DATA SCIENCE CAPSTONE PROJECT

IJAZ AHMED
01.09.2025

OUTLINE

EXECUTIVE SUMMARY

Studied factors influencing Falcon 9 launch success by collecting and preparing data from APIs and web scraping, engineered features for predictive machine learning models, and developed an interactive dashboard for live analytics and visualization

INTRODUCTION

SpaceX has made space travel far more affordable – $62M per launch vs. $165M for competitors – mainly because the first stage is reusable.

Using Machine Learning models, our goal is to identify the best predictive model for determining when the first stage can be reused successfully.

Key Questions:

- How often is reuse successful?
- Which factors (payload, orbit, launch site, booster version, etc.) influence the outcome?

METHODOLOGY

Data Collection

- APIs (SpaceX API)
- Web Scraping (Wikipedia)

Data Preparation

- Cleaning, handling missing values
- Feature engineering

Exploratory Data Analysis (EDA)

- Python (Pandas, Seaborn, Matplotlib)
- SQL (SQLite queries)

Predictive Analysis

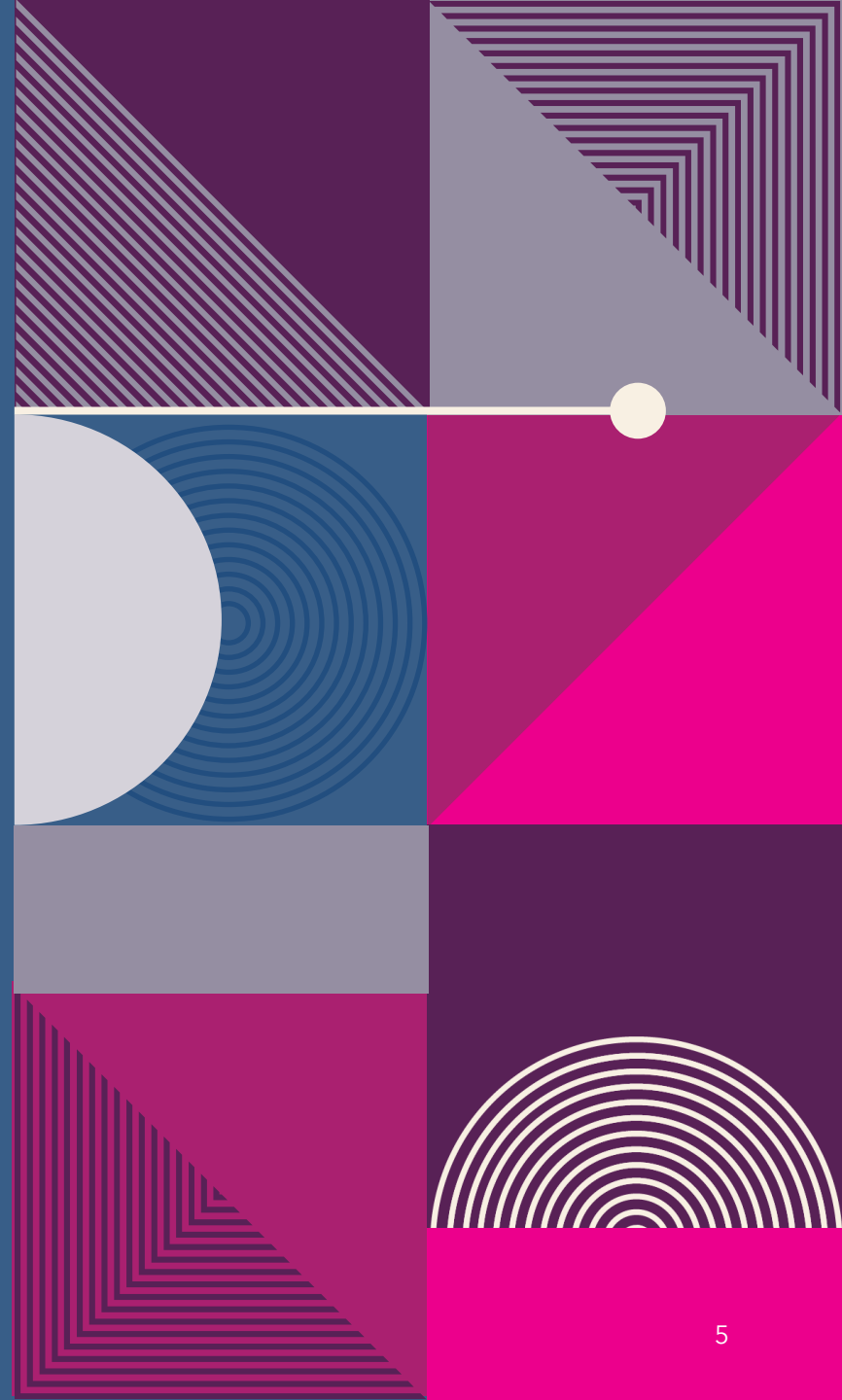
- Apply ML Models (Logistic Regression, SVM, Decision Trees)

Model Evaluation

- Compare performance and select the best model

Interactive Visualizations

- Folium (geospatial mapping)
- Plotly Dash (live analytics dashboard)



