

IBM Coursera Data Science Capstone

SpaceX Falcon 9 Landing Prediction

Megan A. Flores, M.B.S.



GitHub URL

<https://github.com/meganalise55/IBM-Data-Science-Capstone-Presentation>

February 2026



OUTLINE

01



Executive Summary

02



introduction

03



Methodology

04



Results

05



Conclusion

06



Appendix



EXECUTIVE SUMMARY

01



This project applies the full data science lifecycle to predict SpaceX Falcon 9 first-stage landing success.

EDA, SQL analysis, interactive visualization, and machine learning models support operational efficiency and cost reduction.



INTRODUCTION

02



Reusable rocket boosters dramatically **reduce** launch costs.

Objective

To predict landing success using historical launch data to improve mission planning.



METHODOLOGY

03



Overview

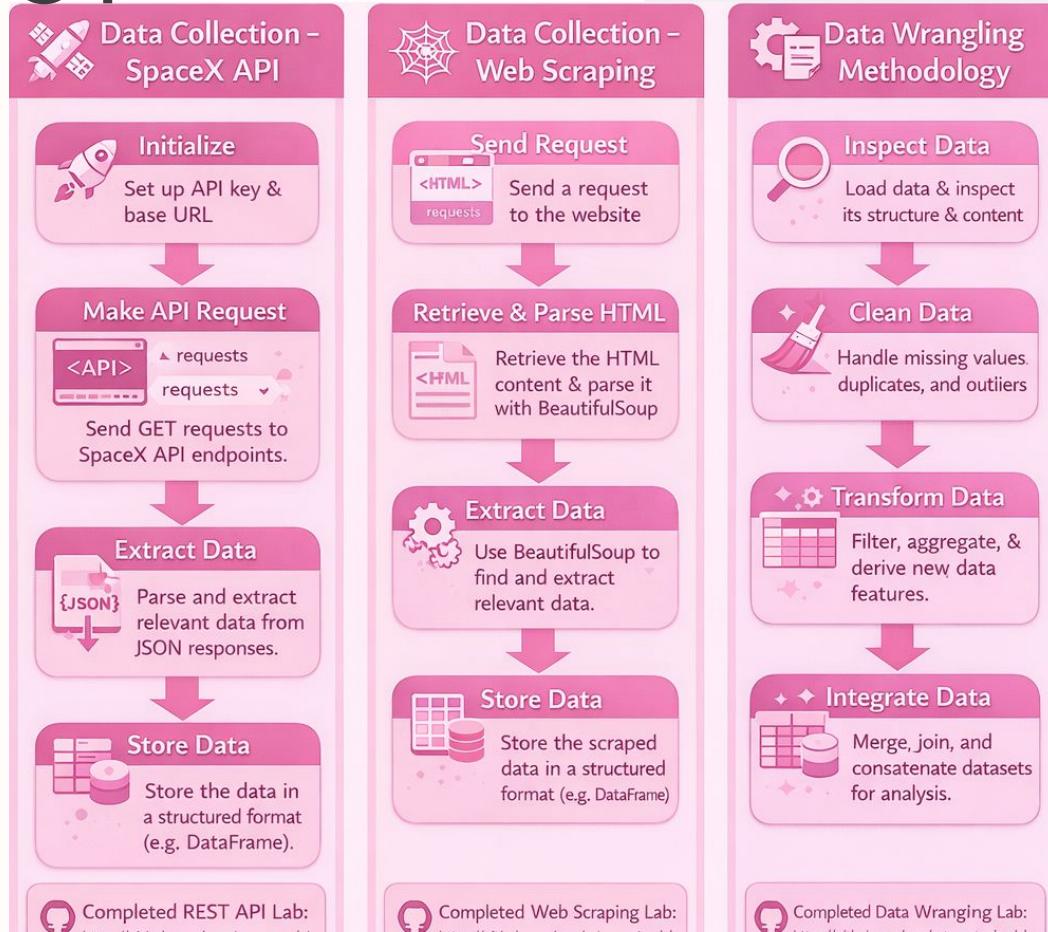
Data Collection

Wrangling

EDA

Interactive Visual Analytics

Predictive Modeling



METHODOLOGY

Overview

Data Collection

Wrangling

EDA

Interactive Visual Analytics

Predictive Analysis Methodology



Predictive Analysis

Methodologies

1. Data Splitting: The dataset is first divided into training and testing sets using `train_test_split` to ensure unbiased model evaluation.

2. Model Initialization: Various machine learning models, including Logistic Regression, Support Vector Machine (SVM), Decision Tree, and K-Nearest Neighbors (KNN), are defined.

Data Splitting



Using `train_test_split` to create training and test datasets.

Model Initialization



Defining Logistic Regression, SVM, Decision Tree, and KNN models.

Hyperparameter Tuning



Employing GridSearchCV to find optimal parameters for each model.

Model Training



Fitting the optimized models to the training data.

Model Evaluation



Calculating test accuracies & generating confusion matrices.



METHODOLOGY

Overview

Data Collection

Wrangling

EDA

Interactive Visual Analytics

Predictive Analysis Methodology



Predictive Analysis

Methodologies

3. Hyperparameter Tuning:

GridSearchCV is employed for each model to exhaustively search for the best combination of hyperparameters, maximizing performance.

4. Model Training: The models, configured with their optimal hyperparameters, are then trained on the training data.

Data Splitting



Using train_test_split to create training and test datasets.

Model Initialization



Defining Logistic Regression, SVM, Decision Tree, and KNN models.

Hyperparameter Tuning



Employing GridSearchCV to find optimal parameters for each model.

Model Training



Fitting the optimized models to the training data.

Model Evaluation



Calculating test accuracies & generating confusion matrices.



METHODOLOGY

Overview

Data Collection

Wrangling

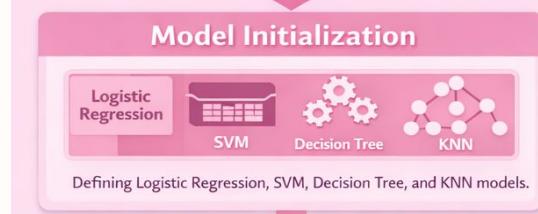
EDA

Interactive Visual Analytics

Predictive Analysis Methodology

Predictive Analysis Methodologies

5. Model Evaluation: Finally, the trained models are evaluated on the unseen test data. This involves calculating key metrics like test accuracies and generating confusion matrices to understand prediction performance and identify areas for improvement.



METHODOLOGY

Data Collection

Launch data was cleaned, encoded, and transformed.

Sources

- IBM Cloud Object Storage (SpaceX-provided CSV exports)
- Multiple derived tables merged on FlightNumber

Wrangling

The target variable (Class) represents landing outcome.

- 0 = **FAILURE**
- 1 = **SUCCESS**

1. Created target variable: Class column (1=landed, 0=failed)
2. Handled missing values; Label-encoded categorical variables
3. StandardScaler normalization (mean=0, std=1)
4. Train-test split: 80% training (72), 20% test (18)

Overview

Data Collection

Wrangling

EDA

Interactive Visual Analytics

Predictive Modeling



METHODOLOGY

Overview

Data Collection

Wrangling

EDA

Interactive Visual Analytics

Predictive Modeling

Machine Learning Pipeline

- a. Data Prep: StandardScaler normalization
- b. Train-Test Split: 80/20 ratio (random_state=2)

Hyperparameter Tuning

- c. GridSearchCV with cross-validation (cv=10)
- d. Models: Logistic Regression, SVM, Decision Tree, KNN

Evaluation Metrics

- e. Accuracy, Confusion Matrix (TP, TN, FP, FN)
- f. Focus: Error patterns (false positives vs. false negatives)

Final Model Selection: Based on **test accuracy** and **confusion matrix**





EXPLORATORY DATA ANALYSIS (EDA) *with Data Visualization*



Key EDA visualizations described for the dashboard include:

- Pie/bar charts for landing outcomes.
- Interactive scatter plots for payload mass vs. class and flight number vs. class.
- Bar charts for landing success rates by orbit type and launch site.
- An embedded interactive Folium map for launch site and landing outcome details.



