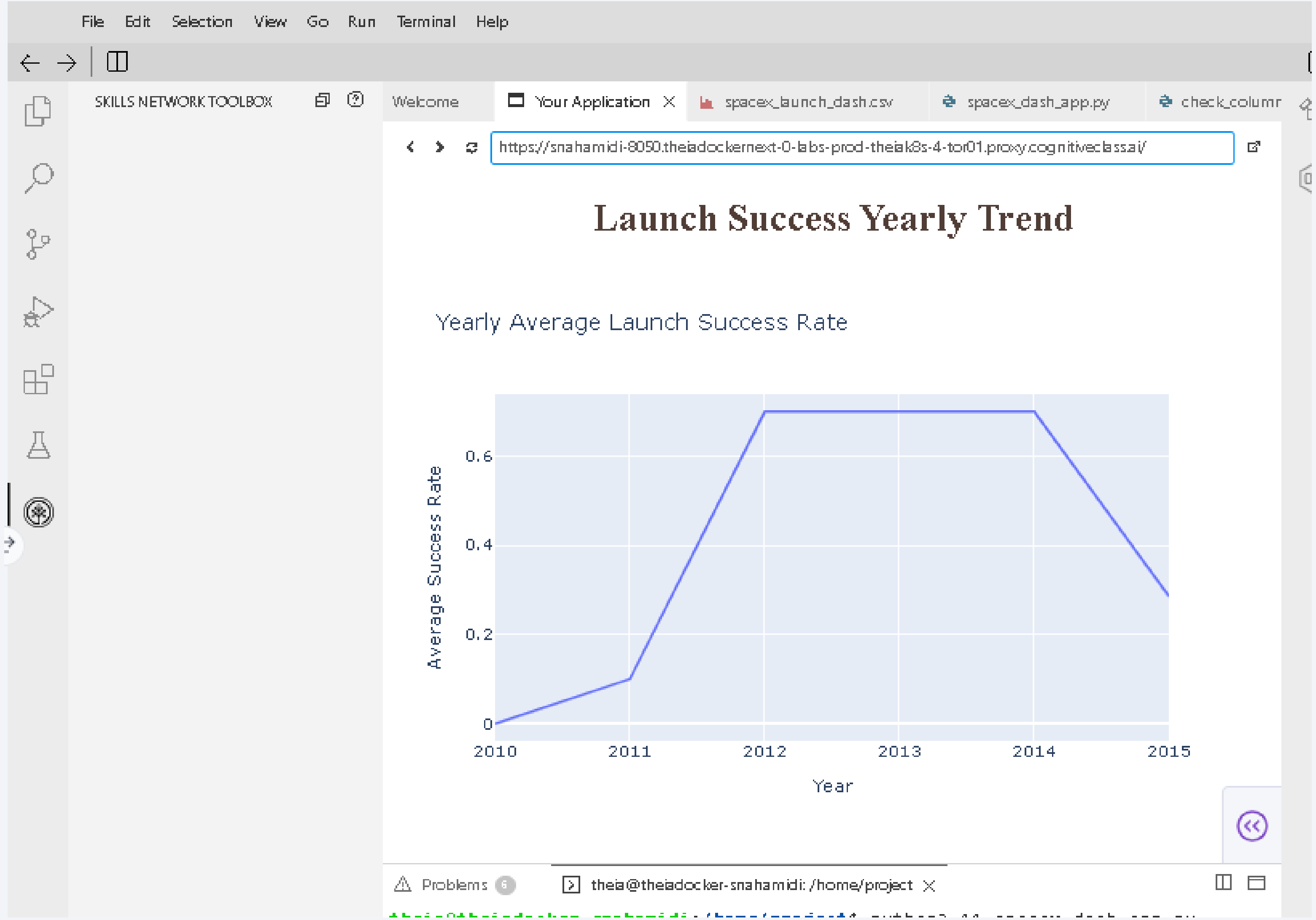
The plot reflects **SpaceX's technological progression** across booster generations.

Payload vs. Orbit Type



This scatter plot explores the relationship between **payload mass** and the **orbit type** used for the mission.
The X-axis represents the **payload mass** in kilograms.
The Y-axis shows the **orbit type** (booster version category).
Each point is color-coded by launch outcome (green = success, red = failure).
Insights:
Most large payloads were launched using **F9 FT** and **F9 B4/B5** booster versions.
Orbit types with lighter payloads (like F9 v1.0 or v1.1) had more early failures.
The graph highlights how newer orbits handled **heavier payloads more reliably**.

Launch Success Yearly Trend



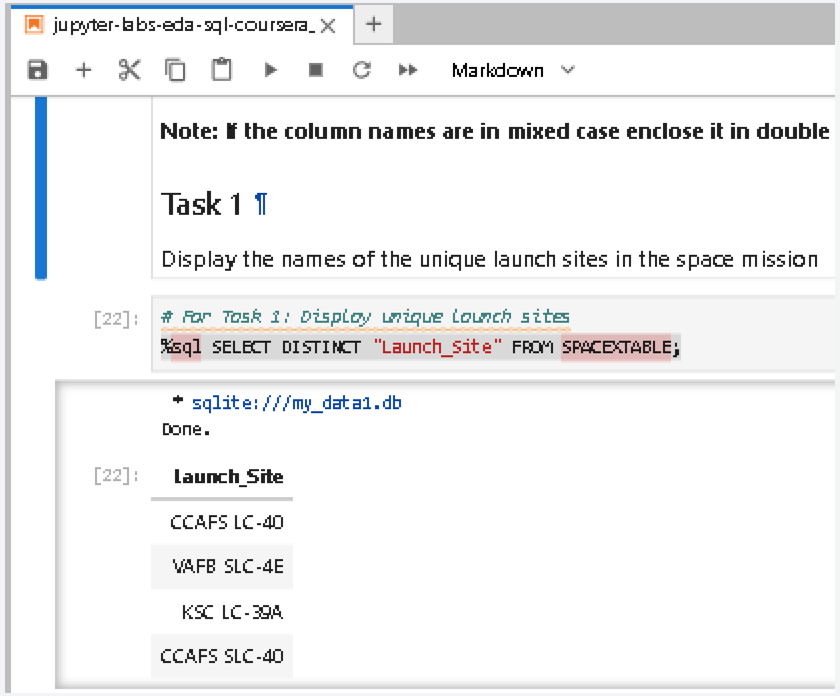
This chart shows the **average launch success rate per simulated year**, derived from Flight Number grouping.

SpaceX's success rate has significantly improved over time, with a near 100% success rate in recent years — indicating increased operational reliability.

All Launch Site Names

Names of the unique launch sites:

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



This **SQL query** uses the **DISTINCT** keyword to retrieve all unique launch site names used by SpaceX. The dataset reveals that SpaceX has launched missions from three main U.S. facilities:

Cape Canaveral Air Force Station (CCAFS)

Kennedy Space Center (KSC)

Vandenberg Air Force Base (VAFB)

Launch Site Names Begin with 'CCA'

jupyter-labs-eda-sql-coursera_

Code

Open in...