DATA ACCUMULATION

API Calls

- Collected launch data using SpaceX REST API
- Retrieved structured JSON responses with launch details (rocket, payload, orbit, landing outcome)
- Converted JSON into Pandas DataFrame for analysis

Web Scraping

- Used requests module to fetch HTML pages (e.g., Wikipedia Falcon 9 launches)
- Parsed HTML with BeautifulSoup (html.parser)
- Extracted tables containing launch history and outcomes
- Combined with API data for a richer dataset

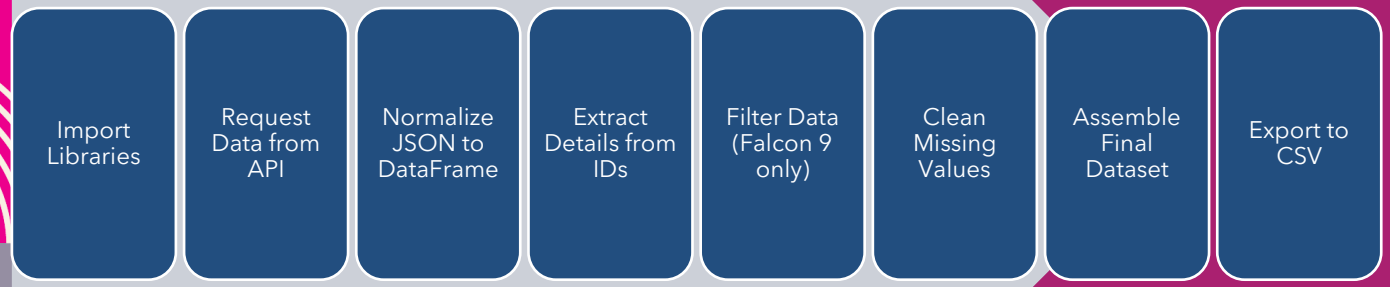
DATA WRANGLING

After collecting raw data, several data wrangling steps were performed. We normalized JSON responses into pandas DataFrames, removed rows with multiple cores or payloads, and restricted the dataset to Falcon 9 launches only.

Missing values were carefully handled: PayloadMass missing entries were replaced with the column mean, while LandingPad retained None values to indicate planned ocean landings. Additional features such as FlightNumber, Block, and ReusedCount were engineered to capture booster reliability and technological improvements.

The final dataset was saved as CSV files to be reused across subsequent labs.

DATA COLLECTION:



WEB SCRAPING

Importing Libraries

requests → send HTTP requests and fetch webpage/API data

BeautifulSoup → parse and extract elements from HTMLpandas → organize data into DataFrames for analysis

GETTING HTTP RESPONSE

Send HTTP Request

Use requests.get(url) to retrieve webpage content.

Example: scraping Wikipedia page on Falcon 9 launches.

Response codes (200 = success).

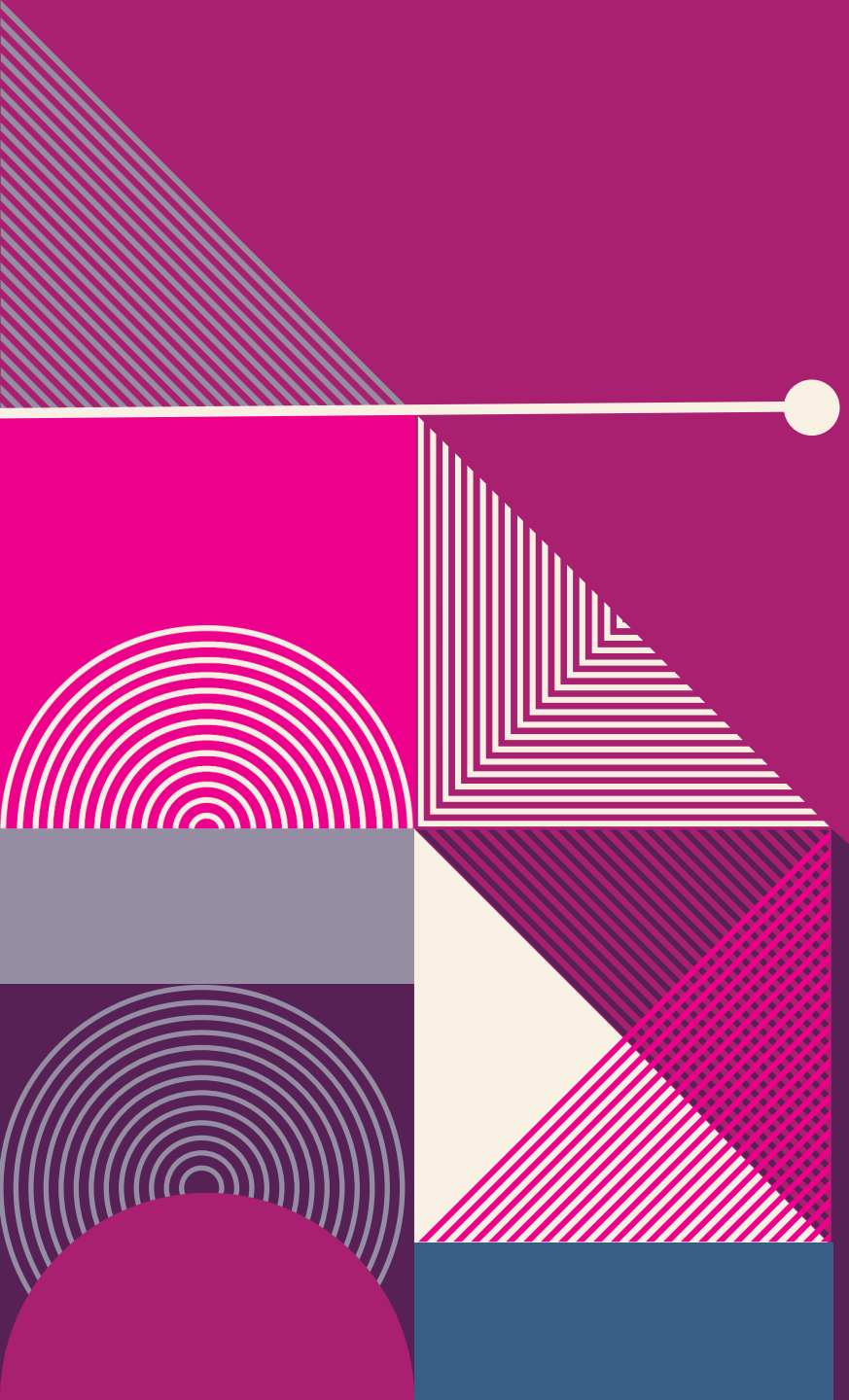
We initially get 403 error for scraping Wikipedia which can be easily resolved using a User-Agent.