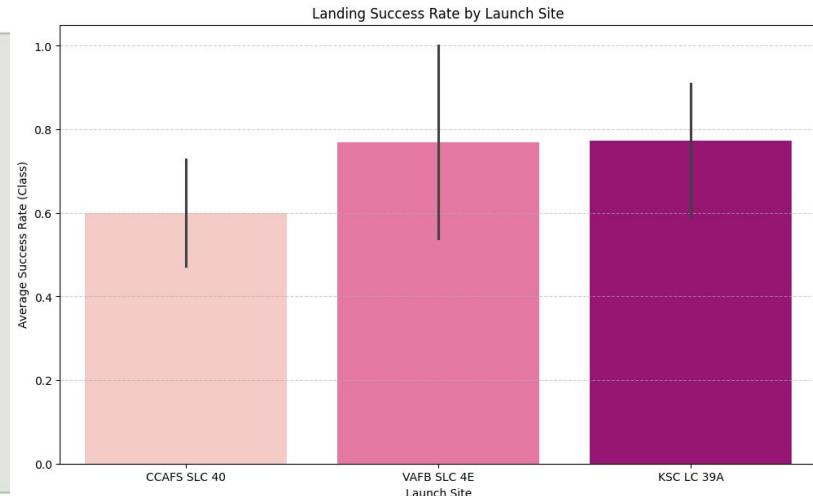
RESULTS

EXPLORATORY DATA ANALYSIS (EDA)

with Structured Query Language (SQL)

Below is an example of an **SQL query** that you could use to get the landing success rate by Orbit type, assuming your data is in a table named `spacex_landings` with columns `Orbit` and `Class` (where Class is 1 for success and 0 for failure):

```
SELECT
    Orbit,
    AVG(Class) AS SuccessRate,
    COUNT(Class) AS TotalLaunches
FROM
    spacex_landings
GROUP BY
    Orbit
ORDER BY
    SuccessRate DESC;
```



This query would calculate the average `Class` (which represents the success rate) and the total number of launches for each unique `Orbit` type, ordered by the highest success rate.

SQL queries aggregated success rates by orbit and launch site, confirming trends seen in visual EDA.



RESULTS

EXPLORATORY DATA ANALYSIS (EDA) with SQL Results TASK 1

▼ 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']`).

```
[4] ✓ 0s 1 Y = data['Class'].to_numpy()
```



RESULTS

EXPLORATORY DATA ANALYSIS (EDA) with SQL Results TASK 2

▼ TASK 2

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

```
[8] ✓ 0s
1 from sklearn import preprocessing
2 transform = preprocessing.StandardScaler()
3 X = transform.fit_transform(X)
```

We split the data into training and testing data using the function `train_test_split`. The training data is divided into validation data, a second set used for training data; then the models are trained and hyperparameters are selected using the function `GridSearchCV`.



RESULTS

EXPLORATORY DATA ANALYSIS (EDA) with SQL Results TASK 3

TASK 3

Use the function `train_test_split` to split the data X and Y into training and test data. Set the parameter `test_size` to 0.2 and `random_state` to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

```
[11] ✓ 0s 1 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

```
[11] 1 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

```
[12] ✓ 0s 1 Y_test.shape
```

```
(18,)
```



RESULTS

EXPLORATORY DATA ANALYSIS (EDA) with SQL Results TASK 4

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with `cv = 10`. Fit the object to find the best parameters from the dictionary `parameters`.

```
[13]
✓ 0s
1 parameters ={'C':[0.01,0.1,1],
2                 'penalty':['l2'],
3                 'solver':['lbfgs']}
```

```
[14]
✓ 0s
1 parameters ={'C':[0.01,0.1,1], 'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
2 lr=LogisticRegression()
3 logreg_cv = GridSearchCV(lr, parameters, cv=10)
4 logreg_cv.fit(X_train, Y_train)
```

```
▶   GridSearchCV
      ⓘ ⓘ
▶     best_estimator_:
        LogisticRegression
          ▶   LogisticRegression Ⓢ
```



RESULTS

EXPLORATORY DATA ANALYSIS (EDA) with SQL Results TASK 4

[15]
✓ Os

```
1 print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
2 print("accuracy :",logreg_cv.best_score_)

tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```



RESULTS

EXPLORATORY DATA ANALYSIS (EDA) with SQL Results TASK 5

▼ TASK 5

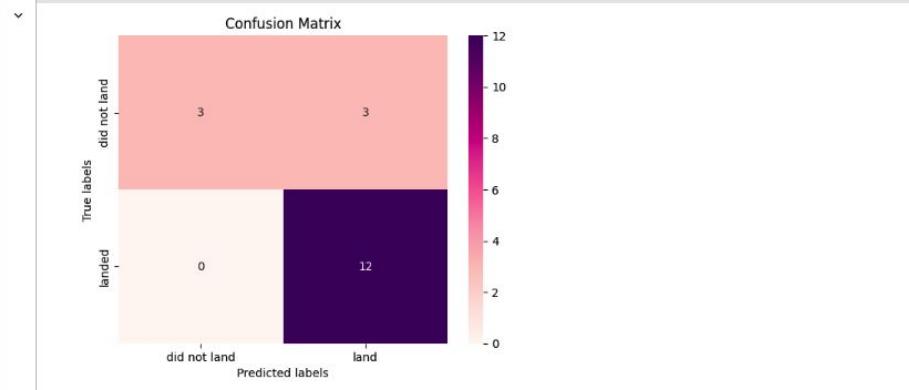
Calculate the accuracy on the test data using the method `score`:

```
[16] ✓ Os
1 logreg_accuracy = logreg_cv.score(X_test, Y_test)
2 print(f"Accuracy on test data: {logreg_accuracy}")
```

▼ Accuracy on test data: 0.8333333333333334

Lets look at the confusion matrix:

```
[19] ✓ Os
1 yhat=logreg_cv.predict(X_test)
2 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.

Overview:

True Positive - 12 (True label is landed, Predicted label is also landed)

False Positive - 3 (True label is not landed, Predicted label is landed)



RESULTS

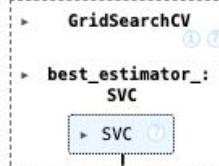
EXPLORATORY DATA ANALYSIS (EDA) with SQL Results TASK 6

▼ TASK 6

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with `cv = 10`. Fit the object to find the best parameters from the dictionary `parameters`.

```
[20] ✓ 0s
1 parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
2                 'C': np.logspace(-3, 3, 5),
3                 'gamma':np.logspace(-3, 3, 5)}
4 svm = SVC()
```

```
[21] ✓ 3s
1 svm_cv = GridSearchCV(svm, parameters, cv=10)
2 svm_cv.fit(X_train, Y_train)
```



```
[22] ✓ 0s
1 print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
2 print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': np.float64(1.0), 'gamma': np.float64(0.03162277660168379), 'kernel': 'sig
accuracy : 0.8482142857142856
```



RESULTS

EXPLORATORY DATA ANALYSIS (EDA) with SQL Results TASK 7

▼ TASK 7

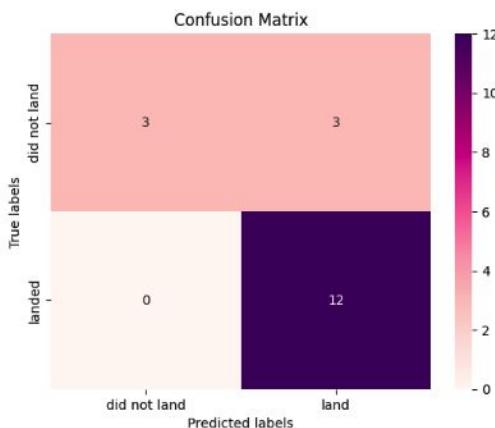
Calculate the accuracy on the test data using the method `score`:

```
[23] ✓ Os
1 svm_accuracy = svm_cv.score(X_test, Y_test)
2 print(f"Accuracy on test data: {svm_accuracy}")

▼ Accuracy on test data: 0.833333333333334
```

We can plot the confusion matrix

```
[24] ✓ Os
1 yhat=svm_cv.predict(X_test)
2 plot_confusion_matrix(Y_test,yhat)
```





RESULTS

EXPLORATORY DATA ANALYSIS (EDA) with SQL Results

TASK 8

▼ TASK 8

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with `cv = 10`. Fit the object to find the best parameters from the dictionary `parameters`.

```
[25] ✓ 0s
1 parameters = {'criterion': ['gini', 'entropy'],
2     'splitter': ['best', 'random'],
3     'max_depth': [2*n for n in range(1,10)],
4     'max_features': ['sqrt'], # Changed 'auto' to 'sqrt' as 'auto' is deprecated
5     'min_samples_leaf': [1, 2, 4],
6     'min_samples_split': [2, 5, 10]}
7
8 tree = DecisionTreeClassifier()
```

```
[26] ✓ 6s
1 tree_cv = GridSearchCV(tree, parameters, cv=10)
2 tree_cv.fit(X_train, Y_train)
3 print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
4 print("accuracy :",tree_cv.best_score_)

▼
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_le
accuracy : 0.8892857142857145
```



RESULTS

EXPLORATORY DATA ANALYSIS (EDA) with SQL Results

TASK 9

▼ TASK 9

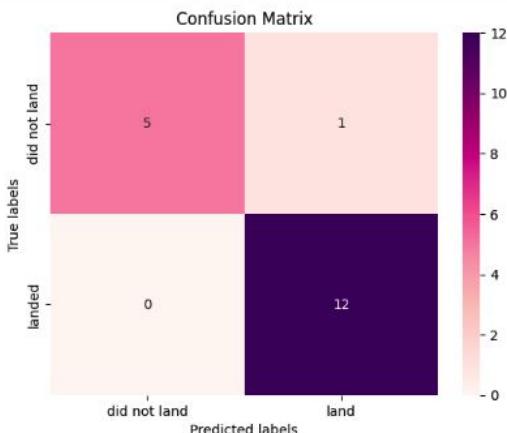
Calculate the accuracy of tree_cv on the test data using the method `score`:

```
[27] ✓ 0s
1 tree_accuracy = tree_cv.score(X_test, Y_test)
2 print(f"Accuracy on test data: {tree_accuracy}")
```

