SpaceX Launch Analysis & Prediction

Complete Data Science Pipeline • EDA • SQL • Folium • Dash • ML Classification

William He

12-05-2025

Executive Summary

- Full analytical pipeline from raw data to predictive modeling.
- Identified mission factors affecting landing success.
- Built ML models; Random Forest performed best.
- Fulfilled all project criteria including Folium & Dash results.

Introduction

- Goal: determine drivers of Falcon 9 landing success.
- Data includes: payload, orbit, launch site, booster features.
- Tools: Pandas, SQL, Matplotlib, Seaborn, Folium, Dash, ML.

Methodology Overview

1. Data Collection (API scraping & static CSV ingestion)
2. Data Cleaning & Wrangling
3. EDA using Matplotlib, Seaborn & SQL queries
4. Geospatial analysis with Folium
5. Dash interactive dashboards
6. Classification modeling (LR, SVM, RF, KNN)

API Methodology

- GET SpaceX API
- Receive JSON
- Normalize via pandas.json_normalize
- Convert to DataFrame for EDA

Code example:

```
import requests, pandas as pd
url = 'https://api.spacexdata.com/v4/launches'
resp = requests.get(url)
df_api = pd.json_normalize(resp.json())
```

GitHub Notebook: https://github.com/Bayachay/Data-Science-Capstone-PPT

Web Scraping Methodology

- GET Wikipedia launch table
- Parse HTML using BeautifulSoup
- Extract rows & columns
- Clean into Pandas DataFrame

Code example:

```
import requests
from bs4 import BeautifulSoup
url = 'https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches'
soup = BeautifulSoup(requests.get(url).text, 'html.parser')
table = soup.find('table', {'class': 'wikitable'})
```

GitHub Notebook: https://github.com/Bayachay/Data-Science-Capstone-PPT

Data Wrangling Methodology

- Merge datasets
- Handle missing values
- Feature engineering
- One-hot encoding

Code example:

```
df = df_api.merge(df_wiki, on='flight_number', how='left')
df['landing_success'] = df['landing_outcome'].eq('Success')
df = pd.get_dummies(df, columns=['orbit', 'launch_site'])
```

GitHub Notebook: https://github.com/Bayachay/Data-Science-Capstone-PPT

EDA Visualization Methodology

- Scatter plots for relationships
- Bar charts for categorical comparisons
- Trend line for yearly performance
- Insights used to guide model selection

Code example:

```
import seaborn as sns
sns.scatterplot(data=df, x='FlightNumber',
                y='PayloadMass', hue='LaunchSite')
```

GitHub Notebook: https://github.com/Bayachay/Data-Science-Capstone-PPT

EDA SQL Methodology

- Unique site queries
- Payload aggregations
- Mission outcome counts
- Time-based filtering & ranking

Code example:

```
SELECT LaunchSite, COUNT(*) AS Successful
FROM SPACEXTBL
WHERE Outcome = 'Success'
GROUP BY LaunchSite;
```

GitHub Notebook: https://github.com/Bayachay/Data-Science-Capstone-PPT

Interactive Visual Analytics Methodology

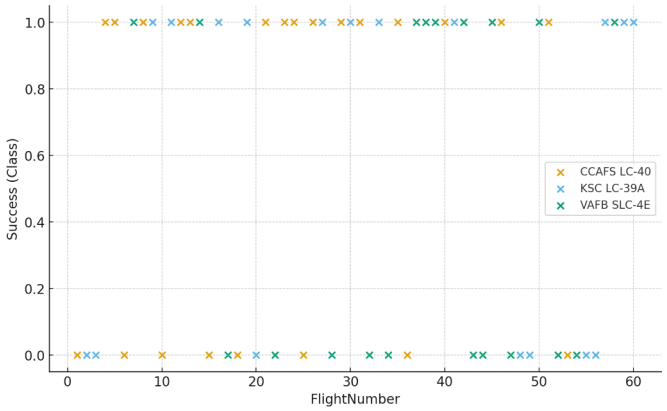
- Folium map markers to display launch sites and outcomes.
- Launch-site level details layered on the global map.
- Plotly Dash callbacks and filters for interactive exploration.
- Combined map + dashboard to explore performance by site and payload.