PARSE HTML CONTENT

- Load raw HTML into **BeautifulSoup**.
- Locate elements with tags (<table>, <tr>, <td>) or CSS selectors.
- Example: extracting launch tables from Wikipedia.

EXTRACT DATA ELEMENTS

Identify relevant features: Date, Booster version, Payload mass, Orbit, Landing outcome.
Loop through table rows to collect structured values.



BUILD DATAFRAME

- Organize extracted data into a **pandas DataFrame**.
- Ensures tabular structure for analysis.

CLEAN & FORMAT DATA

Convert dates → datetime format.

Handle missing values (NaN replacements).

Convert numeric columns (e.g., PayloadMass) to float.

SAVE DATASET (CSV)

We save csv files for every notebook wherever possible

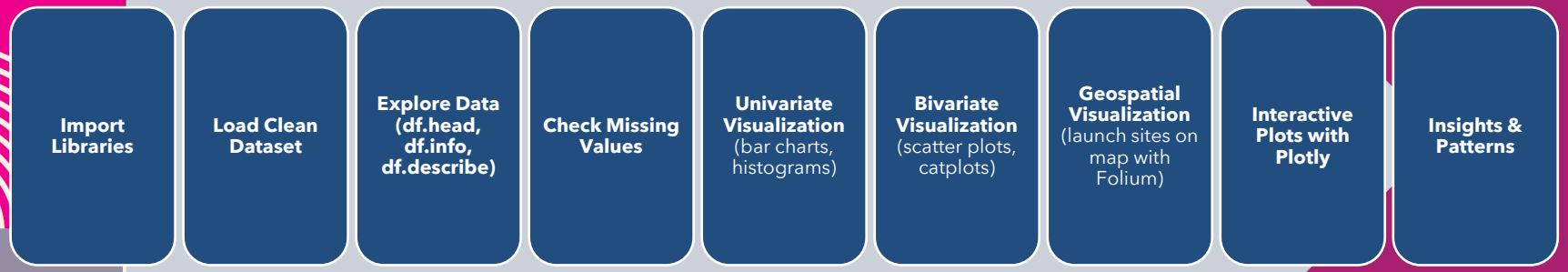
EXPLORATORY DATA ANALYSIS

Focused on Falcon9

Fixed the null values

Extracted info for each Orbit, Launch Site Payload

PROCESS - EDA


Import Libraries
Load Clean Dataset
Explore Data (df.head, df.info, df.describe)
Check Missing Values
Univariate Visualization (bar charts, histograms)
Bivariate Visualization (scatter plots, catplots)
Geospatial Visualization (launch sites on map with Folium)
Interactive Plots with Plotly
Insights & Patterns

EXPLORE DATA

EDA was conducted both in Python and SQL.
Using pandas, matplotlib, and seaborn, we examined the distributions of payload masses, the frequency of launches from different sites, and the relationship between orbits and landing success.
SQL queries were used for summarization, such as calculating the average payload mass by orbit or counting the number of successful landings per launch site.
Visualization tools such as Folium helped us map launch locations geographically, while Plotly allowed interactive exploration of payload vs. success probability.
Key insights showed that success rates improved significantly over time, heavier payloads influenced outcomes, and certain launch sites consistently performed better.

FIXING MISSING VALUES

Ensured completeness.
Only certain categorical columns

VISUALIZATIONS - EDA

Bar charts: success counts by orbit, launch site.

Histograms: payload distribution.

Showed which categories dominate.

Scatter plots / catplots: FlightNumber vs. PayloadMass colored by success.

Identified patterns: later flights → higher success

FOLIUM AND PLOTLY

Folium maps to display launch sites with markers.

Circle markers and clusters to explore success patterns geographically.

Dropdown menus to filter by orbit or launch site.

Scatter and line plots showing success by payload/flight.

