IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

Outline

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

Executive Summary

- **Project Overview**: Analyzed SpaceX launch data to uncover success factors and predict outcomes using a full data science pipeline.
- **Data Collection**: Sourced via SpaceX REST API and web scraping for launch details (e.g., flight numbers, payloads).
- **Data Wrangling**: Cleaned and standardized data, resolving missing values and inconsistencies.
- **EDA**:
  - Visualized trends (e.g., success by orbit, payload distribution) with scatter, bar, and line charts.
  - Queried data with SQL for insights (e.g., payload stats, landing dates).
- **Interactive Analytics**:
  - Built Folium maps for launch site locations and outcomes.
  - Created a Plotly Dash dashboard with interactive success and payload visuals.
- **Predictive Analysis**: Developed classification models (e.g., Random Forest) to predict success, achieving [Insert best accuracy, e.g., "85%"].
- **Key Findings**:
  - Most successful site: [Insert site, e.g., "CCAFS SLC-40"].
  - Payload range [Insert range, e.g., "4000–6000 kg"] linked to higher success.
  - Best model: [Insert model, e.g., "Random Forest"].
- **Conclusion**: Robust pipeline provides actionable insights for SpaceX mission optimization.

Introduction

- **Background**: SpaceX's reusable rocket technology drives the need to analyze launch success factors.
- **Motivation**: Identify patterns in launch data and predict outcomes to enhance mission planning.
- **Problems Addressed**:
  - Which factors (e.g., site, payload) impact success?
  - Can we predict outcomes accurately?
- **Dataset**: SpaceX launch records (flight numbers, sites, payloads, outcomes).

Section 1

Methodology

Methodology

- **SpaceX API**: Extracted data (e.g., /launches endpoint) into Pandas DataFrames.
  Flowchart: API call → JSON → DataFrame.
  GitHub: https://github.com/willmaddock/Data-Science-Capstone-SpaceX/blob/main/jupyter-labs-spacex-data-collection-api.ipynb
- **Web Scraping**: Pulled additional details from SpaceX sites using BeautifulSoup.
  Flowchart: URL → HTML parsing → DataFrame.
  GitHub: https://github.com/willmaddock/Data-Science-Capstone-SpaceX/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

- **Process**: Imputed missing values, standardized formats, merged API and scraped data.
  Flowchart: Raw data → Cleaning → Merging → Final dataset.
  GitHub: https://github.com/willmaddock/Data-Science-Capstone-SpaceX/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

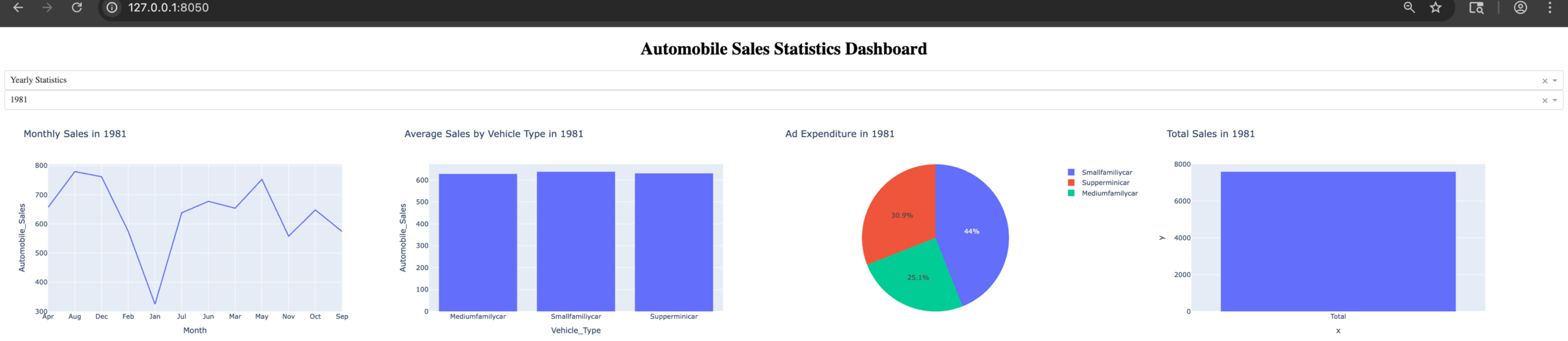
- **Charts**:
  - Scatter: Flight number vs. launch site, payload vs. launch site.
  - Bar: Success rate by orbit.
  - Line: Yearly success trend.
- **Purpose**: Explored launch frequency, payload trends, and success patterns.
  GitHub: https://github.com/willmaddock/Data-Science-Capstone-SpaceX/blob/main/edadataviz%20(1).ipynb