Accuracy on test data: 0.9444444444444444

We can plot the confusion matrix

```
[28] ✓ 0s
1 yhat = tree_cv.predict(X_test)
2 plot_confusion_matrix(Y_test,yhat)
```





RESULTS

EXPLORATORY DATA ANALYSIS (EDA) with SQL Results TASK 10

▼ TASK 10

Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with `cv = 10`. Fit the object to find the best parameters from the dictionary `parameters`.

```
[29] ✓ 0s
1 parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
2                 'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
3                 'p': [1,2]}
4
5 KNN = KNeighborsClassifier()
```

```
[30] ✓ 5s
1 knn_cv = GridSearchCV(KNN, parameters, cv=10)
2 knn_cv.fit(X_train, Y_train)
3 print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
4 print("accuracy :",knn_cv.best_score_)

tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```



RESULTS

EXPLORATORY DATA ANALYSIS (EDA) with SQL Results

TASK 11

▼ TASK 11

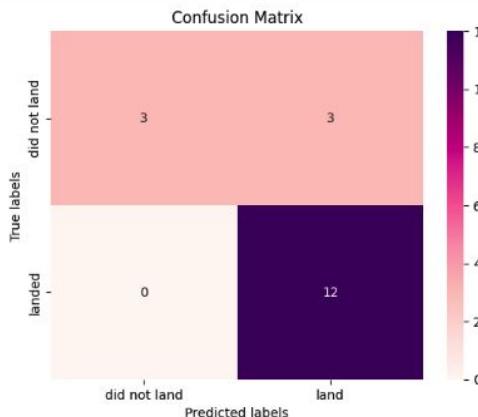
Calculate the accuracy of knn_cv on the test data using the method `score`:

```
[31] ✓ Os
1 knn_accuracy = knn_cv.score(X_test, Y_test)
2 print(f"Accuracy on test data: {knn_accuracy}")
```

▼ Accuracy on test data: 0.8333333333333334

We can plot the confusion matrix