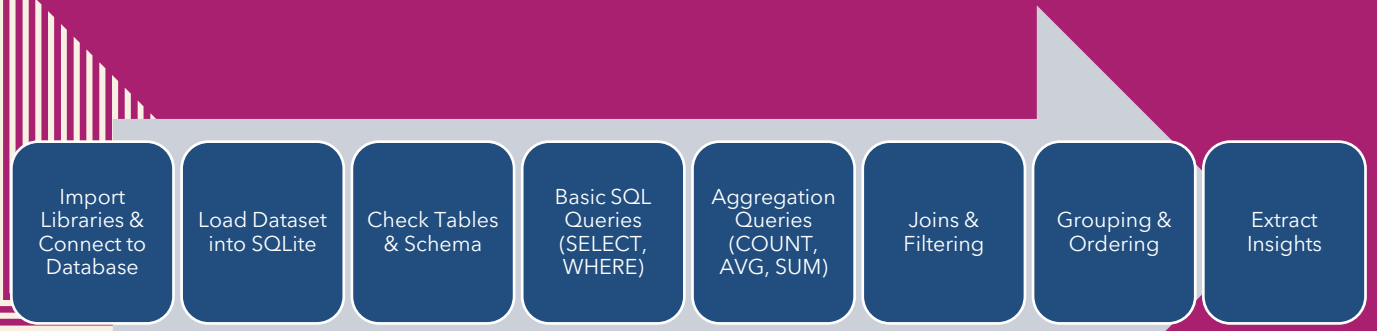
EDA - INSIGHTS

Success rate improved as FlightNumber increased.

Higher payloads correlated with different orbits (e.g., GTO riskier than LEO).

Certain launch sites performed more reliably.

EDA - SQL


Import Libraries & Connect to Database
Load Dataset into SQLite
Check Tables & Schema
Basic SQL Queries (SELECT, WHERE)
Aggregation Queries (COUNT, AVG, SUM)
Joins & Filtering
Grouping & Ordering
Extract Insights

Import Libraries & Connect to Database

Used sqlite3 and pandas for database connection and query execution.

Created a cursor for running SQL commands.

Load Dataset into SQLite

Loaded cleaned Falcon 9 dataset (dataset_part_2.csv).

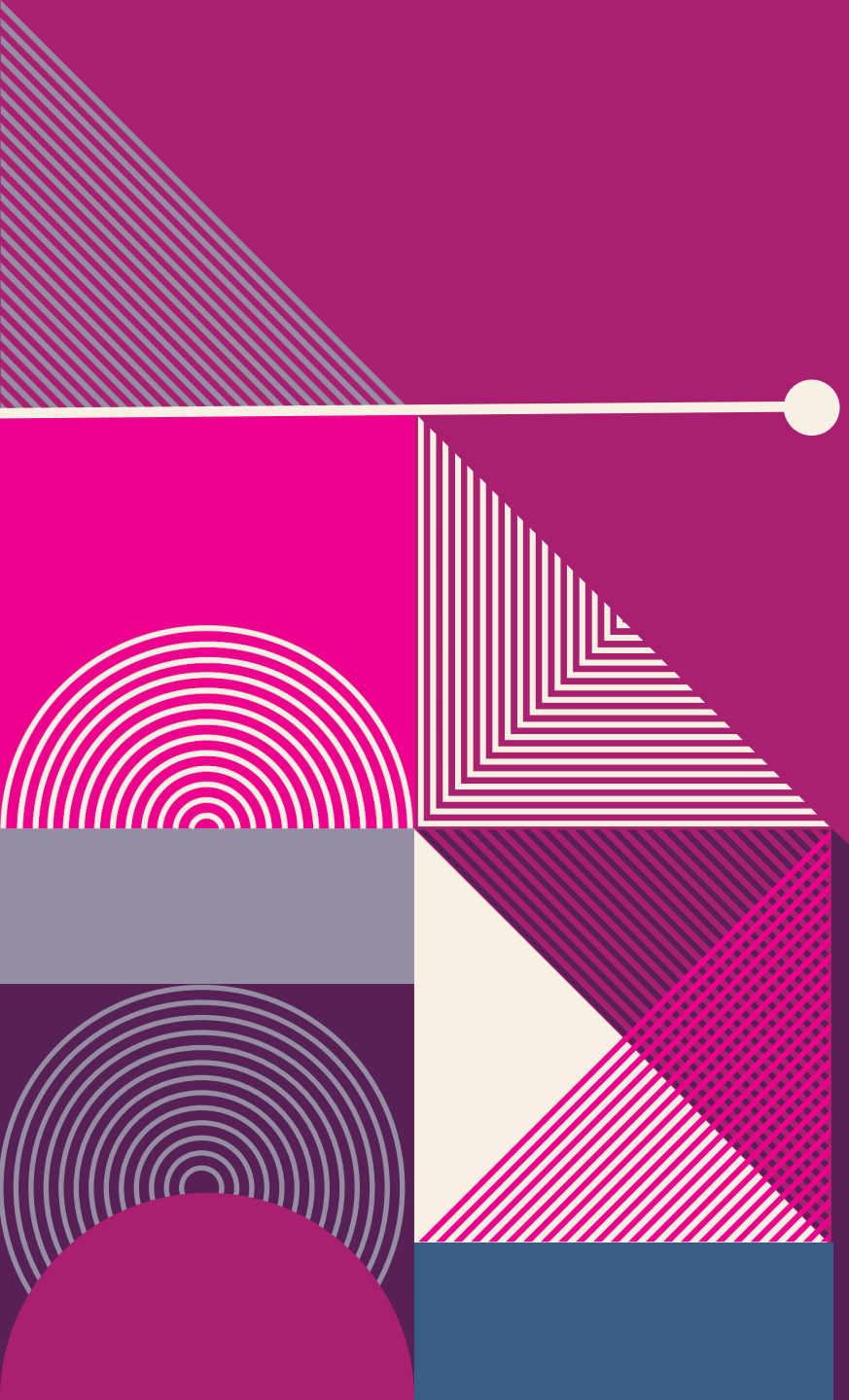
Stored in an SQLite database for query-based exploration.

```
%sql SELECT * FROM SPACEXTBL
```