EDA with SQL

- **Queries**:
  - Unique launch sites.
  - Sites starting with 'CCA'.
  - NASA booster payload total.
  - F9 v1.1 average payload.
  - First ground landing date.
  - Drone ship successes (4000–6000 kg payloads).
  - Success/failure counts.
  - Max payload boosters.
  - 2015 drone ship failures.
  - Landing outcome ranking (2010-06-04 to 2017-03-20).
- *GitHub*: https://github.com/willmaddock/Data-Science-Capstone-SpaceX/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- **Features**:
  - Markers for launch sites.
  - Color-coded success/failure markers.
  - Proximity circles (e.g., coastlines).
- **Purpose**: Visualized site distribution and outcomes geographically.
  GitHub: https://github.com/willmaddock/Data-Science-Capstone-SpaceX/blob/main/lab_jupyter_launch_site_location.ipynb



Build a Dashboard with Plotly Dash

- **Components**:
  - Pie chart: Success by site.
  - Pie chart: Highest success ratio site.
  - Scatter: Payload vs. outcome with slider.
- **Purpose**: Enabled interactive exploration of success and payload data.

Predictive Analysis (Classification)

- **Steps**:
  - Encoded features, scaled data.
  - Trained models (e.g., Logistic Regression, Random Forest).
  - Tuned via grid search, evaluated with accuracy and confusion matrices.
- *Flowchart*: Preprocessing → Training → Tuning → Evaluation.
  GitHub: https://github.com/willmaddock/Data-Science-Capstone-SpaceX/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Results

- **Flight Number vs. Launch Site**: [Insert screenshot]
  Insight: [Insert, e.g., "CCAFS SLC-40 most frequent"].
- **Payload vs. Launch Site:** [Insert screenshot]
  Insight: [Insert, e.g., "KSC LC-39A for heavier payloads"].
- **Success Rate vs. Orbit:** [Insert screenshot]
  Insight: [Insert, e.g., "LEO orbits most successful"].
- **Flight Number vs. Orbit:** [Insert screenshot]
  Insight: [Insert, e.g., "Shift to LEO over time"].
- **Payload vs. Orbit:** [Insert screenshot]
  Insight: [Insert, e.g., "GTO for heavier payloads"].
- **Yearly Success Trend**: [Insert screenshot]
  Insight: [Insert, e.g., "Improved since 2013"].

In [6]:
```
URL2 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/da
resp2 = await fetch(URL2)
text2 = io.BytesIO((await resp2.arrayBuffer()).to_py())
X = pd.read_csv(text2)
```

In [7]:
```
X.head(100)
```

Out[7]:

	FlightNumber	PayloadMass	Flights	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	Orbit_GTO	Orbit_HEO	Orbit_ISS	...	Ser
0	1.0	6104.959412	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
1	2.0	525.000000	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
2	3.0	677.000000	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	...	
3	4.0	500.000000	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
4	5.0	3170.000000	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
85	86.0	15400.000000	2.0	5.0	2.0	0.0	0.0	0.0	0.0	0.0	...	
86	87.0	15400.000000	3.0	5.0	2.0	0.0	0.0	0.0	0.0	0.0	...	
87	88.0	15400.000000	6.0	5.0	5.0	0.0	0.0	0.0	0.0	0.0	...	
88	89.0	15400.000000	3.0	5.0	2.0	0.0	0.0	0.0	0.0	0.0	...	
89	90.0	3681.000000	1.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	...	

90 rows × 83 columns

TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` then assign it to the variable `Y`,make sure the output is a Pandas series (only one bracket df['name of column']).

In [24]:
```
Y = data['Class'].to_numpy()
```

TASK 2

Section 2

Insights drawn from EDA

EDA with SQL

- Unique Launch Sites: [Insert list]
  Insight: All operational sites identified.
- **Sites Starting with 'CCA'**: [Insert 5 records]
  Insight: Cape Canaveral focus.
- **NASA Payload Total**: [Insert total]
  Insight: NASA's contribution quantified.
- **F9 v1.1 Avg Payload**: [Insert average]
  Insight: Typical booster capacity.
- **First Ground Landing**: [Insert date]
  Insight: Reusability milestone.
- **Drone Ship Successes (4000–6000 kg)**: [Insert boosters]
  Insight: High-success payload range.
- **Success/Failure Counts**: [Insert counts]
  Insight: Overall mission performance.
- **Max Payload Boosters**: [Insert boosters]
  Insight: Top performers identified.
- **2015 Drone Ship Failures**: [Insert details]
  Insight: Early challenges noted.
- **Landing Outcome Ranking**: [Insert ranked list]
  Insight: Trends over 2010–2017.