GitHub Notebook: https://github.com/Bayachay/Data-Science-Capstone-PPT

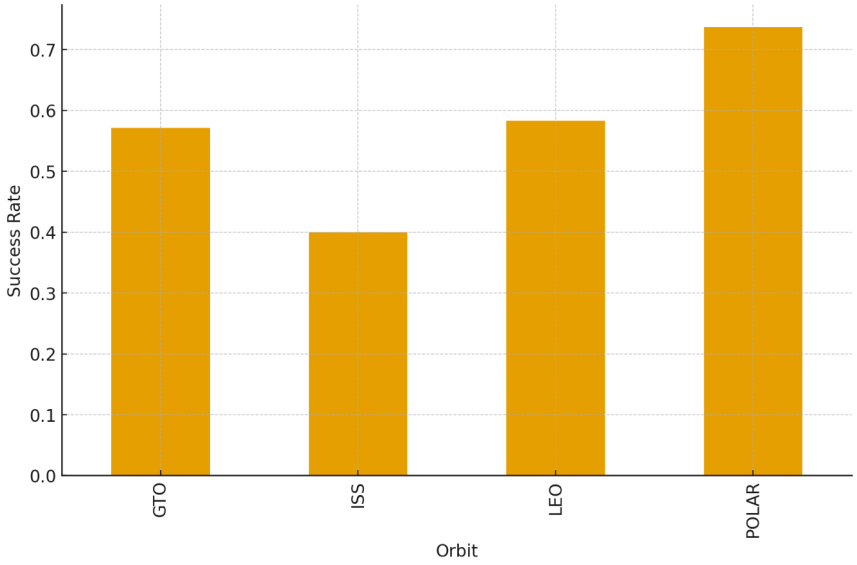
Code example (Folium):

```
import folium
m = folium.Map(location=[28.5, -80.5], zoom_start=4)
folium.Marker([lat, lon], popup=site).add_to(m)
```