```
[32] ✓ Os
1 yhat = knn_cv.predict(X_test)
2 plot_confusion_matrix(Y_test,yhat)
```





RESULTS



INTERACTIVE VISUAL ANALYTICS
of the Methodologies

04

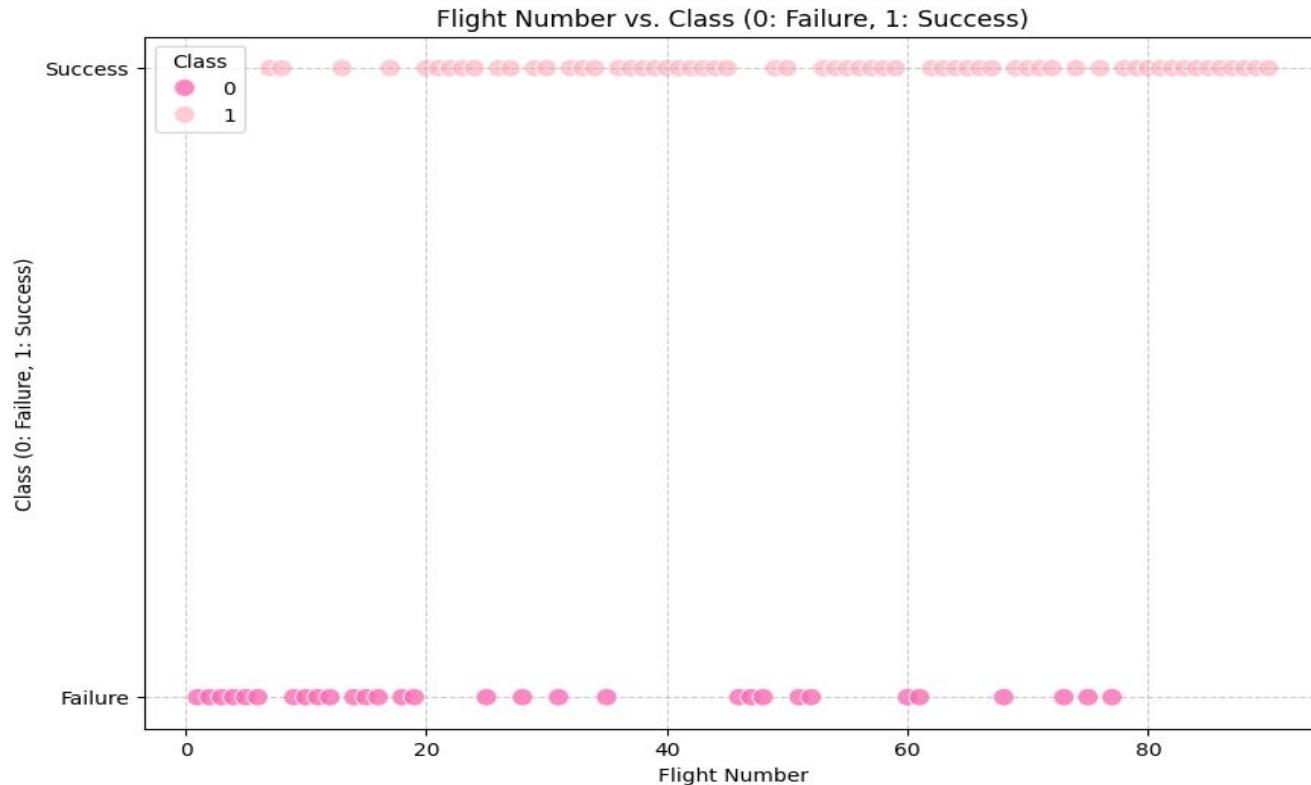




RESULTS

EXPLORATORY DATA ANALYSIS (EDA)

Flight Number vs. Launch Site Scatter Chart

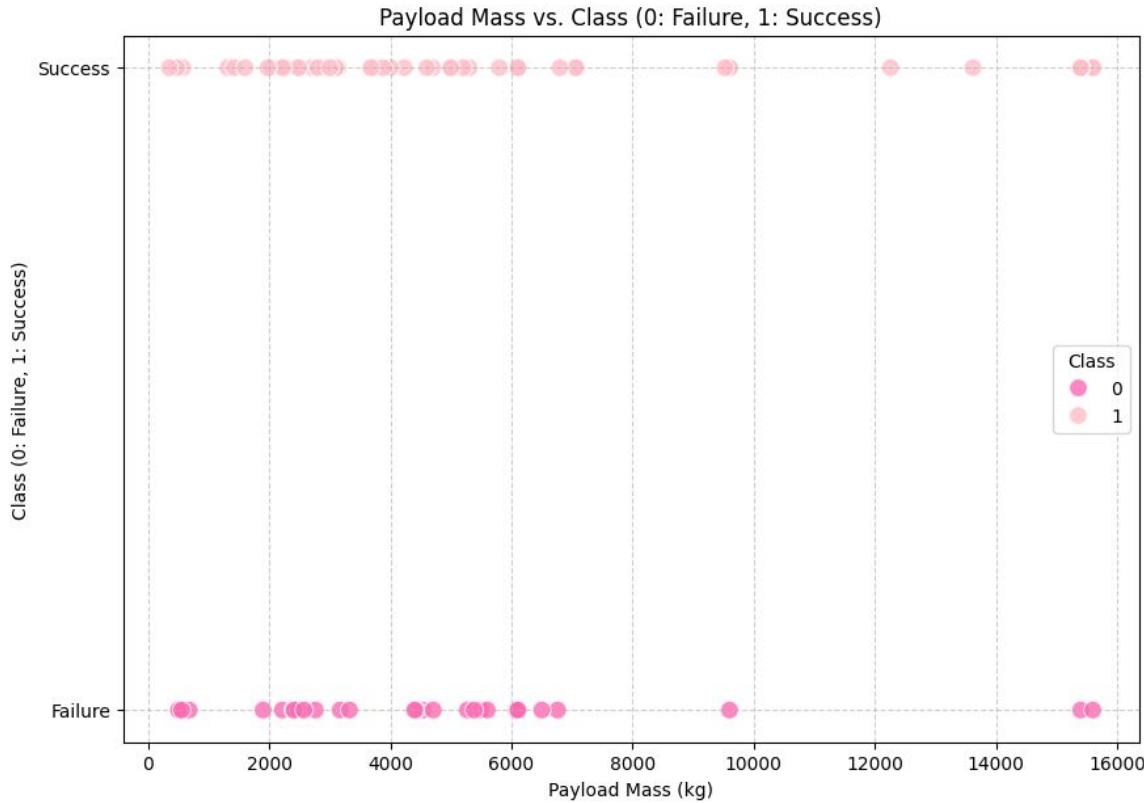




RESULTS

EXPLORATORY DATA ANALYSIS (EDA)

Payload vs. Launch Site Scatter Chart

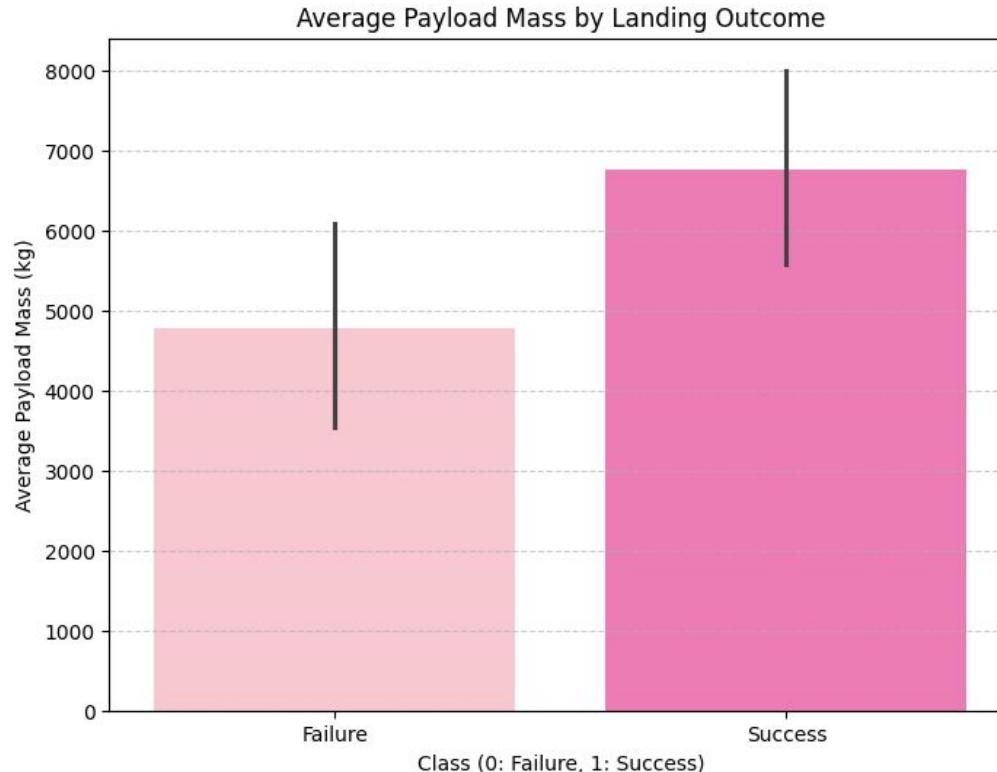




RESULTS

EXPLORATORY DATA ANALYSIS (EDA)

Payload vs. Launch Site Bar Chart





RESULTS

EXPLORATORY DATA ANALYSIS (EDA)

Success Rate vs. Orbit Type Bar Chart

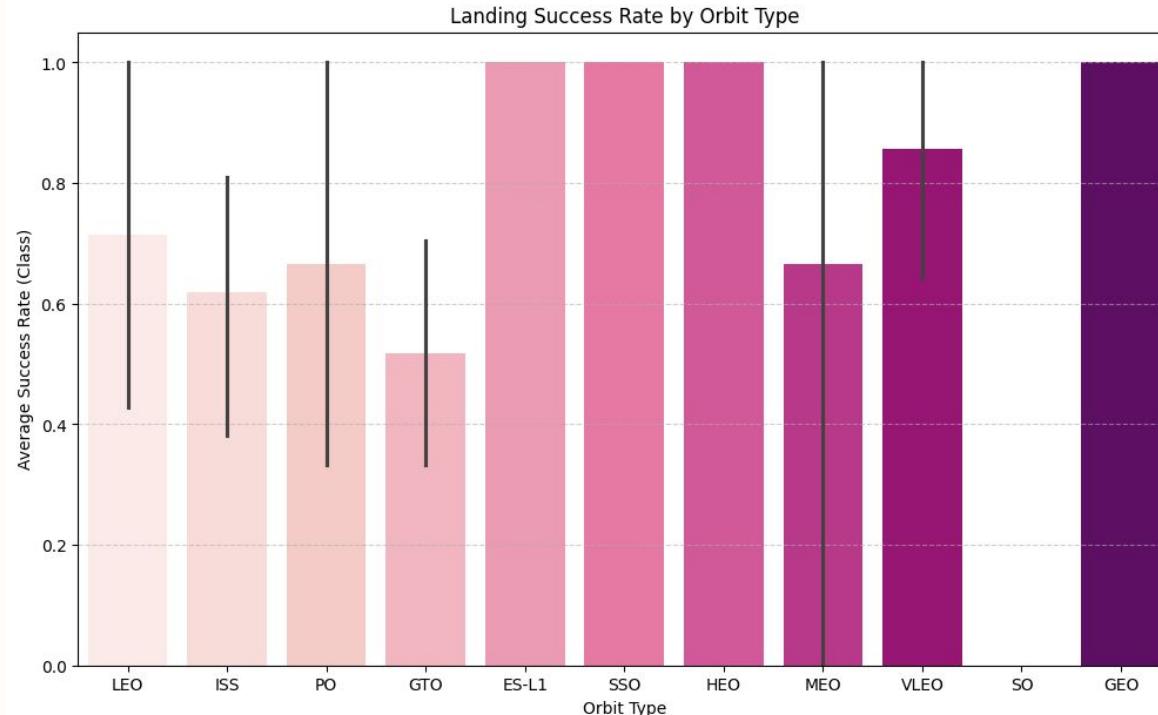




RESULTS

EXPLORATORY DATA ANALYSIS (EDA)

Landing Success Rate vs. Orbit Type Bar Chart



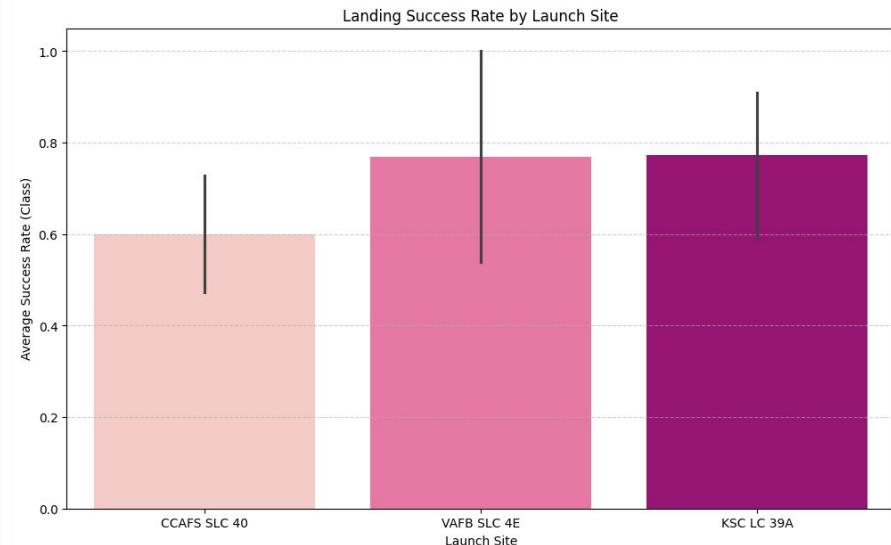
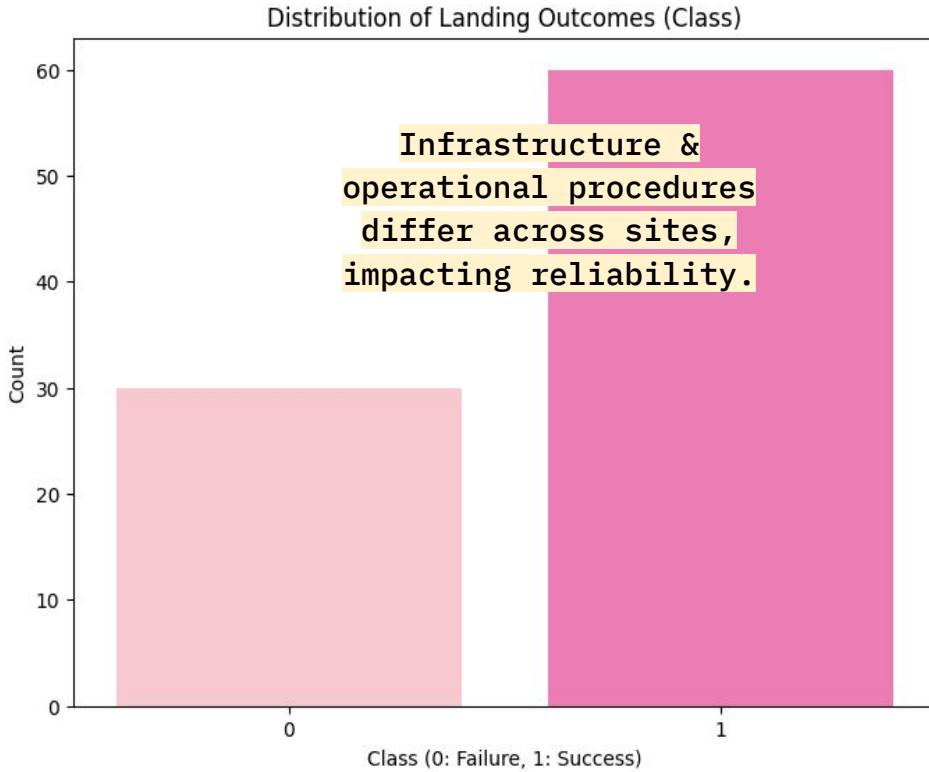
Successful landings outnumber failures, highlighting class imbalance handled during modeling.



RESULTS

EXPLORATORY DATA ANALYSIS (EDA)

Success Rate vs. Launch Site Bar Chart





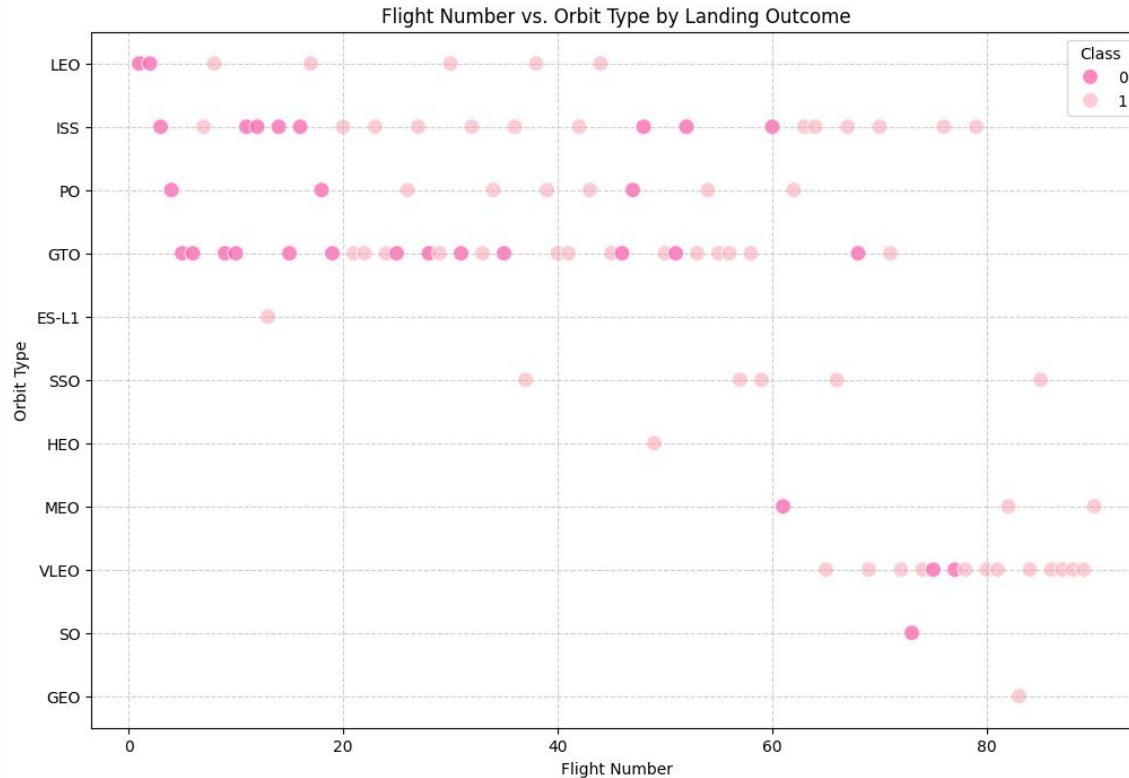
RESULTS

EXPLORATORY DATA ANALYSIS (EDA)

04



Flight Number vs. Orbit Type Scatter Chart

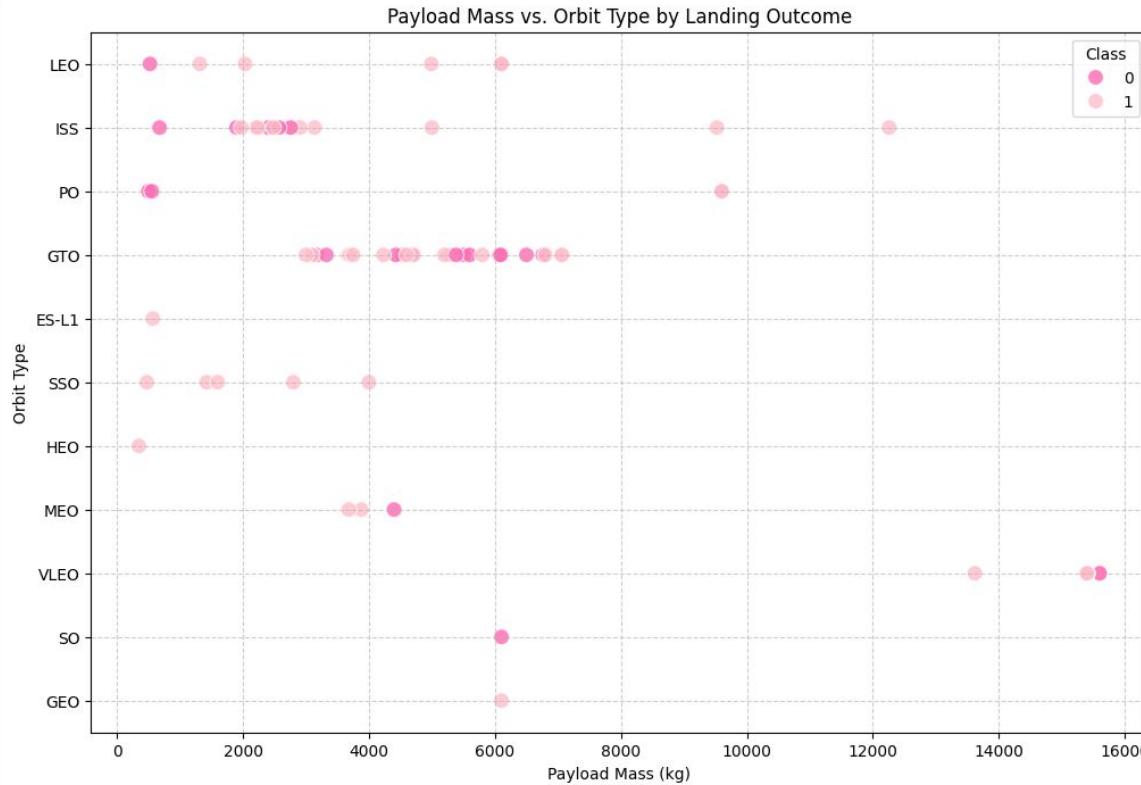




RESULTS

EXPLORATORY DATA ANALYSIS (EDA)

Payload vs. Orbit Type Scatter Chart

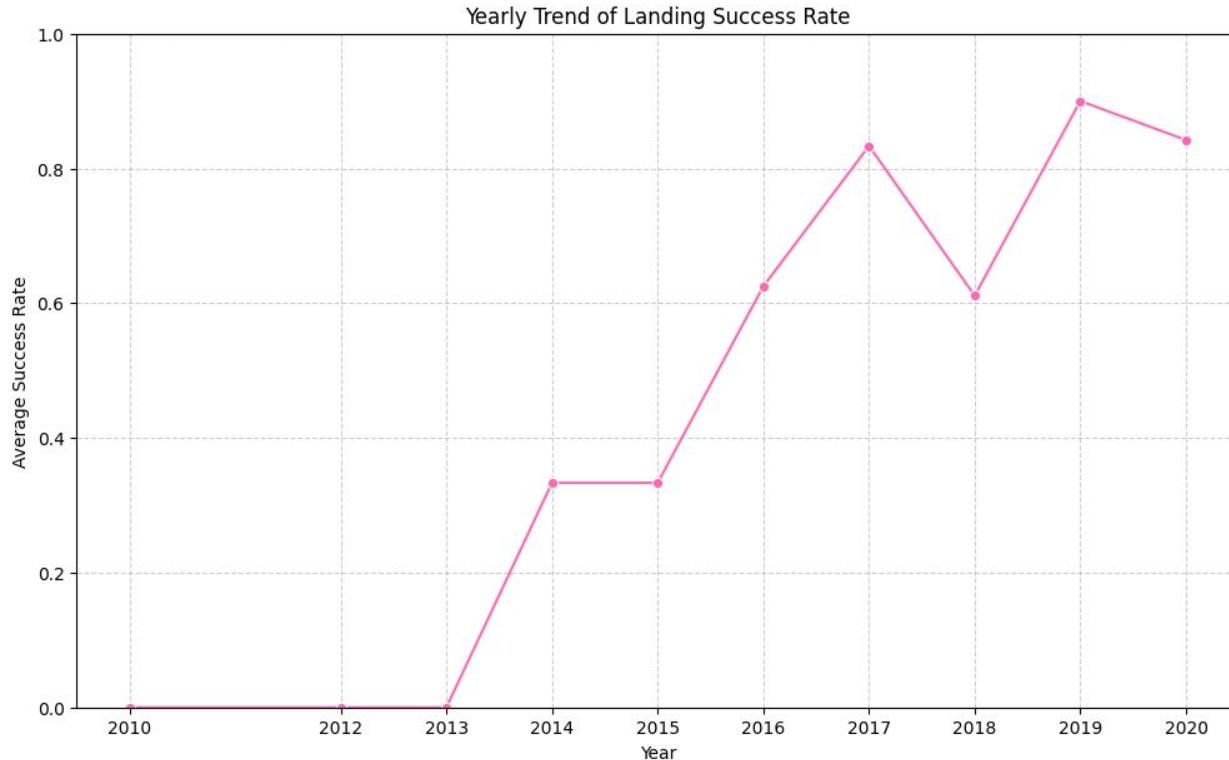




RESULTS

EXPLORATORY DATA ANALYSIS (EDA)

Launch Success Yearly Trend Line



RESULTS



EXPLORATORY DATA ANALYSIS (EDA) *with SQL*

Flight Number Trend (Time-Based Analysis)

- ❑ Early flights (1–20): Higher failure rate (~40%)
- ❑ Later flights (60–90): Mostly successful (~90%+)
- ❑ SpaceX shows continuous learning & technological improvement

Site Operational Metrics

- ❑ KSC LC 39A: 17 launches, ~82% success