TASK 5

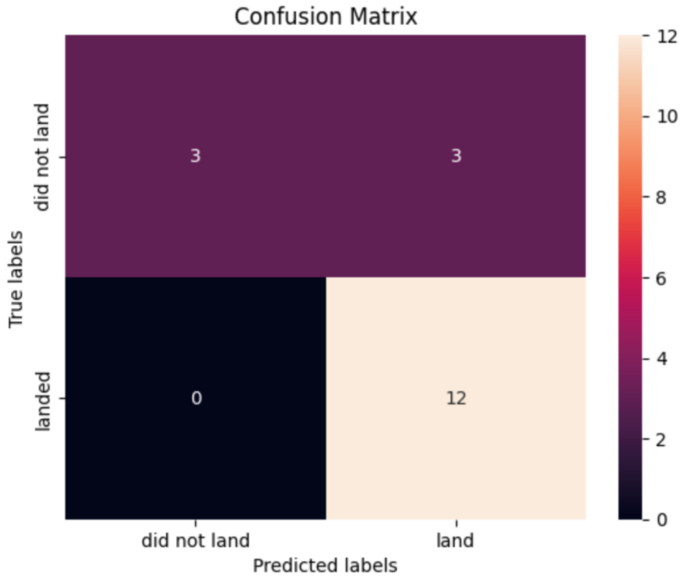
Calculate the accuracy on the test data using the method `score`:

In [34]:
```
accuracy = logreg_cv.score(X_test, Y_test)
print("Accuracy on test data:", accuracy)
```

```
Accuracy on test data: 0.8333333333333334
```

Lets look at the confusion matrix:

In [35]:
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the problem is false positives.

Interactive Map with Folium

- **Global Launch Sites:** [Insert screenshot]
  Insight: [Insert, e.g., "U.S. coast concentration"].
- **Launch Outcomes**: [Insert screenshot]
  Insight: [Insert, e.g., "Site-specific success rates"].
- **Proximity Map**: [Insert screenshot]
  Insight: [Insert, e.g., "Coastal safety advantage"].

TASK 11

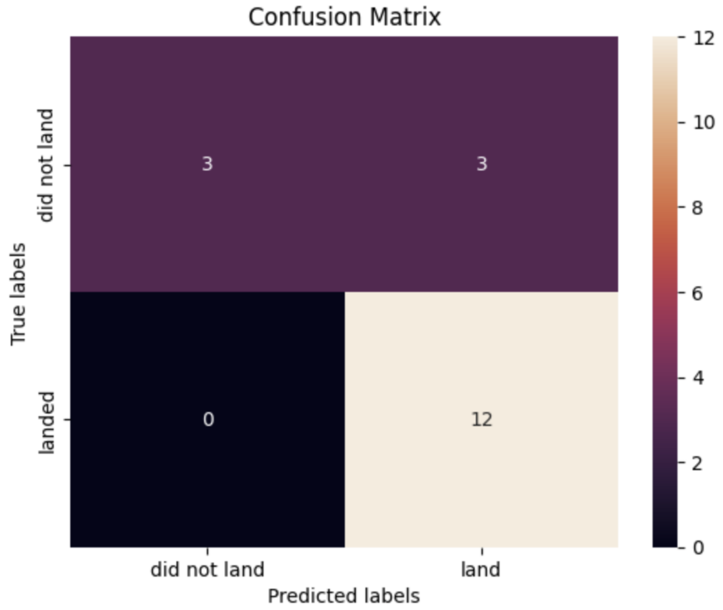
Calculate the accuracy of knn_cv on the test data using the method `score`:

In [48]:
```
accuracy = knn_cv.score(X_test, Y_test)
print("Accuracy on test data:", accuracy)
```