Python

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	L
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	F
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere	0	LEO (ISS)	NASA (COTS) NRO	Success	F



Check Tables & Schema

Verified database structure with SELECT name FROM sqlite_master.

Checked table column names and data types.

Basic SQL Queries

Queried launch data (flight number, payload, orbit).

Applied WHERE conditions to filter by orbit or launch site.

Display the names of the unique launch sites in the space mission

```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE
```

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

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

Aggregation Queries

Used COUNT() for number of launches.

Used AVG() for mean payload mass.

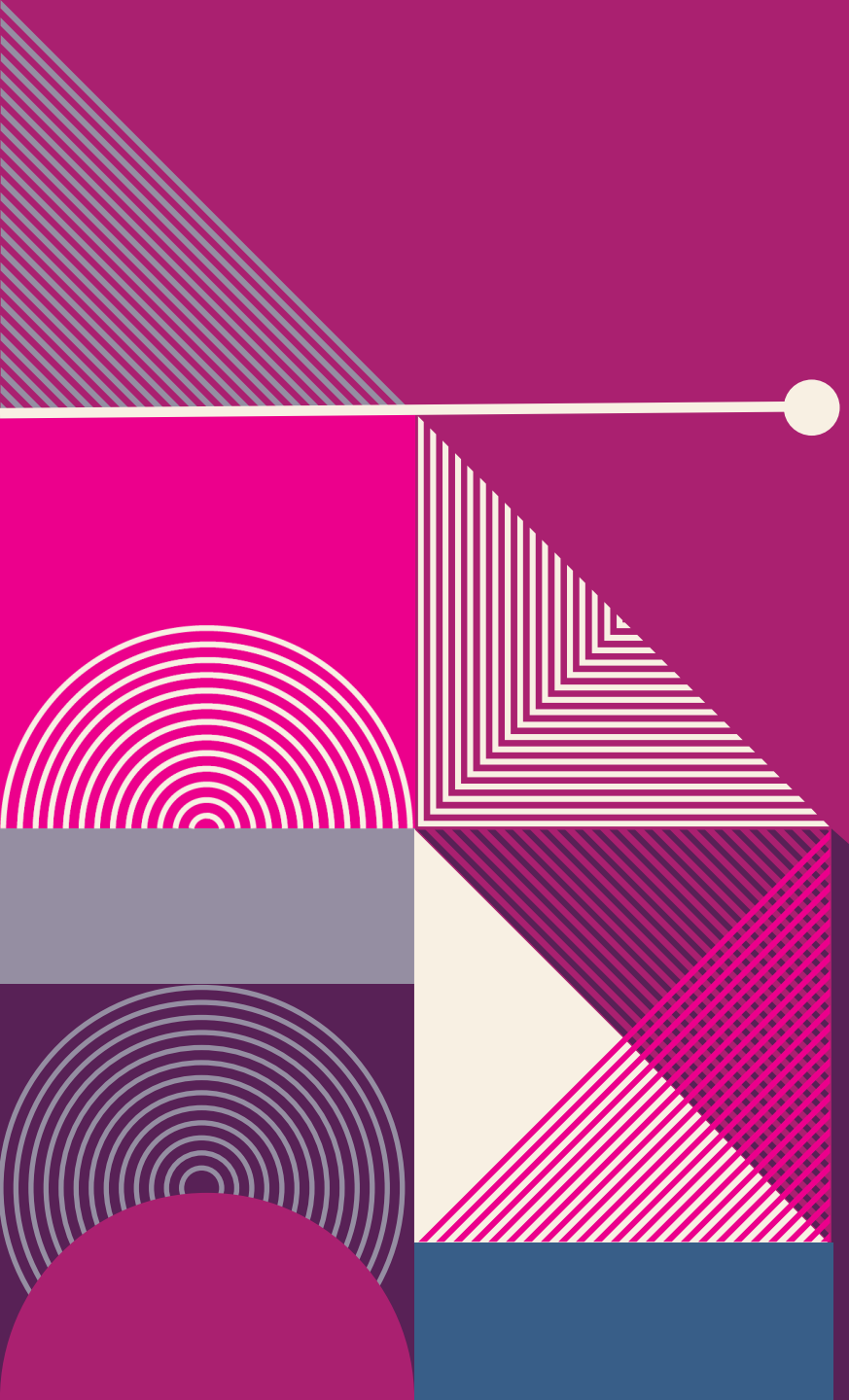
Summarized launch outcomes.