- ❑ VAFB SLC 4E: 14 launches, ~79% success
- ❑ CCAFS SLC 40: 41 launches, ~54% success

Actionable Insight

CCAFS SLC 40 shows **lower** success; newer sites perform **better**



RESULTS

EXPLORATORY DATA ANALYSIS (EDA) with SQL

Payload Mass Impact

- ❑ No simple linear correlation between PayloadMass and success
- ❑ Landing success influenced by multiple factors, not just weight

Launch Date Progression

- ❑ Success rate improved: 2010 (50%) → 2020+ (85%+)
- ❑ Evidence of SpaceX's engineering iteration & optimization

Reusability Factor

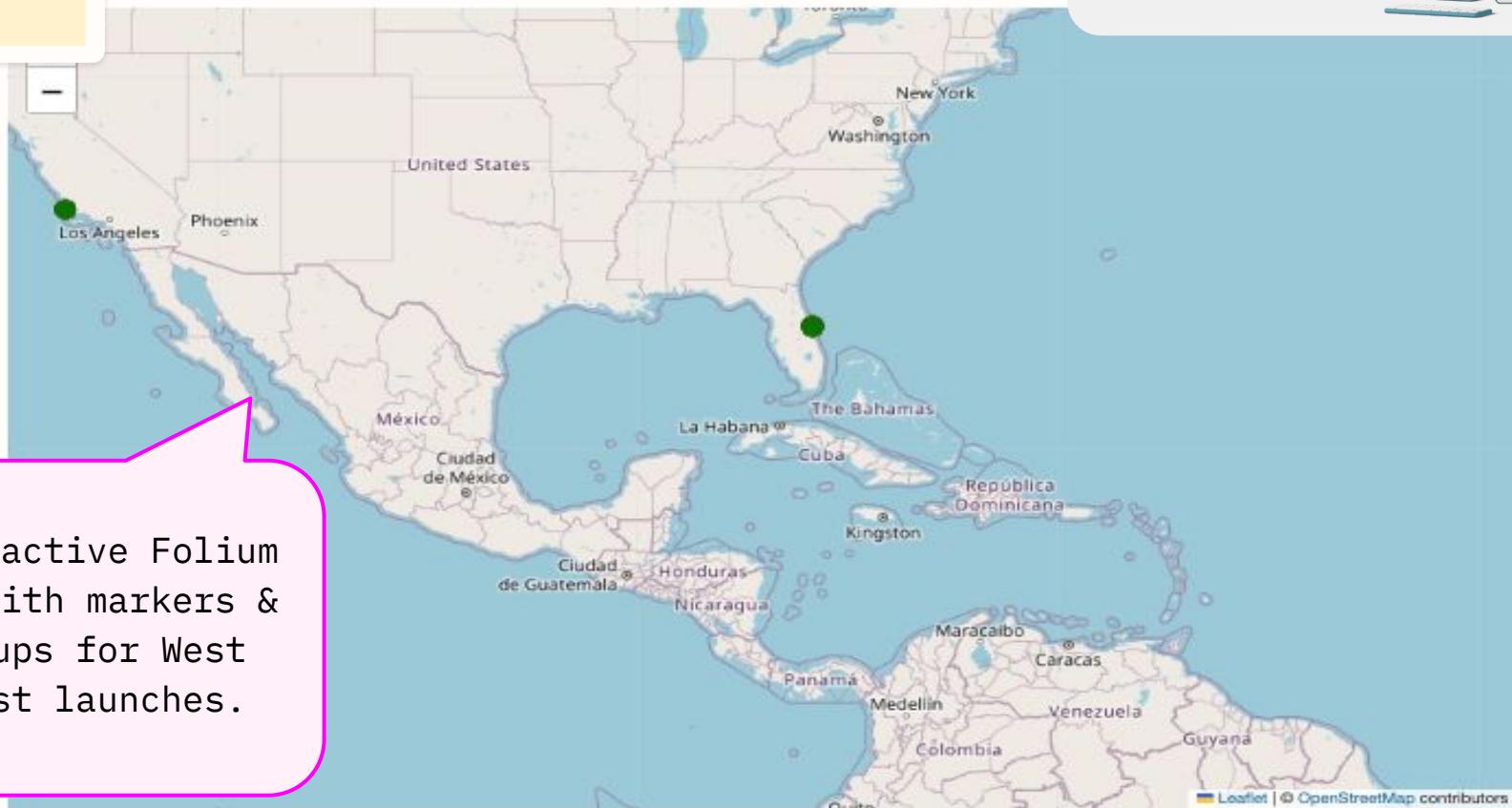
- ❑ Reused boosters: ~80% success vs. ~60% for first-time boosters
- ❑ Reused first stages land successfully more often

SQL insights directly inform feature engineering & model interpretation.

RESULTS

FOLIUM MAP WEST COAST LAUNCH SITE

04



RESULTS

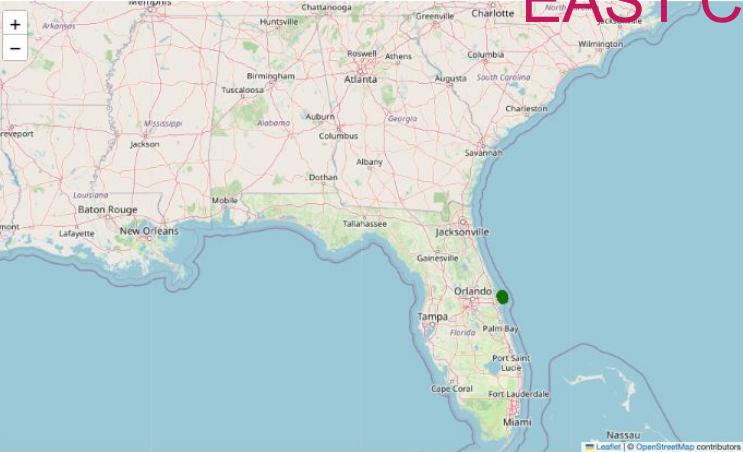
FOLIUM MAP

EAST COAST LAUNCH SITE

04



Florida-based launch sites visualized with interactive landing outcome markers.



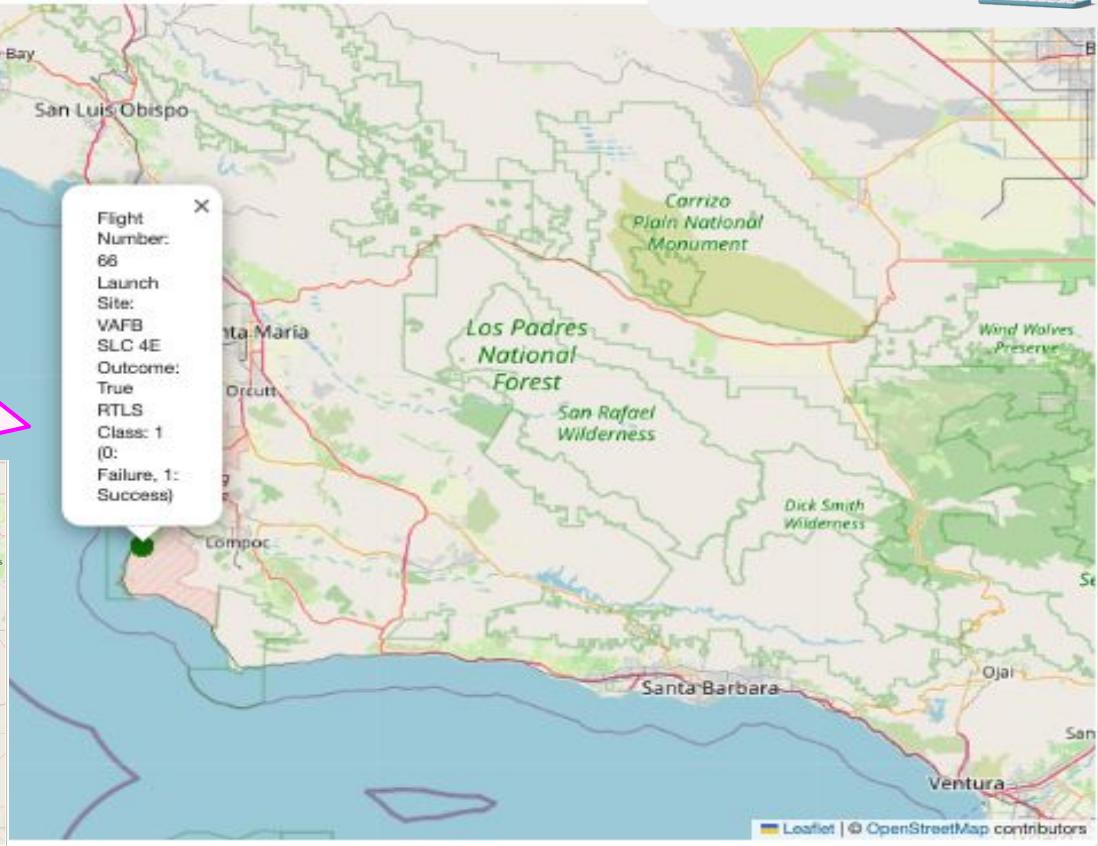
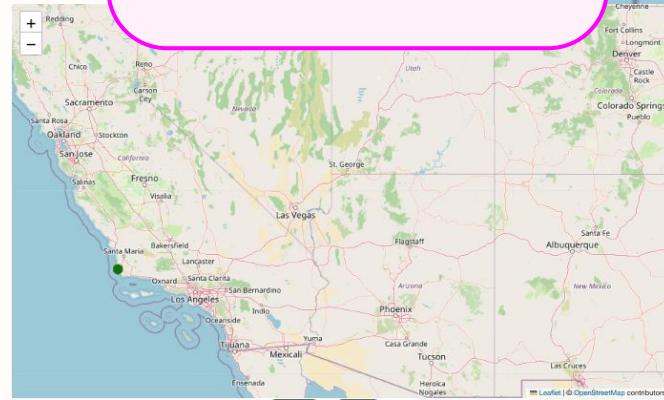
RESULTS

FOLIUM MAP REGIONAL DETAIL

04



Geospatial context
highlights
strategic launch
site placement.



RESULTS

PREDICTIVE ANALYSIS

04



*** Logistic Regression Accuracy: 0.8333333333333334
SVM Accuracy: 0.8333333333333334
Decision Tree Accuracy: 0.7777777777777778
KNN Accuracy: 0.8333333333333334

The method that performs best is Logistic Regression with an accuracy of 0.8333333333333334

90 total launches

Flight numbers 1–90

Payload Mass: 350–15,600 kg

Latitude 28.56–34.63°N (Florida coast)

The model shows zero false negatives and a low false positive rate, indicating reliable success prediction.