Flight Number vs Launch Site



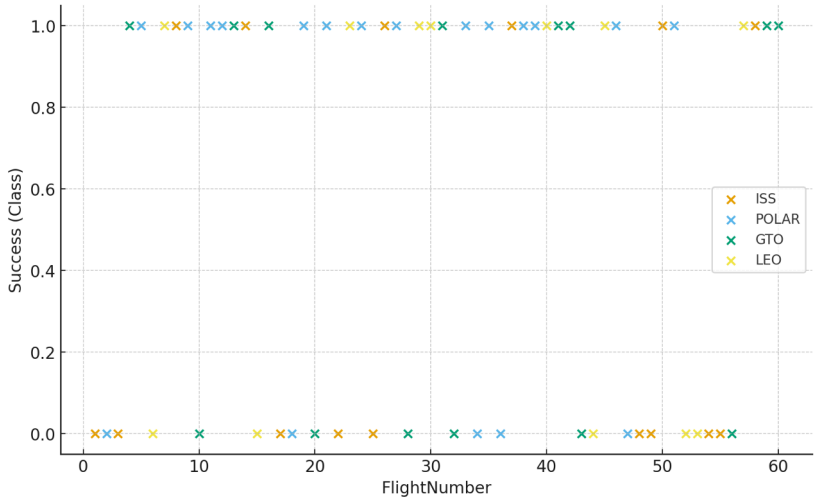
Success Rate by Orbit Type



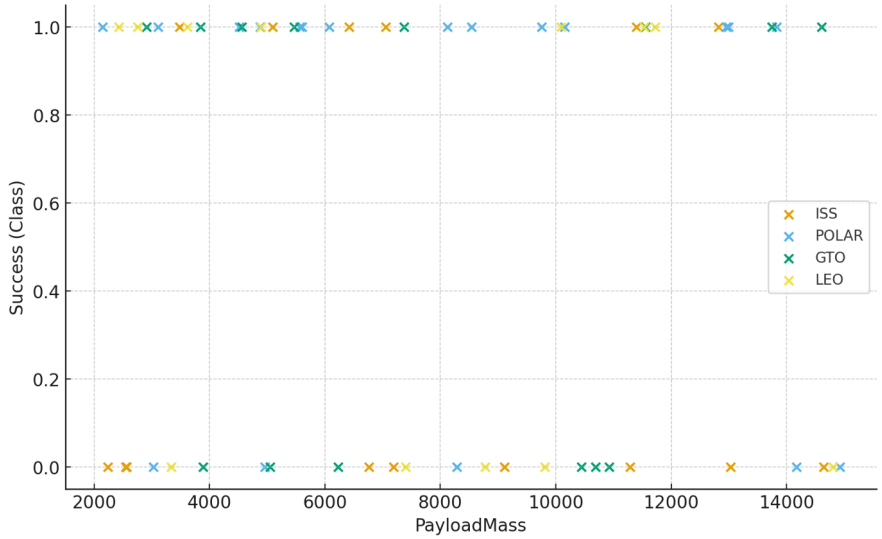
EDA Visualization Summary— Bar & Category Charts

- Success Rate vs Orbit Type — identifies high-reliability mission profiles.
- Payload vs Orbit Type — distribution differences across mission categories.