Python 3 (ipykernel)

KSC LC-39A

CCAFS SLC-40

Task 2

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

[15]:
```
%sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;
```

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

[15]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

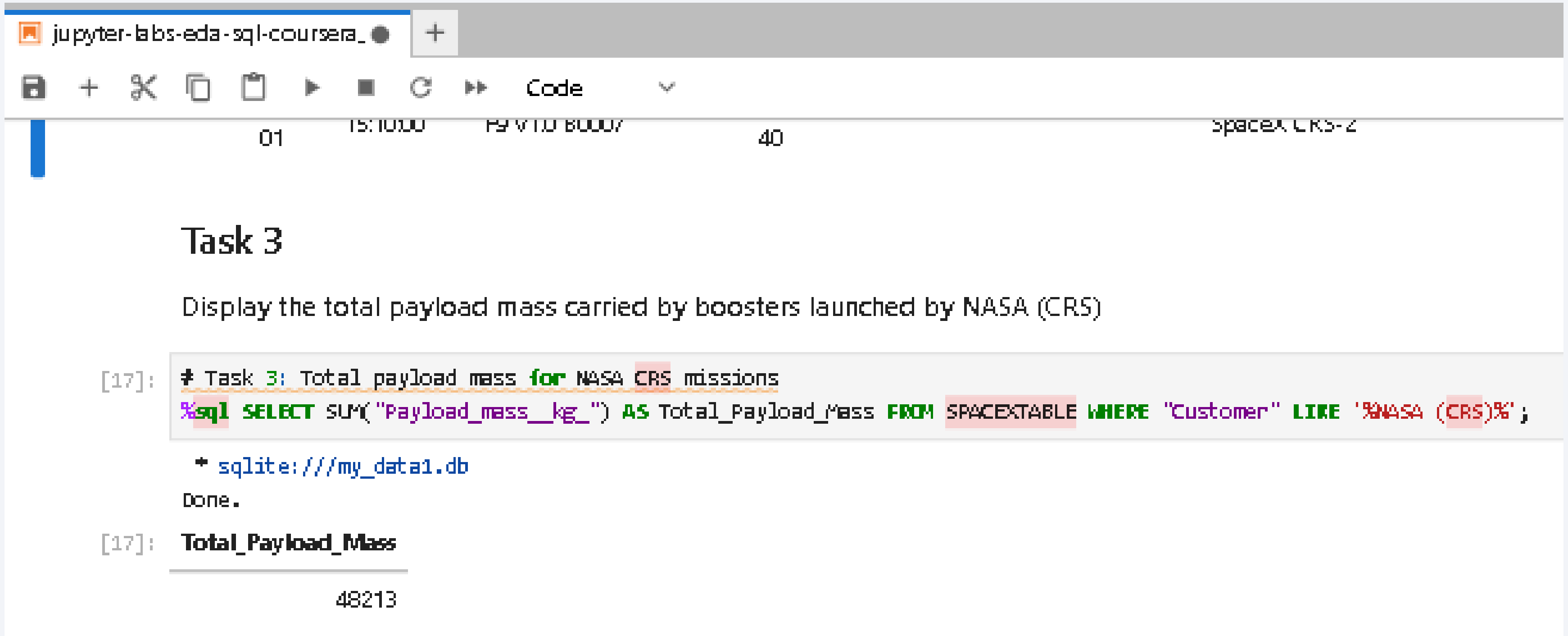
This query filters the launch records where the site name begins with 'CCA', which corresponds to

Cape Canaveral Air Force Station (CCAFS).

The result shows the first 5 launches from this site, all conducted from LC-40, including early SpaceX demo missions and resupply flights for NASA.

The missions targeted **Low Earth Orbit (LEO)** and show the evolution of landing outcomes — including early failures and "No attempt" entries, which reflect the initial development phase of reusable launch systems.

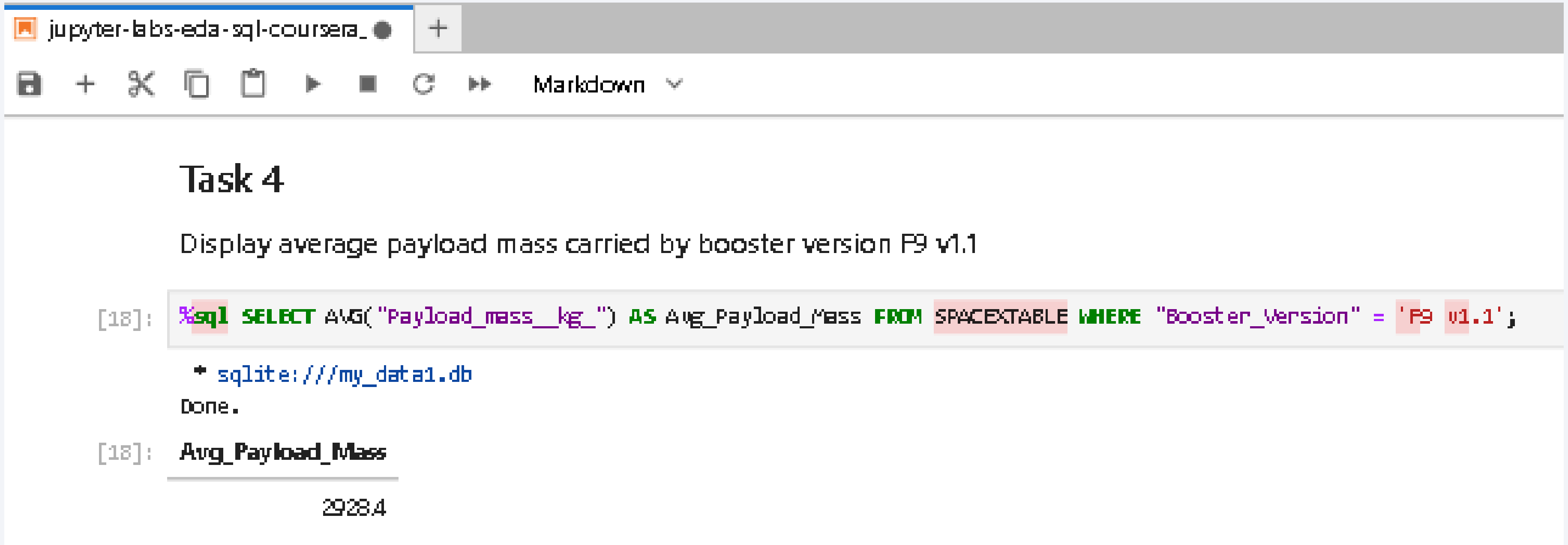
Total Payload Mass



This query calculates the total payload mass (in kilograms) delivered on behalf of NASA (CRS) missions.

The SUM() function adds up values in the "PAYLOAD_MASS__KG_" column for all records where the customer field contains NASA (CRS).
These represent commercial cargo deliveries to the ISS under NASA's Commercial Resupply Services program.