```
1 display(data.describe())
```

	FlightNumber	PayloadMass	Flights	Block	ReusedCount	Longitude	Latitude	Class
count	90.000000	90.000000	90.000000	90.000000	90.000000	90.000000	90.000000	90.000000
mean	45.500000	6104.959412	1.788889	3.500000	1.655556	-86.366477	29.449963	0.666667
std	26.124701	4694.671720	1.213172	1.595288	1.710254	14.149518	2.141306	0.474045
min	1.000000	350.000000	1.000000	1.000000	0.000000	-120.610829	28.561857	0.000000
25%	23.250000	2510.750000	1.000000	2.000000	0.000000	-80.603956	28.561857	0.000000
50%	45.500000	4701.500000	1.000000	4.000000	1.000000	-80.577366	28.561857	1.000000
75%	67.750000	8912.750000	2.000000	5.000000	3.000000	-80.577366	28.608058	1.000000
max	90.000000	15600.000000	6.000000	5.000000	5.000000	-80.577366	34.632093	1.000000

RESULTS

PREDICTIVE ANALYSIS

04



▼ TASK 12

Find the method performs best:

```
1 print(f"Logistic Regression Accuracy: {logreg_accuracy}")
2 print(f"SVM Accuracy: {svm_accuracy}")
3 print(f"Decision Tree Accuracy: {tree_accuracy}")
4 print(f"KNN Accuracy: {knn_accuracy}")
5
6 accuracies = {
7     "Logistic Regression": logreg_accuracy,
8     "SVM": svm_accuracy,
9     "Decision Tree": tree_accuracy,
10    "KNN": knn_accuracy
11 }
12
13 best_method = max(accuracies, key=accuracies.get)
14 best_accuracy = accuracies[best_method]
15
16 print(f"\nThe method that performs best is {best_method} with an accuracy of {best_accuracy}")
```