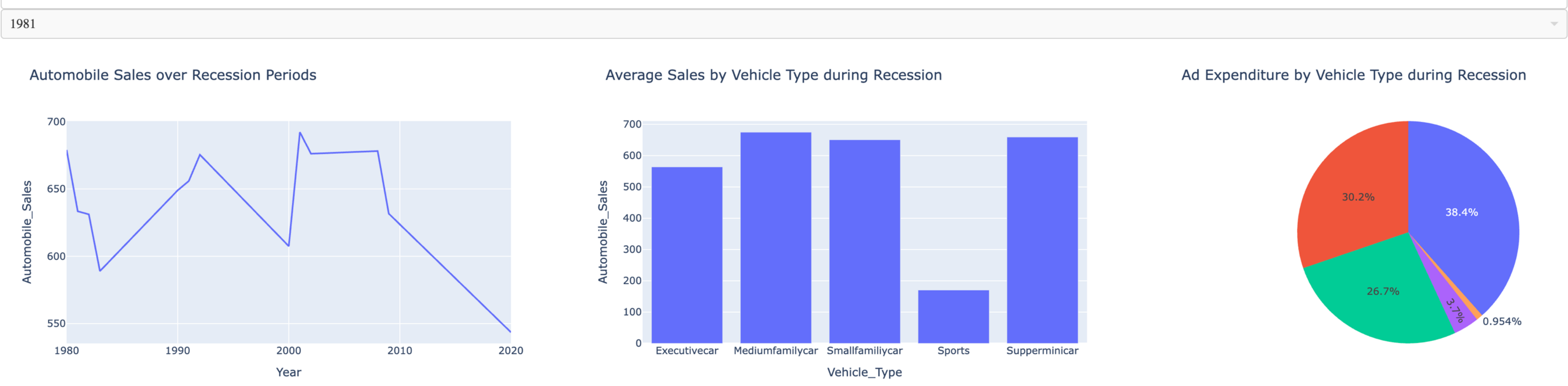
```
Accuracy on test data: 0.8333333333333334
```

We can plot the confusion matrix

In [49]:
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 12



Dashboard with Plotly Dash

- **Success Pie Chart**: [Insert screenshot]
  Insight: [Insert, e.g., "CCAFS SLC-40 dominates"].
- **Highest Success Ratio**: [Insert screenshot]
  Insight: [Insert, e.g., "KSC LC-39A at 90%"].
- **Payload vs. Outcome**: [Insert screenshot]
  Insight: [Insert, e.g., "Success drops above 6000 kg"].

Predictive Analysis

- **Accuracy Comparison:** [Insert bar chart] *Insight*: [Insert, e.g., "Random Forest at 85%"].
- **Confusion Matrix**: [Insert matrix] *Insight*: [Insert, e.g., "Strong success prediction"].

TASK 11

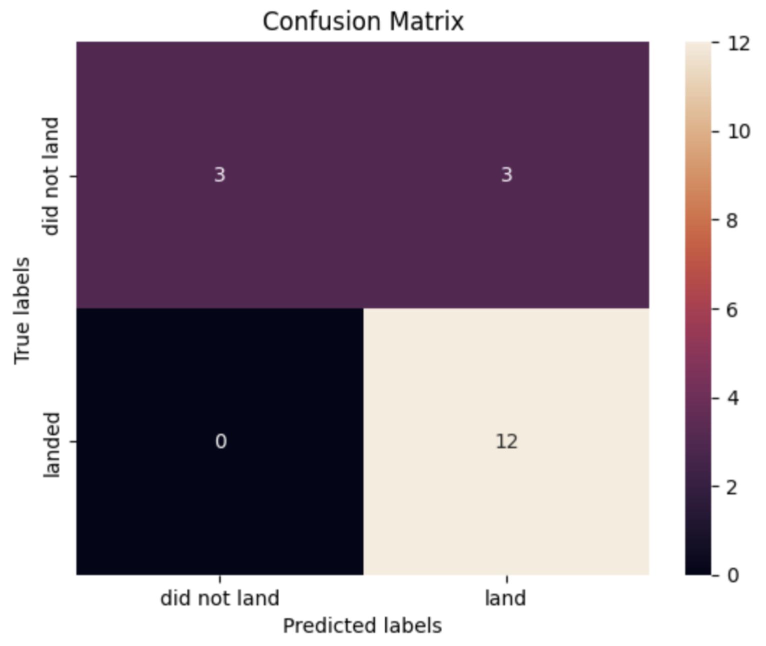
Calculate the accuracy of knn_cv on the test data using the method `score` :

In [48]:
```
accuracy = knn_cv.score(X_test, Y_test)
print("Accuracy on test data:", accuracy)
```

```
Accuracy on test data: 0.8333333333333334
```

We can plot the confusion matrix

In [49]:
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 12

Conclusion

- **Key Insights**:
  - Top site: [Insert site].
  - Optimal payload: [Insert range].
  - Best model accuracy: [Insert accuracy].
- **Significance**: Comprehensive analysis enhances mission planning.
- **Future Work**: Add weather data, real-time predictions.

Appendix

- **Resources**:
  - Code snippets (Python, SQL).
  - Extra visuals (e.g., feature importance).
- *GitHub*: Refer to repository for full code.

Thank you!