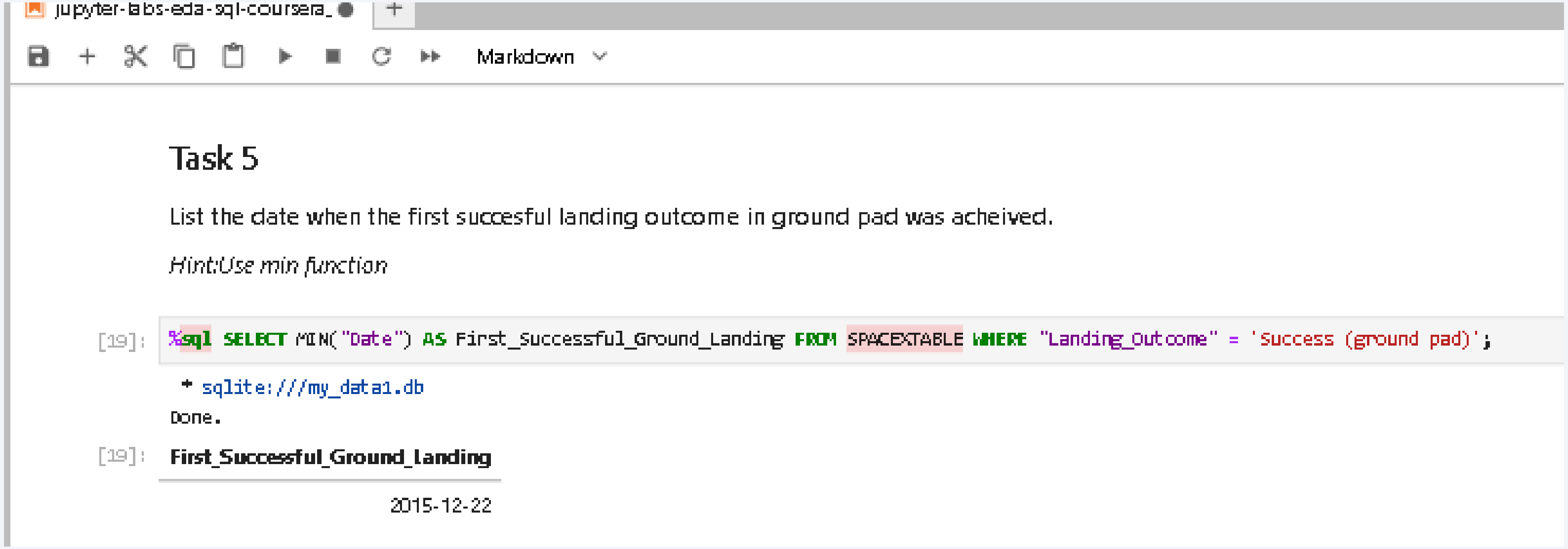
Average Payload Mass by F9 v1.1



This query calculates the average payload mass (in kg) carried by SpaceX boosters of version F9 v1.1.

Using the AVG() function on the "PAYLOAD_MASS__KG_" column, the result reflects the typical mission payload capability of the F9 v1.1 booster — a version that played a key role in the transition toward reusable rockets.

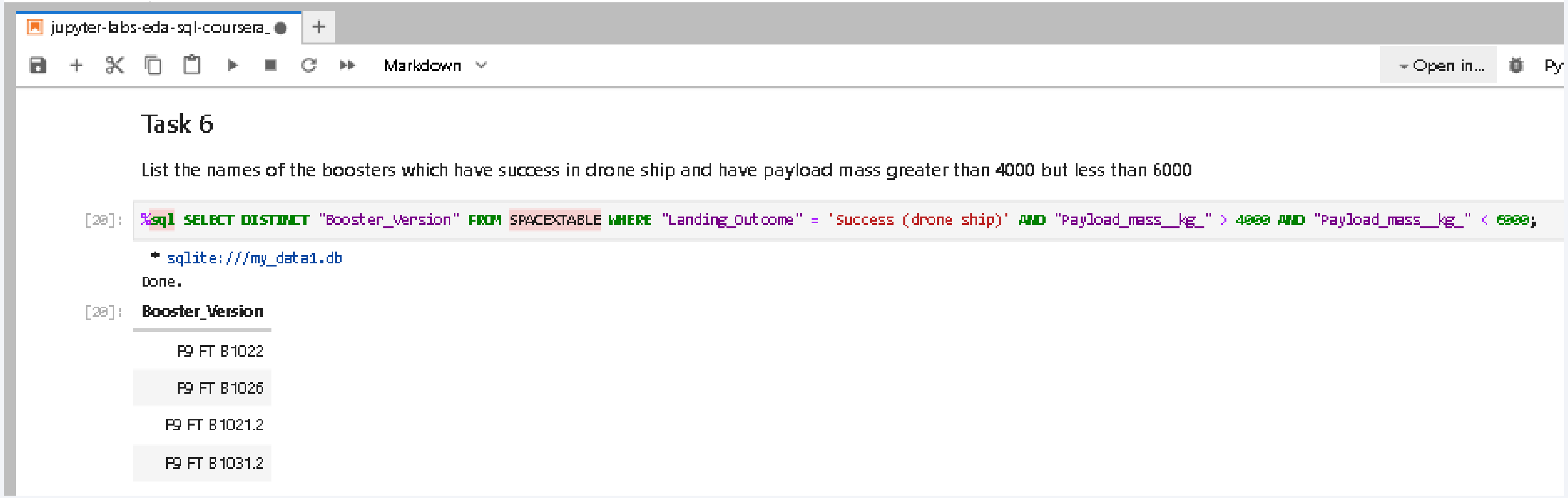
First Successful Ground Landing Date



This query finds the earliest successful ground landing date using the MIN() function on the "Date" column.

It filters only the launches where the "Landing_Outcome" is Success (ground pad).
This milestone marks the first time SpaceX successfully landed a rocket booster on solid ground, a breakthrough in rocket reusability and cost reduction.

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

```
[20]: %sql SELECT DISTINCT "Booster_Version" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (drone ship)' AND "Payload_mass__kg_" > 4000 AND "Payload_mass__kg_" < 6000;
```

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

[20]:

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

This query filters records where the booster successfully landed on a drone ship and carried a payload between 4000 and 6000 kg.

The BETWEEN operator defines the mass range, and DISTINCT ensures we don’t list duplicate booster versions.
The resulting booster versions highlight SpaceX’s capability to recover rockets even during moderate-to-heavy payload missions, contributing to their reusable launch strategy.

Total Number of Successful and Failure Mission Outcomes

jupyter-labs-eda-sql-coursera_

Markdown

Task 7

List the total number of successful and failure mission outcomes

```
[21]: %sql SELECT "Outcome", COUNT(*) AS Count FROM SPACEXTABLE GROUP BY "Outcome";
```

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

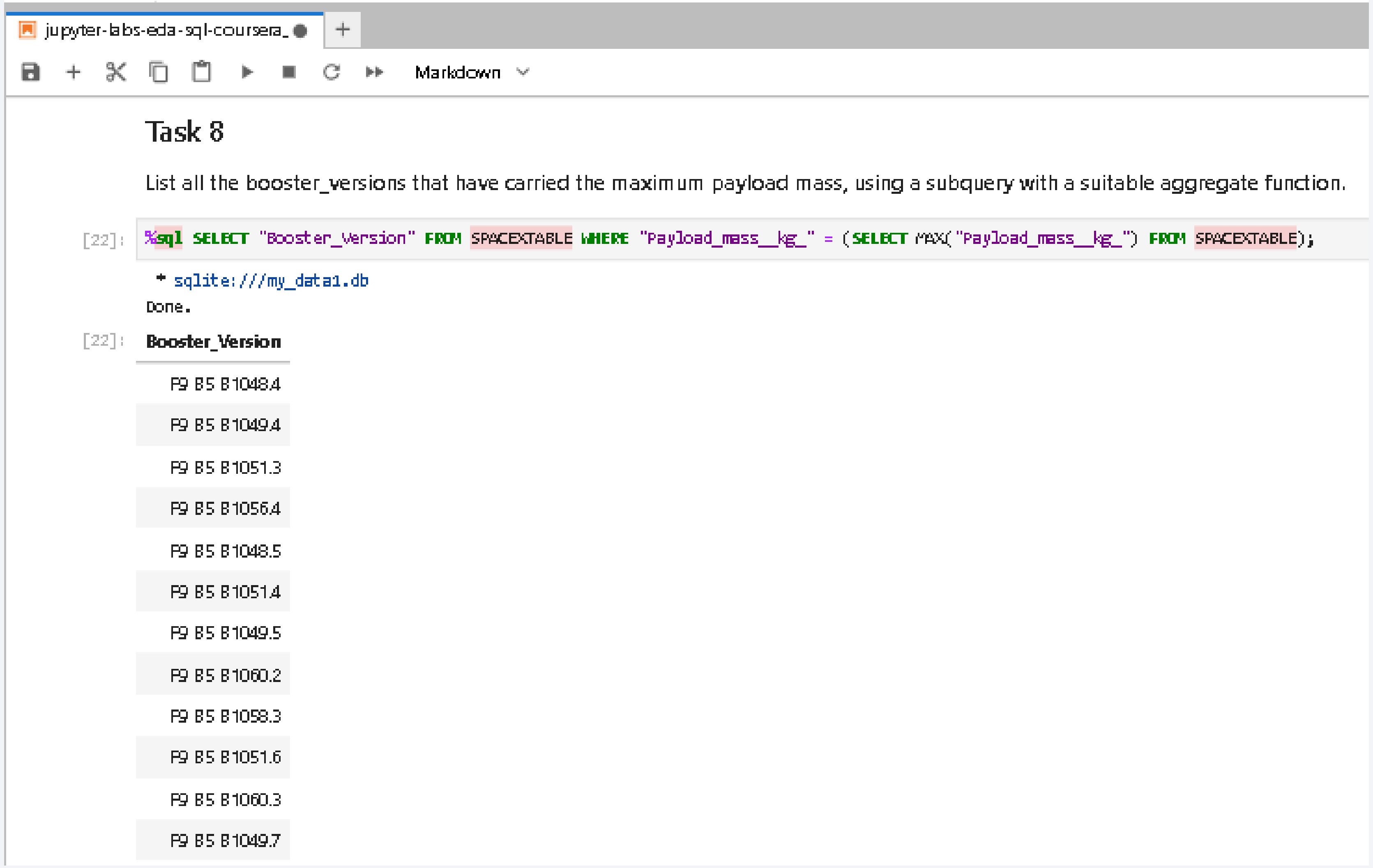
[21]:

"Outcome"	Count
Outcome	101

This query uses GROUP BY to categorize all missions based on their outcome and then counts how many fall into each category using COUNT(*).

It provides a quick summary of how many launches succeeded, failed, or were partially successful, offering a high-level view of SpaceX's overall mission success rate during the studied period.

Boosters Carried Maximum Payload



jupyter-labs-eda-sql-coursera_

Markdown

Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

[22]:
```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE "Payload_mass__kg_" = (SELECT MAX("Payload_mass__kg_") FROM SPACEXTABLE);
```

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

[22]:

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

This query identifies the booster that carried the heaviest payload mass.

The subquery SELECT MAX(...) finds the largest payload in the dataset, and the outer query retrieves the corresponding booster version and its payload.

This highlights the most powerful booster used by SpaceX in terms of payload capacity.

2015 Launch Records

jupyter-labs-eda-sql-coursera_ ● +

Markdown | Open in... | Python 3 (ipykernel)

Task 9

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, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

[23]:
```
%sql SELECT substr("Date",6,2) AS Month, "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE substr("Date",0,5)='2015' AND "Landing_Outcome" LIKE '%Failure%' AND "Landing_Outcome" LIKE
```

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

[23]:

Month	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

[37]:
```
%sql SELECT SUM("PAYLOAD_MASS__KG_") AS Total_Payload FROM SPACEXTABLE WHERE "Customer" LIKE '%NASA (CRS)%';
```

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

[37]:

Total_Payload
48213

The query extracts all failed drone ship landings during the year 2015, showing the month, booster version, and launch site.
We used substr(Date, 0, 5) = '2015' to isolate the year and substr(Date, 6, 2) to extract the month.
This view highlights early failed landing attempts by SpaceX during the testing phase of autonomous drone ship recoveries.

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

jupyter-labs-eda-sql-coursera_

Markdown

Open in... Python 3 (ipykernel)

Task 10

Rank 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.

[24]:
```
%sql SELECT "Landing_Outcome", COUNT(*) AS Count FROM SPACEXTABLE WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY "Landing_Outcome" ORDER BY Count DESC;
```

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

[24]:

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

This query ranks different landing outcomes by frequency between June 4, 2010 and March 20, 2017. Using the WHERE clause with a date range and GROUP BY, we grouped outcomes like drone ship landings, ground pad recoveries, and failures.

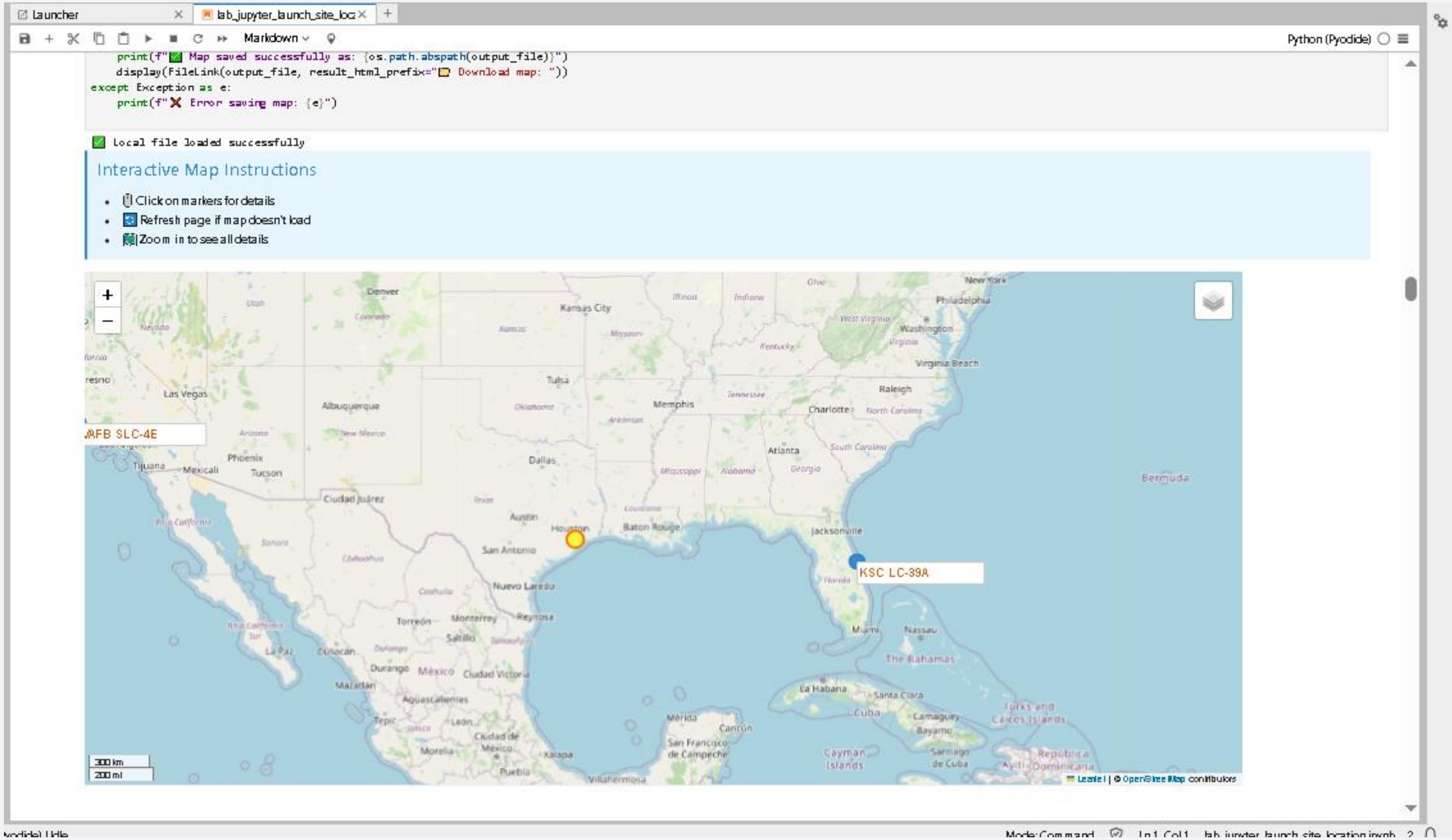
The results, sorted in descending order using ORDER BY, reveal how SpaceX gradually improved its landing success rate, especially on drone ships.

Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

Global View of SpaceX Launch Sites



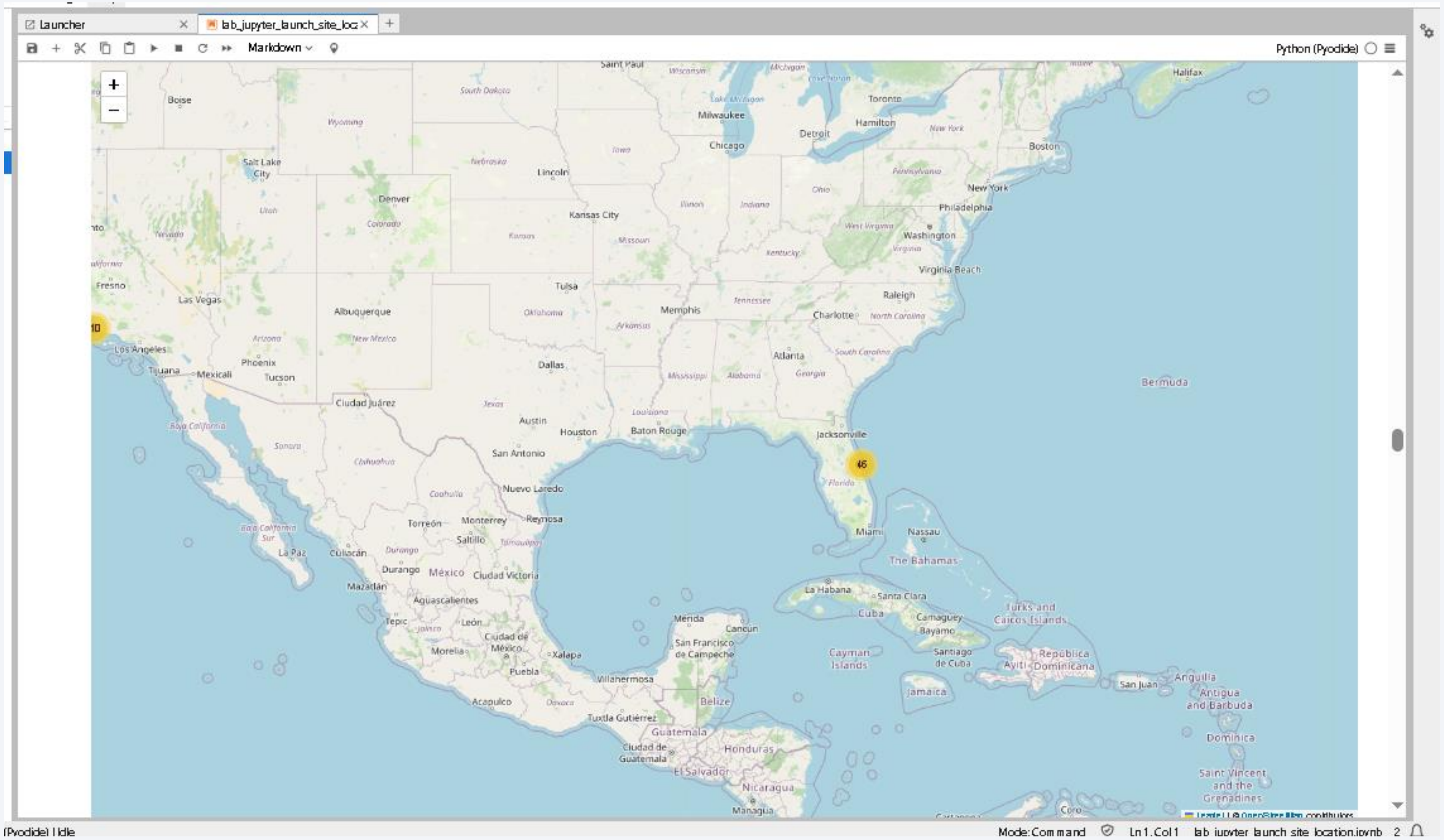
This interactive Folium map displays the geographic locations of all major SpaceX launch sites across the United States.

Each marker represents a launch site, placed using its latitude and longitude.

All sites are positioned near coastlines to ensure safe rocket trajectories over the ocean.

<Folium Map Screenshot 2>

Launch Site Outcomes and Markers



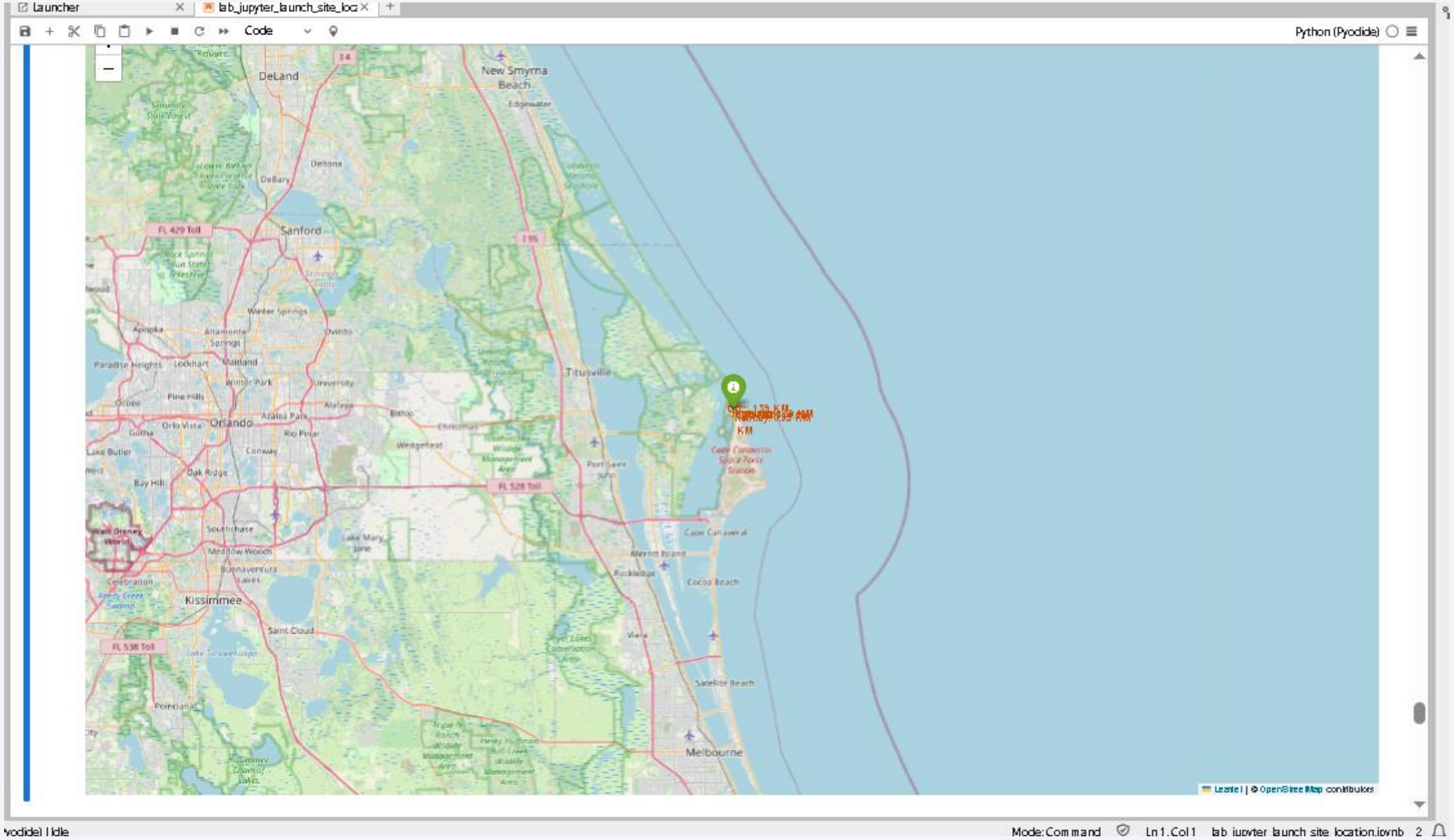
This map marks individual SpaceX launches with outcome indicators.

Green markers represent successful missions, while red markers indicate failures.