```
... Logistic Regression Accuracy: 0.8333333333333334
SVM Accuracy: 0.8333333333333334
Decision Tree Accuracy: 0.7777777777777778
KNN Accuracy: 0.8333333333333334
```

The method that performs best is Logistic Regression with an accuracy of 0.8333333333333334

RESULTS

PREDICTIVE ANALYSIS

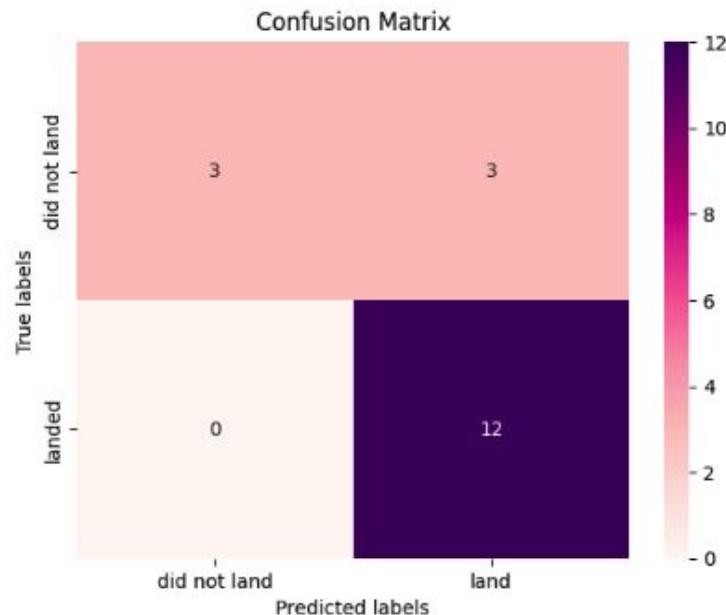
Confusion Matrix for Logistic Regression (LR)

Confusion Matrix for Logistic Regression

04



```
1 yhat_lr = logreg_cv.predict(X_test)  
2 plot_confusion_matrix(Y_test, yhat_lr)
```



RESULTS

04

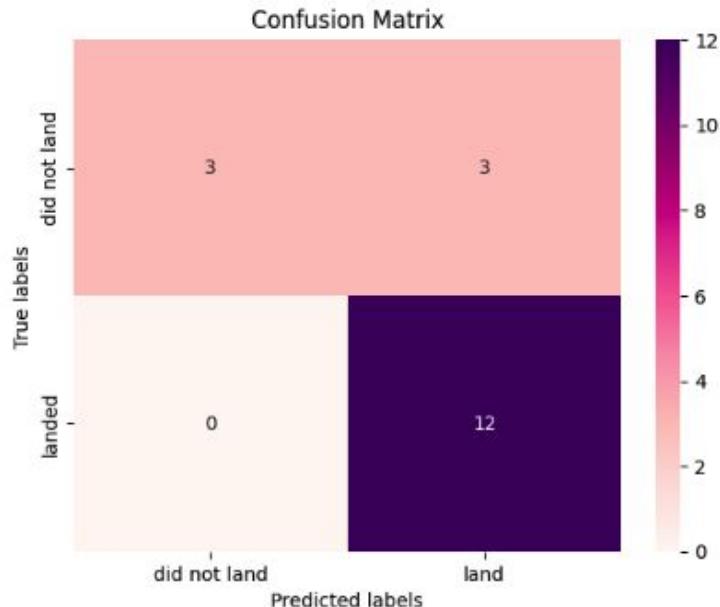


PREDICTIVE ANALYSIS

Confusion Matrix for Support Vector Machine (SVM)

Confusion Matrix for SVM