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

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX
```

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

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



Joins & Filtering

Joined tables (if split into payloads/launches).

Filtered data to answer specific business questions.

Grouping & Ordering

Grouped results by orbit, launch site, and booster version.

Ordered by payload mass, success rate, or flight number.

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.

```
%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;
```

Python

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

Landing_Outcome	Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5

SQL EDA - INSIGHTS

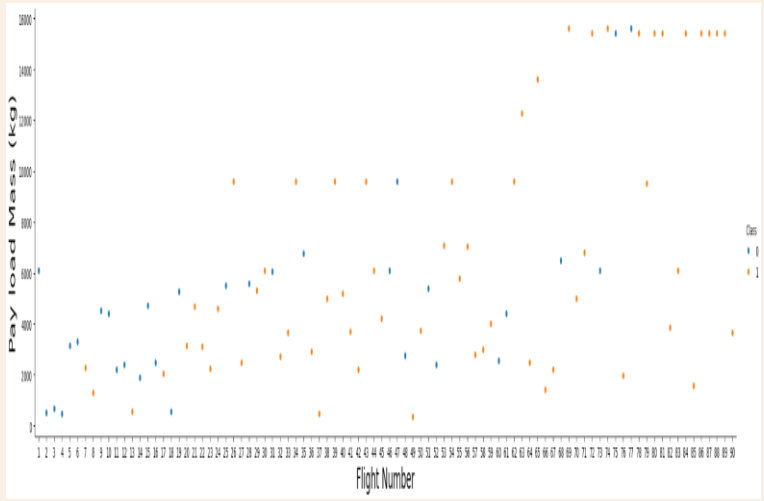
Found relationships between payload mass, orbit, and success rate.

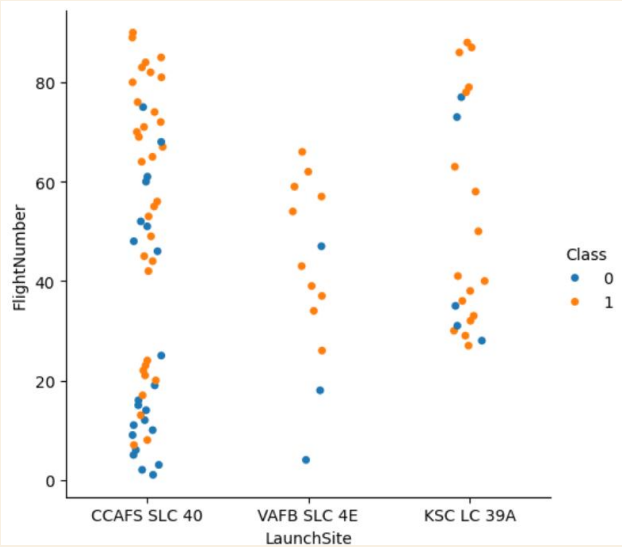
Identified which launch sites and booster versions were most reliable.

EDA: CHARTS

Scatter Plots were plotted to check the correlation of variables with each other:

- Flight Number
- Payload Mass
- Launch Site
- Orbit
- Success Rate, and Yearly trend for the success





EDA: WITH SQL

Explored further patterns within the data for payload (4000<payload<6000)

Successful missions and their relevant parameters

Ranking the Landing outcomes for drones and groundpad

Majority of missions ended in **successful landings**, showing steady improvement.

NASA (CRS) launches carry large payloads; booster F9 v1.1 has moderate average mass.

Success on ground pad and *Success on drone ship* became increasingly frequent after 2015.

Failures were concentrated in specific months, showing early challenges.

SQL provided clear insights into launch sites, payloads, booster versions, and landing outcomes.Analysis shows improvement in mission success rates over time.SQL is effective for answering business questions on reliability and cost prediction

INTERACTIVE FOLIUM MAP

- The Project also included a map in Folium, to interactively see the sites and their outcomes using Markers

INTERACTIVE DASHBOARD

- To make results accessible, we created interactive visualizations. Folium was used to map launch sites with markers showing successful and failed landings.
- Plotly Dash provided a dashboard that allowed users to filter launches by orbit, payload mass, or site and instantly see predicted probabilities.
- This dashboard simulates how stakeholders could use the model in real-time to evaluate upcoming launches.

PREDICTIVE MODELING

- To answer the central question, i.e. Can we predict if the Falcon 9 first stage will land successfully?
- We applied multiple machine learning models.
- Logistic Regression, Support Vector Machines (SVM), and Decision Trees were trained using scikit-learn.
- The data was standardized using StandardScaler and split into training and test sets (80/20).
- Hyperparameter tuning with GridSearchCV ensured optimal model performance.
- Cross-validation accuracy indicated that Decision Trees achieved the highest score during training, but they overfit and underperformed on the test set.
- Logistic Regression and SVM, on the other hand, produced balanced performance, achieving around 83% accuracy on test data.

PREDICTIVE ANALYSIS

- To predict the landing success of Falcon 9's first stage, we trained multiple classification models using GridSearchCV with 10-fold cross-validation. This ensured that hyperparameters were optimized systematically and model performance was validated across multiple folds of the training set.
- The models evaluated were:
- Logistic Regression → A baseline linear classifier that is simple, interpretable, and effective for binary classification tasks.