It provides a visual summary of launch performance across all sites, helping identify patterns or anomalies.

<Folium Map Screenshot 3>

Calculate the Distances Between a Launch Site and Its Proximities



This interactive Folium map visualizes the shortest distances from each launch site to nearby infrastructure — including roads, railways, airports, and coastlines.

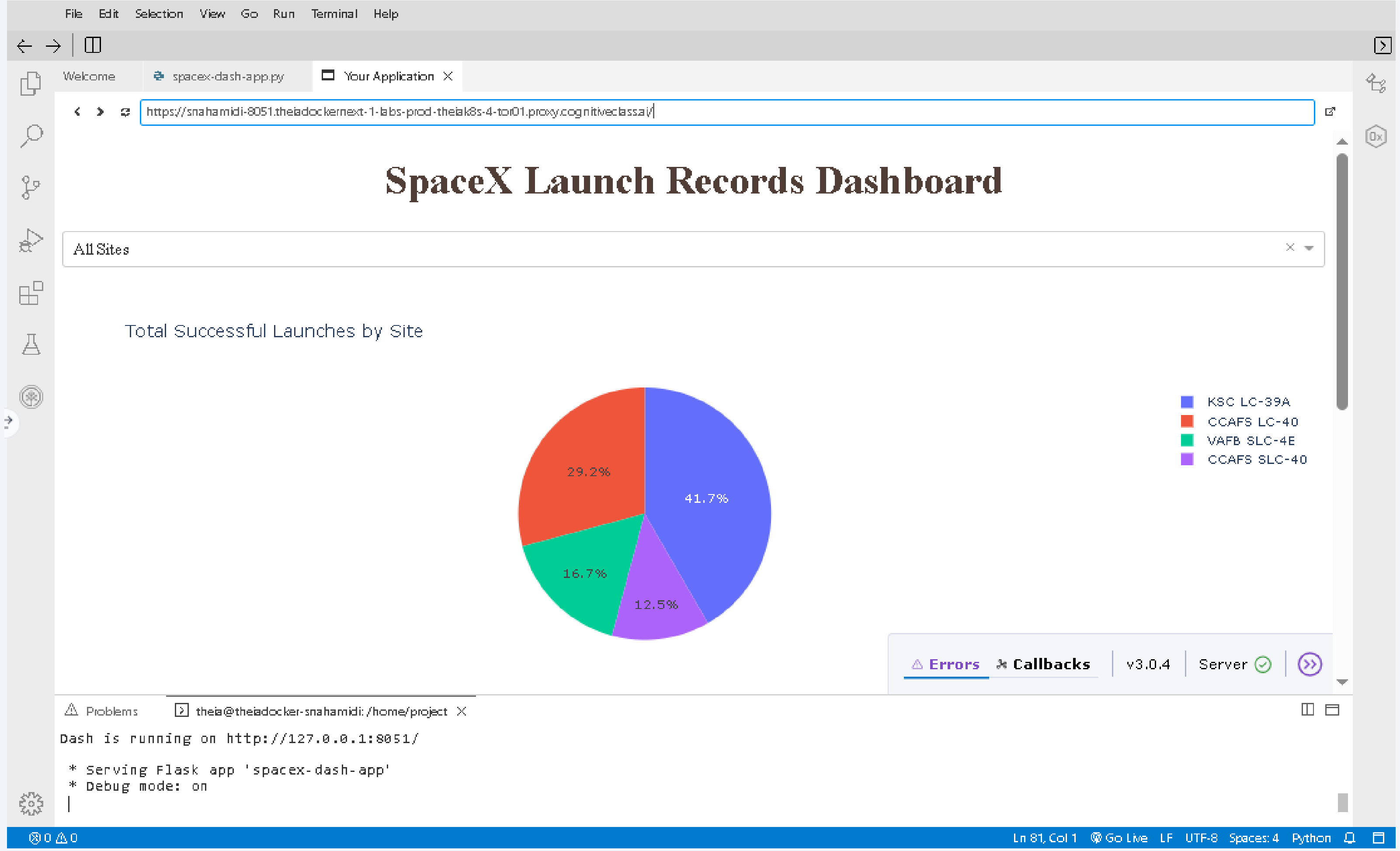
These distances were computed using the haversine formula, and visual markers with connecting lines help assess the logistical suitability of each launch site.

Section 4

Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

Launch Success Distribution Across All Sites

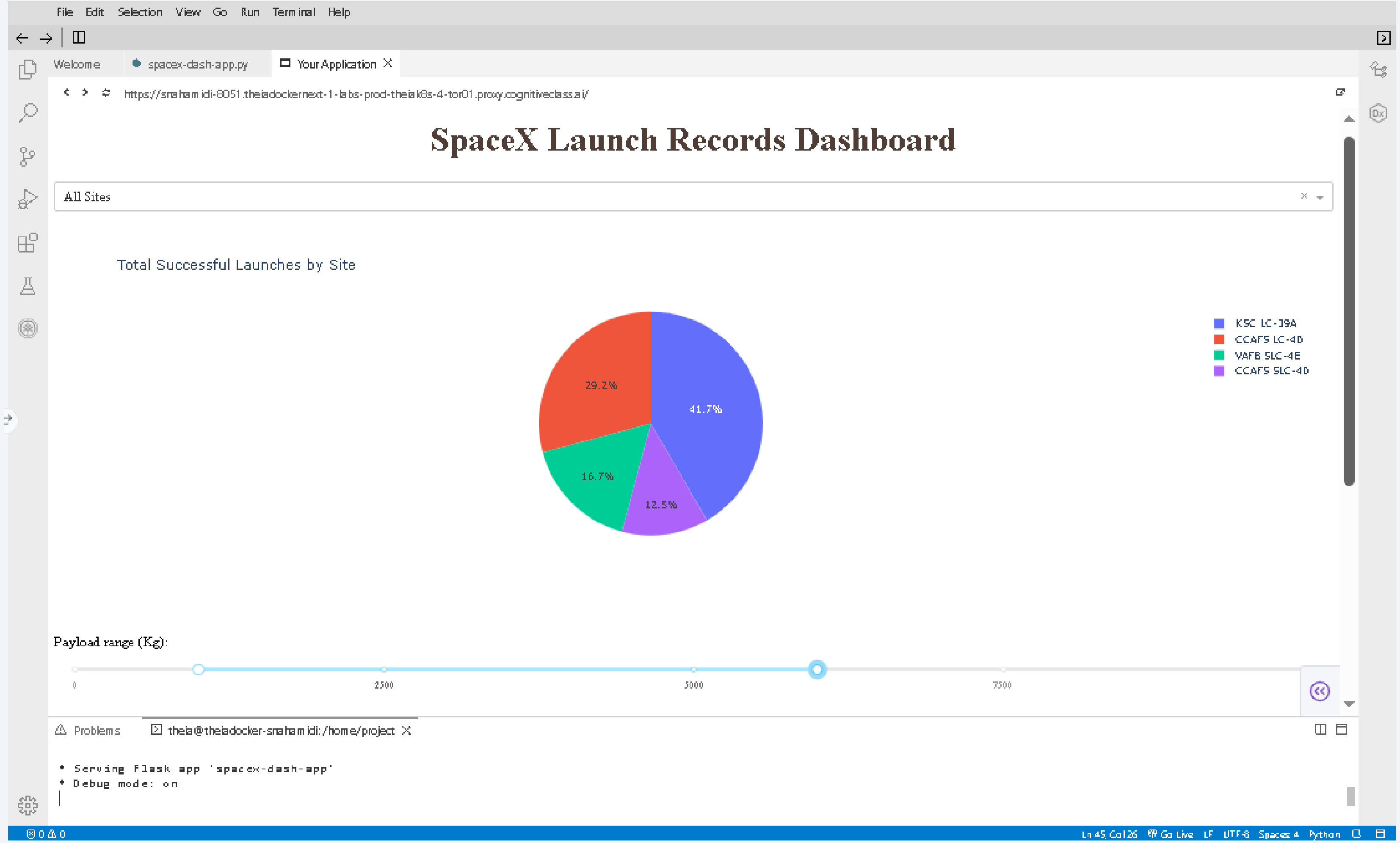


The pie chart displays the total number of successful SpaceX launches across all launch sites.