```
1 yhat_svm = svm_cv.predict(X_test)  
2 plot_confusion_matrix(Y_test, yhat_svm)
```



RESULTS

04

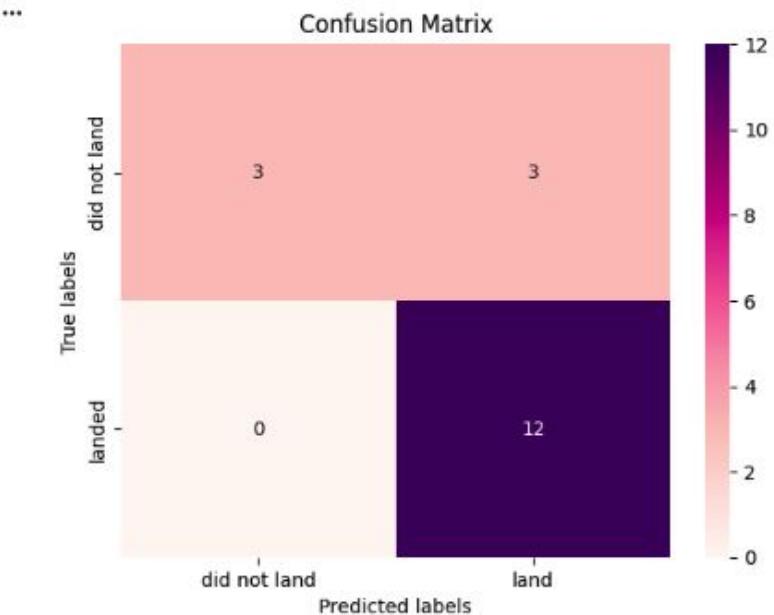
PREDICTIVE ANALYSIS



Confusion Matrix for K-Nearest Neighbors (KNN)

Confusion Matrix for K-Nearest Neighbors

```
1 yhat_knn = knn_cv.predict(X_test)  
2 plot_confusion_matrix(Y_test, yhat_knn)
```





RESULTS

PREDICTIVE ANALYSIS

Model Performance and Best Parameters

Model Training and Best Parameters:

- **Logistic Regression:**
 - Best parameters found: `{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}`
 - Cross-validation accuracy (best score on training data): 0.8464
- **Support Vector Machine (SVM):**
 - Best parameters found: `{'C': 1.0, 'gamma': 0.0316, 'kernel': 'sigmoid'}`
 - Cross-validation accuracy (best score on training data): 0.8482
- **Decision Tree:**
 - Best parameters found: `{'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'random'}`
 - Cross-validation accuracy (best score on training data): 0.8893
- **K-Nearest Neighbors (KNN):**
 - Best parameters found: `{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}`
 - Cross-validation accuracy (best score on training data): 0.8482

RESULTS



PREDICTIVE ANALYSIS

Comparison of Confusion Matrices

All three models (**LR**, **SVM**, and **KNN**) produced identical confusion matrices, resulting in the same accuracy on the test data.

- All three correctly predicted **12** landed outcomes & **3** non-landed outcomes.
- Main issue is the 3 False Positives, where a non-landed event was incorrectly predicted as landed.

LR

True Positives: **12**

True Negatives: **3**

FALSE Positives: **3**

False Negatives: 0

SVM

True Positives: **12**

True Negatives: **3**

False Positives: **3**

False Negatives: 0

KNN

True Positives: **12**

True Negatives: **3**

False Positives: **3**

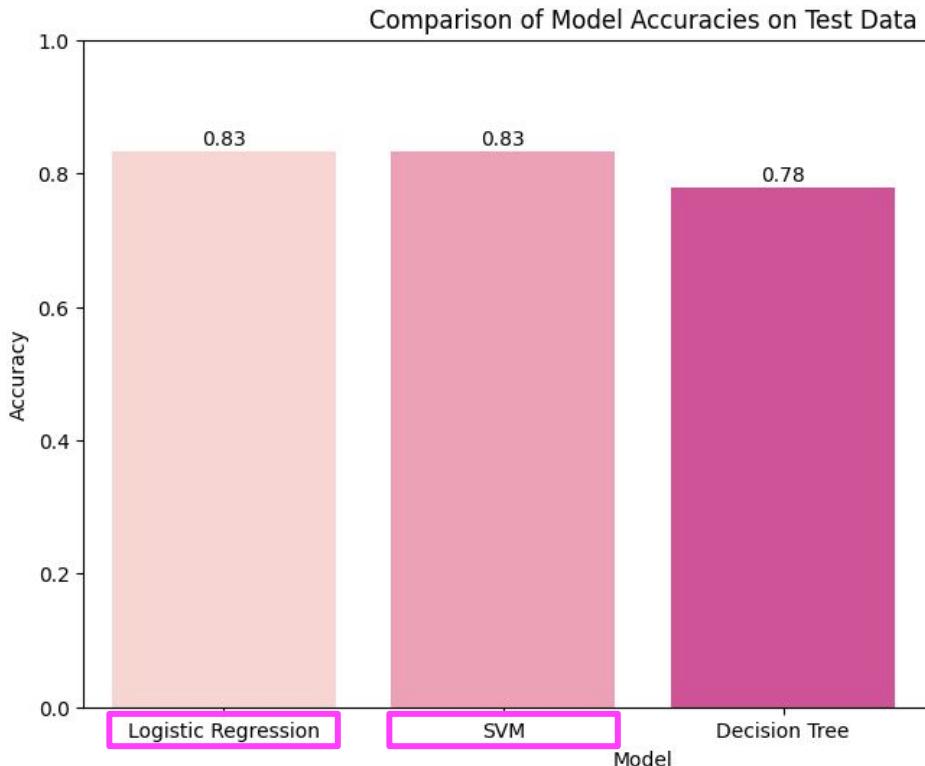
False Negatives: 0

RESULTS

PREDICTIVE ANALYSIS

Model Accuracy Comparison

04



Test Accuracy Results

⭐ 1st Place (Tied)

**Logistic Regression, SVM, KNN @
83.3% Accuracy**
→ Outperforming decision tree

4th Place
Decision Tree @ 77.8% Accuracy



Key Observations

All top models (LR, SVM, KNN) show identical performance patterns



What was the **overall performance** of the predictive models?

Logistic Regression, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) models achieved an identical test accuracy of **83.33%**.

- Decision Tree model performed slightly lower with a test accuracy of 77.78%.



What were the **key challenges** identified in the model predictions?

The top-performing models (Logistic Regression, SVM, KNN) showed a **primary challenge with false positives**, meaning they incorrectly predicted three failed landings as *successful*, while perfectly identifying all successful landings (zero false negatives).

CONCLUSION

DATA ANALYSIS KEY FINDINGS

Summary

- The dataset shows a higher number of successful landings (**Class 1**) compared to failures (**Class 0**), indicating a positive trend in Space X's landing capabilities.
- Early flights had a higher rate of failure, but the **success rate** generally **improved** as the flight number increased.
- Certain orbit types, specifically **ES-L1**, **GEO**, **HEO**, and **SSO**, demonstrated 100% landing success rates within the dataset.
- Launch sites **KSC LC 39A** and **VAFB SLC 4E** generally exhibited higher success rates compared to **CCAFS SLC 40**.



CONCLUSION

DATA ANALYSIS KEY FINDINGS

Summary

- The interactive Folium map visually confirmed the concentration of launch activities at specific sites and provided a geographical perspective on landing outcomes.
- For Logistic Regression, SVM, and KNN, the best hyperparameters on the training data resulted in cross-validation accuracies of **0.8464**, **0.8482**, and **0.8482**, respectively.
 - ◆ Decision Tree had a best cross-validation accuracy of 0.8893.
- On the test set, **Logistic Regression, SVM, and KNN models** achieved 12 true positives, 3 true negatives, 3 false positives, and 0 false negatives.



CONCLUSION

This capstone demonstrates how data science ***improves aerospace decision-making.***

EDA, SQL analysis, interactive maps, and machine learning → collectively enable

- Accurate Landing Predictions
- Cost Savings



CONCLUSION

Project Success

Built end-to-end ML pipeline to predict Falcon 9 first-stage landing outcomes.

Key Findings

- ~83% test accuracy with Logistic Regression, SVM, KNN
- Launch site, orbit type, flight number are strong predictors
- Reused boosters have higher success rates
- Zero false negatives: Model is conservative

Challenges

False positives (3/18); Limited by dataset size and features.



INSIGHTS ➤

Business Application

- ❑ Competitive Advantage Model for cost estimation
- ❑ Bidding Strategy for rocket launch pricing
- ❑ Risk Assessment for mission success prediction

Innovative Insights

- ❑ 100% Success Orbits: ES-L1, GEO, HEO, SSO show perfect records
- ❑ Time-Based Improvement: SpaceX success 50% (2010) → 85%+ (2020+)
- ❑ Site Maturity Effect: Newer facilities outperform older infrastructure

→ Future efforts should focus on techniques to **reduce** false positives in the models, potentially by gathering **more balanced data, engineering more discriminating features, or exploring advanced model architectures and ensemble methods.**

→ The observed improvement in landing success with increasing Flight Number highlights Space X's continuous learning and technological advancements, suggesting that **newer data might further improve predictive model performance.**

IMPACT & NEXT STEPS ➤

- ❑ The detailed description provides a clear blueprint for the actual development of the Plotly Dash dashboard, [outlining its features & intended benefits](#).
- ❑ The next logical step would be to proceed with the implementation of this described Plotly Dash dashboard, [leveraging the EDA and predictive models developed in the notebook](#).

APPENDIX

This capstone demonstrates **mastery** of data science:

- Problem framing
- EDA
- SQL
- Modeling
- Communication



IBM Data Science **Capstone**



Megan A. Flores, M.B.S.

Thank you
for your time & attention.