- Decision Tree Classifier → A non-linear model capable of capturing complex relationships in the data through hierarchical splits.
- Support Vector Machine (SVM) → A powerful model that uses hyperplanes and kernels to separate classes, especially effective in high-dimensional feature spaces.
- K-Nearest Neighbors (KNN) → An instance-based learning algorithm that classifies new samples based on the majority vote of their closest neighbors.
- Each model was tuned with its relevant hyperparameters. For example, Logistic Regression's regularization parameter C and solver were optimized, Decision Trees were tuned with depth, splitter, and minimum sample requirements, SVM was tested with different kernels and gamma values, and KNN was adjusted for different values of k, algorithms, and distance metrics.
- This multi-model approach allowed us to not only identify the best performing algorithm but also compare their strengths, weaknesses, and suitability for deployment in predicting Falcon 9 landing outcomes.

PREDICTIVE MODELS: OBJECTIVE

- Prepare data for training:
- Create labels (Class column)
- Standardize features
- Split into training & test sets
- Train and tune ML models: Logistic Regression, SVM, Decision Tree, KNN
- Compare performance and find the best model

PREDICTIVE ANALYSIS: DATA PREP

- Labels extracted from Class column → NumPy array Y
- Features (X) standardized with StandardScaler
- Data split: 80% train / 20% test (random_state=2)
- Test set size: 18 records

LOGISTIC REGRESSION

- Tuned hyperparameters: C, penalty, solver
- Best parameters found via GridSearchCV (cv=10)
- CV accuracy: ~84%
- Test accuracy: ~83%
- Confusion Matrix:
- True Positives: 12
- False Positives: 3

SUPPORT VECTOR MACHINE

- Kernels tested: linear, rbf, poly, **sigmoid**
- Hyperparameters tuned: C, gamma
- CV accuracy: ~85%
- Test accuracy: ~83%
- Confusion Matrix showed good separation with some misclassifications

DECISION TREE

- For the Decision Tree model, we performed hyperparameter tuning using GridSearchCV with 10-fold cross-validation. The parameters that were optimized included:criterion: The function to measure the quality of a split (gini vs entropy).splitter: The strategy used to choose the split at each node (best vs random).max_depth: The maximum depth of the tree, which controls model complexity.max_features: The number of features considered when looking for the best split.min_samples_split / min_samples_leaf: Minimum number of samples required to split a node or to be at a leaf node.
- After training, the Decision Tree achieved a cross-validation accuracy of ~90%, which initially suggested that the model was highly effective in capturing relationships within the training dataset.
- However, when tested on unseen data, the accuracy dropped significantly to about 72%. This large gap between training (CV) performance and test performance indicates a clear case of overfitting – the model memorized patterns from the training data but struggled to generalize to new situations.
- While Decision Trees are easy to interpret and can model nonlinear decision boundaries, they are prone to overfitting, especially when not properly regularized or when the dataset is relatively small. In practice, techniques like pruning, ensemble methods (Random Forests, Gradient Boosted Trees), or limiting the maximum depth can help improve generalization.
- In summary, although the Decision Tree classifier appeared very strong during training, its poor generalization on the test set makes it less reliable than Logistic Regression or SVM for predicting Falcon 9 landing success.

K-NEAREST NEIGHBORS