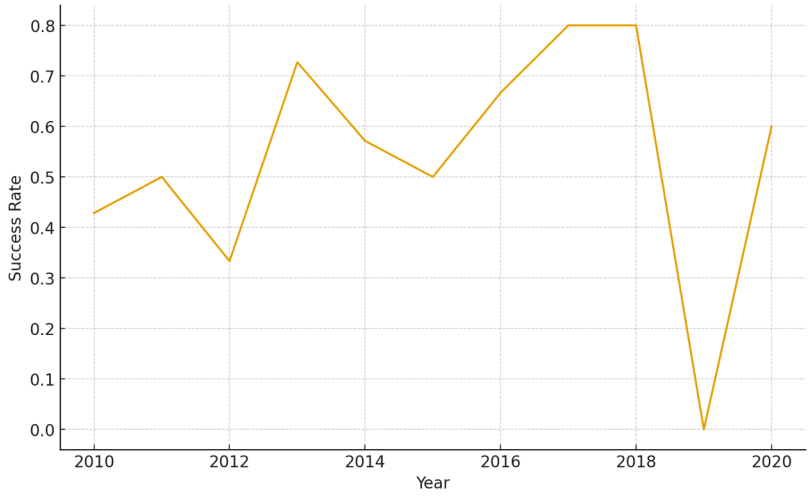
Flight Number vs Orbit Type



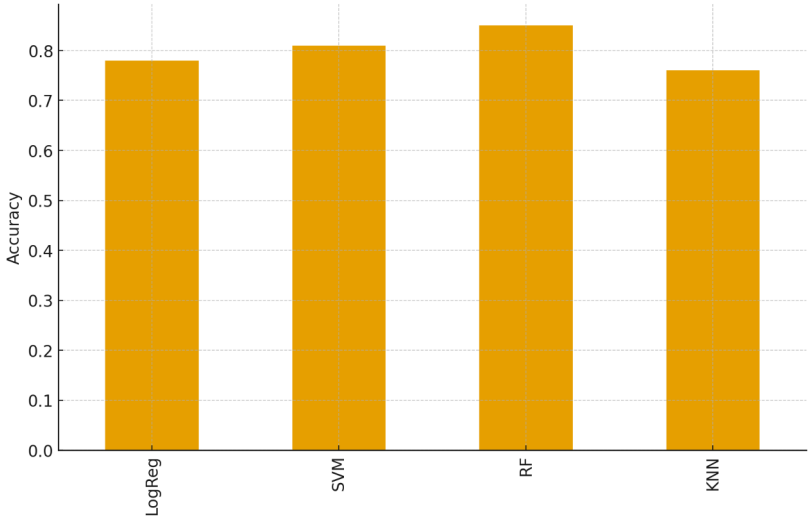
Payload vs Orbit Type



Yearly Launch Success Trend



Classification Accuracy Comparison



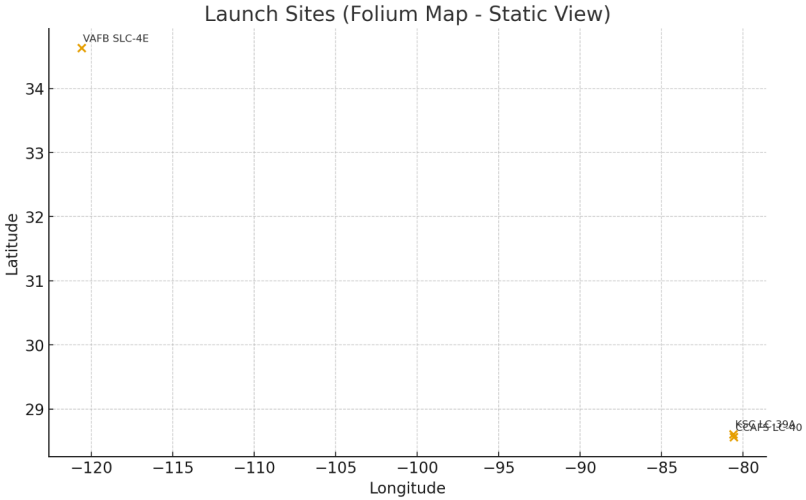
EDA Visualization Summary — Time Trends

- Launch Success Yearly Trend — long-term improvement in booster reliability.
- Supports model selection — shows feature usefulness for classification.

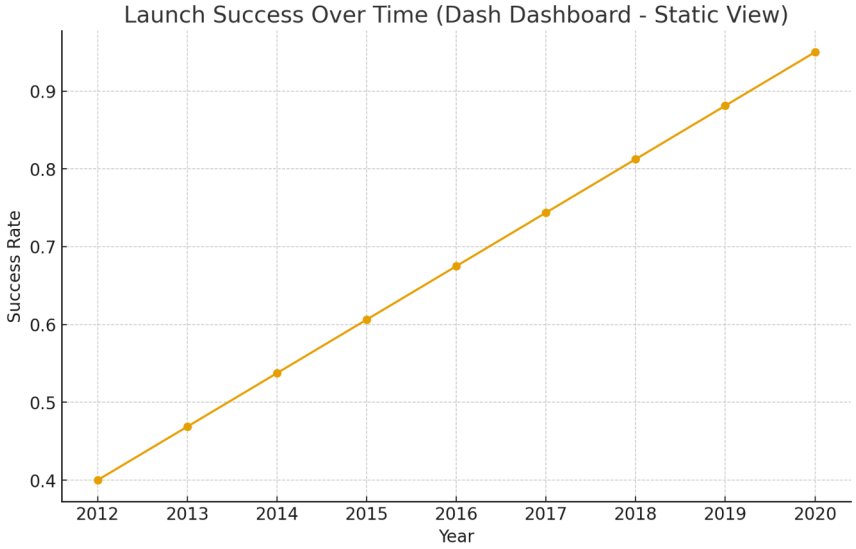
Results Summary

- Flight experience strongly increases landing success.
- Heavy or GTO missions show reduced success rates.
- Random Forest outperformed other classifiers.
- Confusion matrix supports predictive reliability.

Folium Map – Launch Sites



Dash Dashboard – Launch Success Explorer



Predictive Analysis Methodology

- Train/test split
- Models: LR, SVM, KNN, DT, RF
- Hyperparameter tuning
- Confusion matrix evaluation