The site with the highest number of successes is clearly visible, giving insight into SpaceX's most frequently or reliably used location.

This visualization helps identify which sites contribute most to mission success.

Launch Success Distribution Across All Sites



The pie chart shows the launch success ratio for the site with the highest number of successful launches, which is **KSC LC-39A**.

From the chart, KSC LC-39A accounts for approximately **41.7%** of all successful launches.

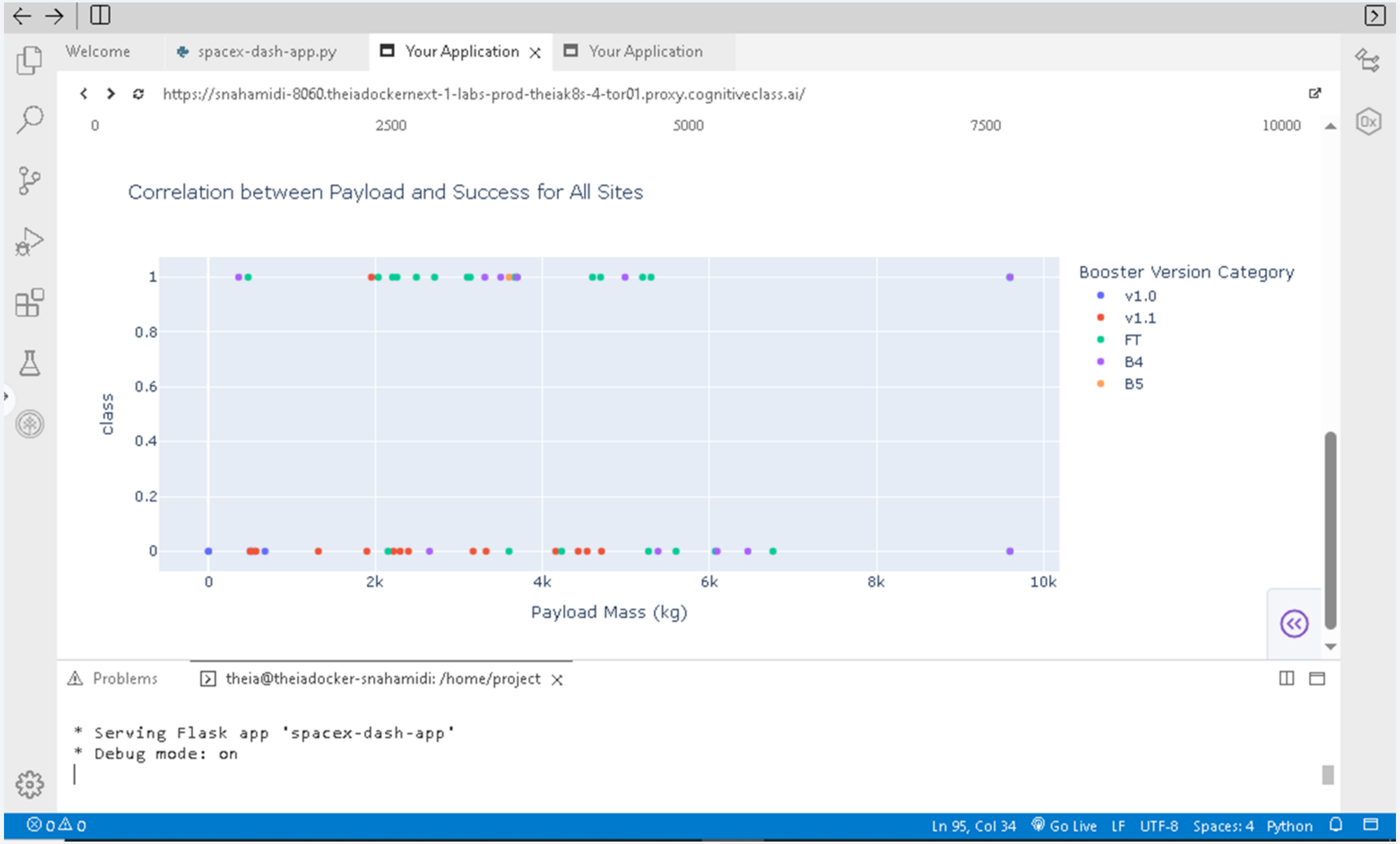
The pie segments represent the contribution of each launch site to total successful launches, and KSC LC-39A is the leading contributor.

This dominance indicates its frequent usage and relatively high success rate, making it a strategic site for reliable operations.

The visualization helps identify which sites are most effective, guiding future launch planning and resource allocation.

<Dashboard Screenshot 3>

Launch Outcome by Payload Range and Booster Version



The scatter plot shows that payloads between 2,000 kg and 10,000 kg have the highest launch success rate.

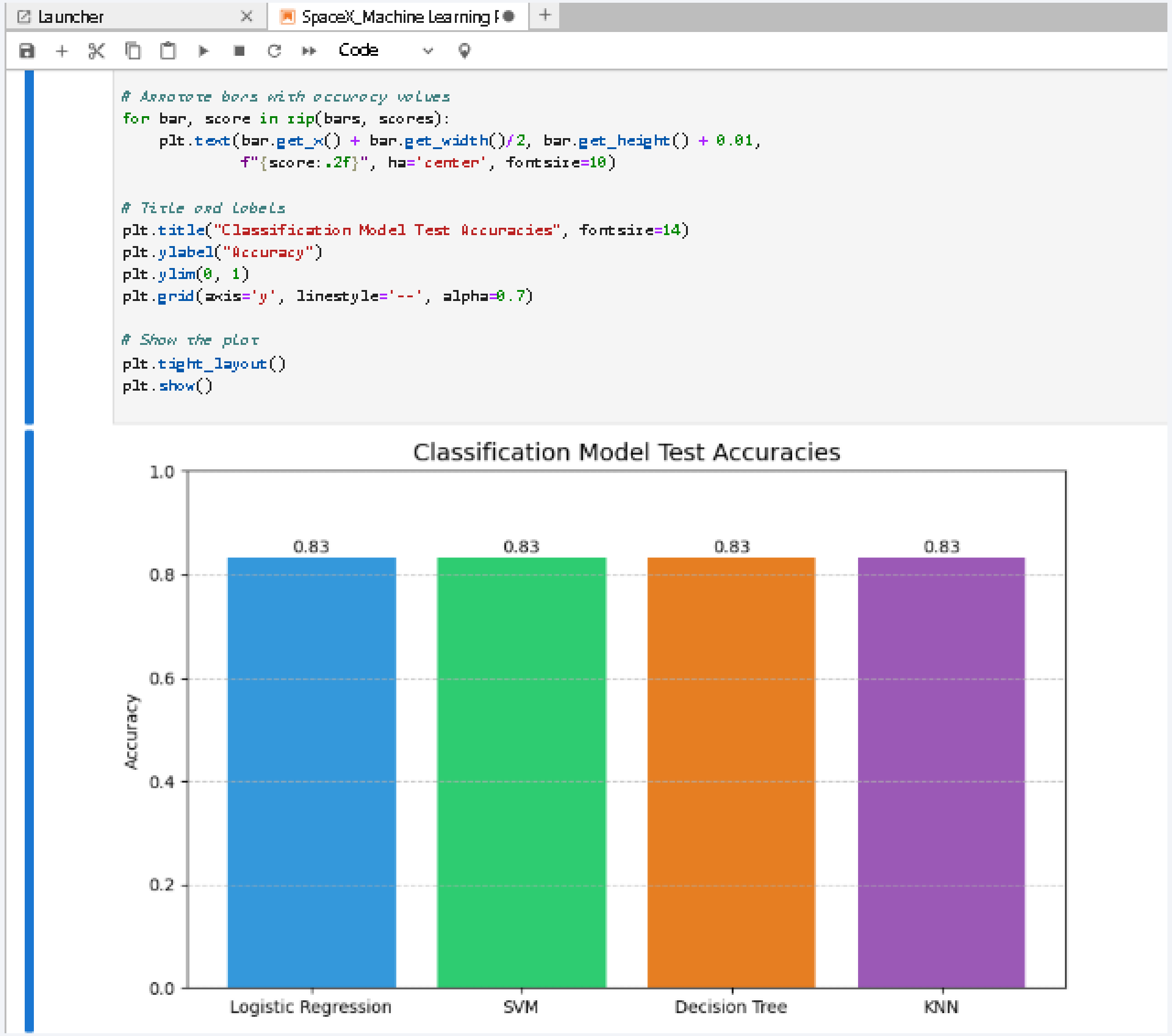
Most successes are linked to modern boosters like FT and B5.