In this experiment, we implemented the K-Nearest Neighbors (KNN) algorithm, a simple yet powerful classification method that relies on the similarity between data points. The core idea of KNN is that a data point will likely belong to the same class as its nearest neighbors in feature space.

We used GridSearchCV with 10-fold cross-validation to carefully tune the hyperparameters. The main parameters optimized were:n_neighbors – the number of nearest neighbors to consider (values from 1 to 10 were tested).algorithm – method used to compute nearest neighbors (auto, ball tree, kd tree, brute).p – the distance metric where p=1 corresponds to Manhattan distance and p=2 corresponds to Euclidean distance.

While the KNN model performed reasonably well, its accuracy was consistently lower compared to Logistic Regression, SVM, and Decision Trees. This indicates that the KNN model is less suitable for this dataset.

One key limitation observed is that KNN is very sensitive to the choice of k. With small values of k, the model can overfit and become noisy, while with larger values of k, the decision boundaries become too smooth, reducing accuracy.

Another challenge is that KNN performance heavily depends on the size and distribution of the dataset. Since our dataset is relatively small (only ~100 samples with 18 in the test set), KNN was not able to achieve strong generalization compared to other models.

MODELS

- SVM and KNN performed better than others, generally.
- Decision Tree worked best for Validation Data only.

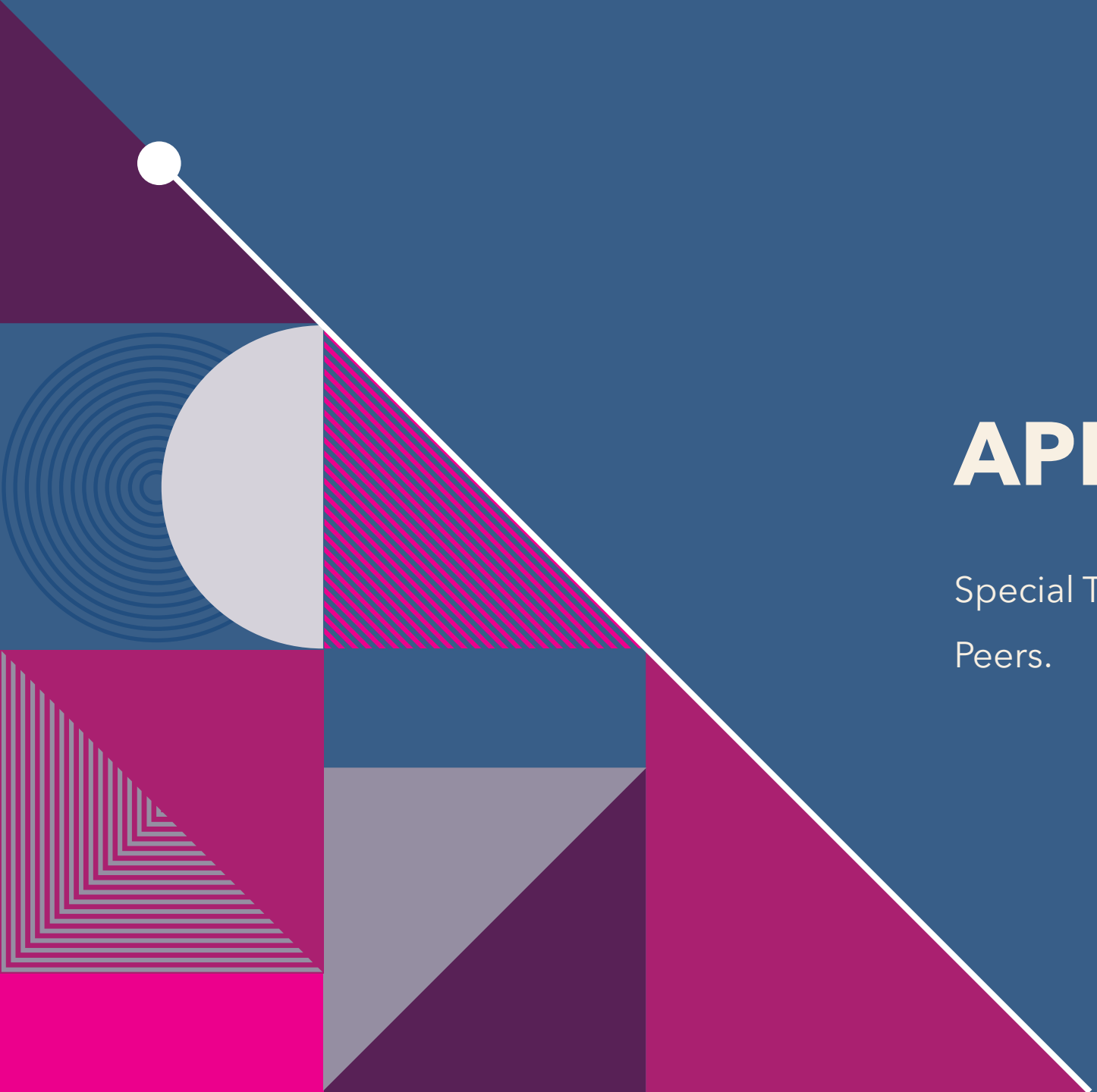
Metric	Test Accuracy	Validation Accuracy
Logistic Regression	0.833	0.8464
SVM	**0.833**	**0.848**
DecisionTree	0.722	0.9017
KNN	**0.833**	**0.848**

MODEL COMPARISON

- Logistic Regression was the first classification model tested. It is a simple yet powerful method that provides interpretable coefficients and clear decision boundaries. After hyperparameter tuning, Logistic Regression achieved a cross-validation accuracy of around 84% and a test set accuracy of about 83%. The consistency between training and test results shows that this model generalizes well and avoids overfitting.
- Support Vector Machine (SVM) performed slightly better during cross-validation, achieving close to 85% accuracy, but on the test set it also stabilized at 83%. This demonstrates that SVM is a strong competitor, capable of capturing complex patterns in the data, but its advantage over Logistic Regression was not significant in terms of real-world predictive performance.
- Decision Trees achieved the highest cross-validation accuracy of almost 90%, which at first glance looked promising. However, when evaluated on the unseen test data, the performance dropped to around 72%, indicating that the model had overfit the training data. While Decision Trees can capture nonlinear relationships, they require pruning or ensemble methods (like Random Forests) to generalize well.
- These results clearly highlight the importance of evaluating models not just on training or cross-validation accuracy, but also on test set performance. A model that performs well in training but poorly in testing is not reliable for deployment.
- Considering accuracy, stability, and interpretability, Logistic Regression emerged as the most reliable model. It provided balanced performance, strong generalization, and the advantage of being straightforward to explain to stakeholders, making it the preferred choice for predicting Falcon 9 first-stage landing success.

CONCLUSIONS

- Best Models were KNN and SVM
- Reused Boosters perform better
- Orbits like LEO, ISS have higher success rates than GTO
- Heavier payloads reduce success probability



APPENDIX

Special Thanks to Coursera, Instructors, and Fellow-Peers.