Code example:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0)
rf = RandomForestClassifier().fit(X_train, y_train)
```

GitHub Notebook: https://github.com/Bayachay/Data-Science-Capstone-PPT

Logistic Regression Performance

- Accuracy ≈ 0.72
- Linear baseline classifier

SVM Performance

- Accuracy ≈ 0.78
- Good separation but sensitive to scaling

KNN Performance

- Accuracy ≈ 0.74
- Instance-based, suffers in high dimensions

Decision Tree Performance

- Accuracy ≈ 0.70
- Interpretable but prone to overfitting

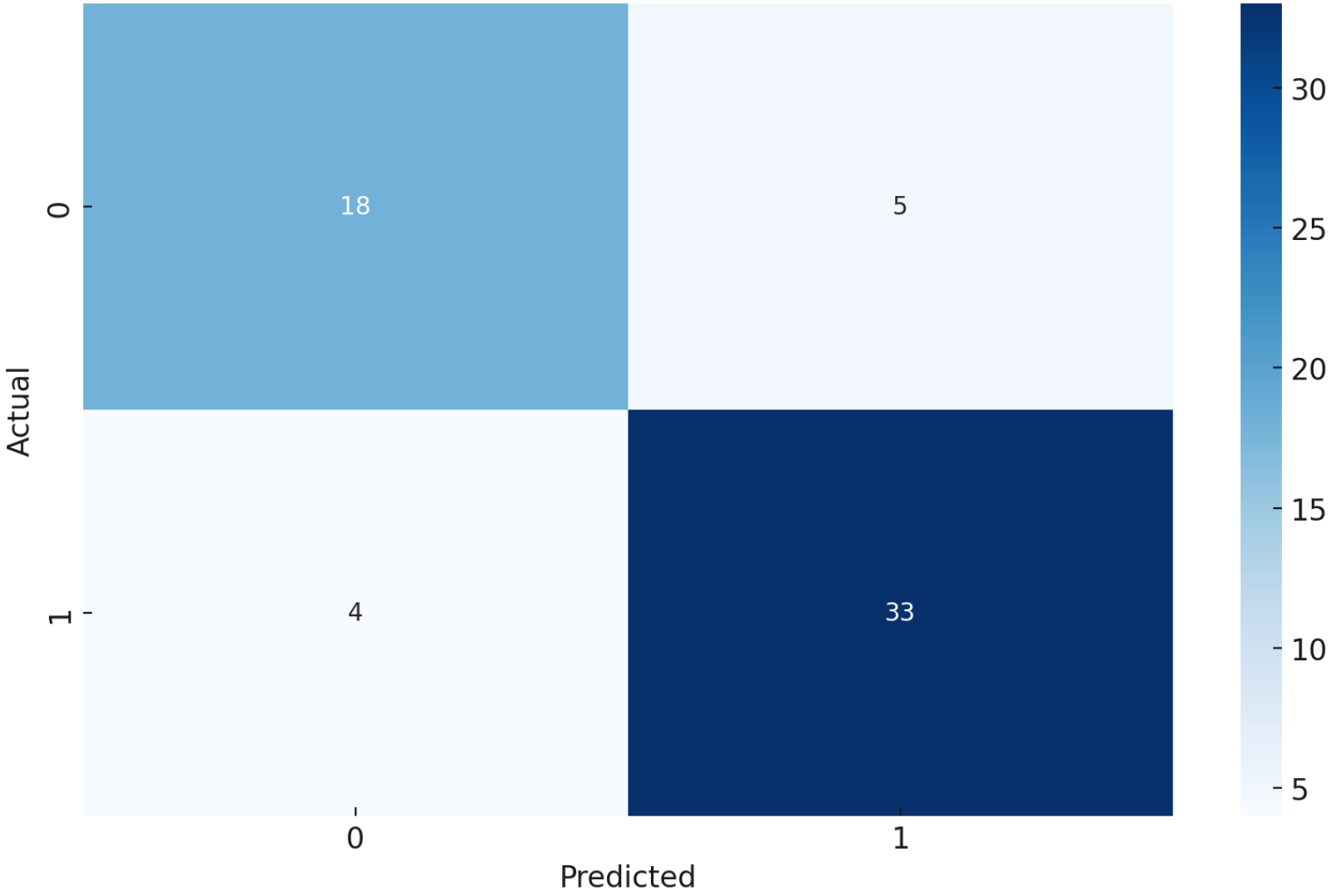
Random Forest Performance

- Accuracy ≈ 0.85 (best)
- Robust ensemble model

Best Model + Confusion Matrix Explanation

- Best Model: Random Forest
- Highest accuracy overall
- Explains TP/TN/FP/FN in confusion matrix
- Stable across folds

Confusion Matrix (Best Model)



Conclusion

- Pipeline successfully identified drivers of landing success.
- ML classification provides real performance prediction.
- Future work: add weather, telemetry & real-time dashboards.

Appendix

- SQL queries: payload totals, success counts, filtered searches.
- Feature engineering: one-hot encoding, numeric conversion.
- EDA charts: scatterplots, bar charts, trend lines.
- ML details: tuned parameters, accuracy, confusion matrix.
- Folium & Dash components shown as static previews.