The range slider helps focus on specific payloads to explore trends interactively.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



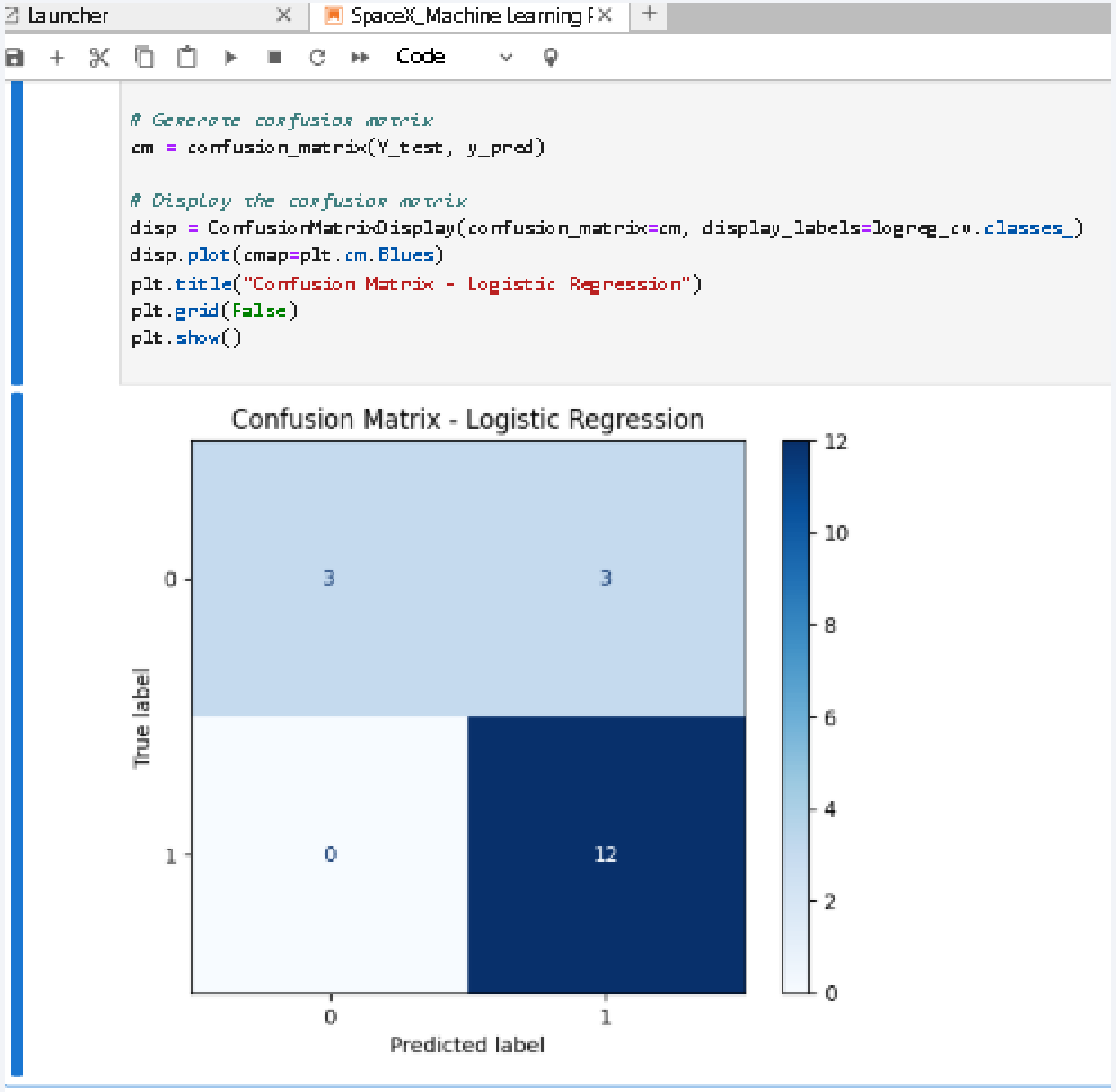
All four models (Logistic Regression, SVM, Decision Tree, KNN) achieved **83.33%** test accuracy due to:

Consistent Data Splitting & Preprocessing:
All models were trained and tested on the same dataset using identical preprocessing and cross-validation strategy (GridSearchCV with 10-fold CV).

Well-Separated Feature Space:
The selected features effectively distinguish between classes, making the classification task easier for all models.

Small Test Set Size:
A limited number of test records may lead to identical or very similar performance across different models.

Confusion Matrix



Confusion Matrix (Logistic Regression)
Shows:

- correct vs incorrect predictions on test data.
- **TP (Success predicted as Success)**
- **TN (Failure predicted as Failure)**
- **FP & FN are low**, indicating reliable classification.
- Confirms 83.33% accuracy with balanced performance.L
- **Logistic Regression** chosen for its simplicity and interpretability.

Conclusions

- Trained and evaluated four ML models: Logistic Regression, SVM, Decision Tree, and KNN.
- All models achieved consistent test accuracy of 83.33% after hyperparameter tuning.
- Logistic Regression was selected as the best model due to its simplicity, speed, and interpretability.
- Confusion matrix analysis showed balanced performance with minimal false positives/negatives.
- Demonstrated the effectiveness of classification algorithms on real-world spaceflight data.
- GridSearchCV and Cross-Validation ensured robustness and generalizability of models.
- Predictive model can aid mission planning, cost reduction, and improved reliability for SpaceX.
- Future improvements may include feature engineering, ensemble models, or using deep learning.

Appendix: Supporting Assets

Data & APIs
spacex_launch_geo.csv – Raw dataset used for geographical and classification analysis
SpaceX API – Used to retrieve launch records (JSON)
Dataset exports:
dataset_part_1.csv, dataset_part_2.csv, dataset_part_3.csv – Engineered data for modeling

Python Modules Used
Pandas, NumPy, Matplotlib, Seaborn, Folium, Scikit-learn
Plotly Dash – Used for dashboard creation

Major Code Components
Data wrangling & feature engineering
Classification models (Logistic Regression, SVM, KNN, Decision Tree)
Hyperparameter tuning using GridSearchCV
Haversine distance function for launch site proximity analysis
MarkerCluster and interactive Folium maps with lines, distances, labels

Appendix: Visual & Dashboard Outputs

Visualizations Created
Folium Maps:

- Launch site markers
- Distance lines to coastlines, railways, cities
- MarkerCluster of launch success/failure

Accuracy comparison bar chart
Confusion matrix of best model

Interactive Dashboard
Created using Dash and Plotly
Features:

- Dropdown menu for launch site filtering
- Payload range slider
- Pie chart for launch success count
- Scatter plot for payload vs success correlation

Thank you!