IBM Data Science Capstone Project

Cong Wang
2023-12-04

Outline

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

Executive Summary

Project Stages

1. **Data Gathering:**
2. **Data Cleaning and Transformation:**
3. **Exploratory Data Analysis:**
4. **Interactive Visual Data Exploration:**
5. **Predictive Analysis (Classification):**

Project result

- Explored data through EDA.
- Conducted geospatial analysis and Created an interactive dashboard.
- Developed four ML models with **~83.33%** accuracy.
- Models tended to over-predict successful landings, indicating the need for more data.

Introduction

Background

SpaceX has disrupted the space industry by offering significantly lower launch costs ($62 million vs. $165 million USD), largely due to successful rocket stage recovery.

Challenge

Space Y aims to compete with SpaceX and has tasked us with training a machine learning model to predict the success of the Falcon 9 first stage recovery.

Objective

This project seeks to predict the success of SpaceX Falcon 9 first stage landings, enabling cost estimation and informed competition in the commercial space launch market.

Section 1:

Methodology

Summary of Data Analysis Methodology

Data Collection

- **SpaceX API & Wikipedia**: Combined data from SpaceX public API and SpaceX Wikipedia page.
- **GET requests**: Making GET requests to the SpaceX REST API.
- **Web Scraping**: Extracting additional data.

Data Wrangling

- **Value Counts**: Applied `.value_counts()` to analyze the data.

Exploratory Data Analysis (EDA)

- **SQL Queries**: Manipulated and evaluated the SpaceX dataset.
- **Pandas & Matplotlib**: Visualized relationships and patterns.

Interactive Visual Analytics

- **Folium**: Performed geospatial analytics.
- **Plotly Dash**: Created an interactive dashboard.

Data Modelling and Evaluation

- **Scikit-Learn**: Pre-processing and data splitting.
- **Model Training**: Employed classification models.
- **Hyperparameter Tuning**: Utilized GridSearchCV.
- **Model Assessment**:
  - Confusion matrices for model evaluation.
  - Accuracy assessment for each model.

Data Collection

Space X Data Acquisition

1. API Data Collection

- **Source**: Space X Public API
- **Columns**:
  - `FlightNumber`, `Date`, `BoosterVersion`, `PayloadMass`
  - `Orbit`, `LaunchSite`, `Outcome`, `Flights`, `GridFins`
  - `Reused`, `Legs`, `LandingPad`, `Block`, `ReusedCount`
  - `Serial`, `Longitude`, `Latitude`

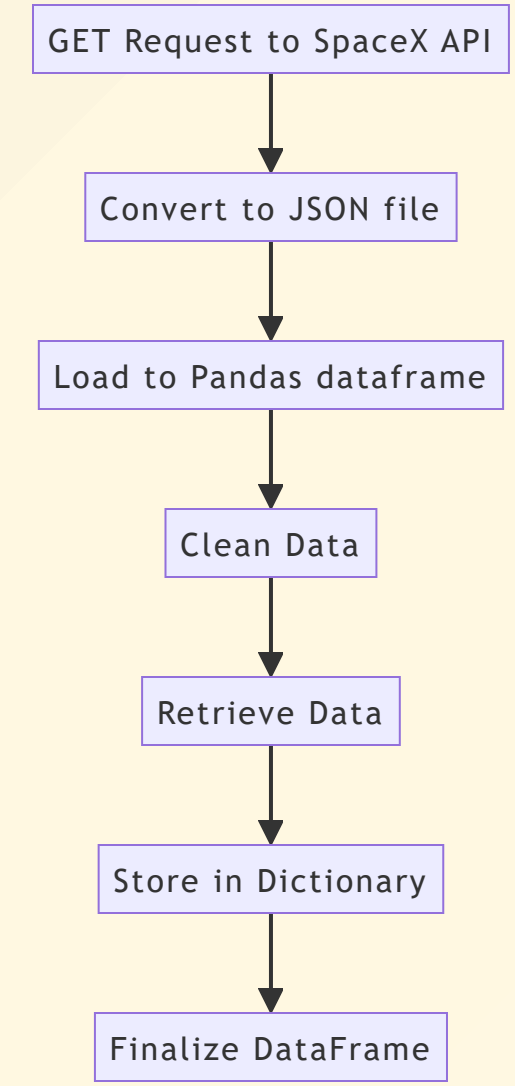
2. Web Scraping

- **Source**: Space X Wikipedia Entry
- **Columns**:
  - `Flight No.`, `Launch site`, `Payload`, `PayloadMass`
  - `Orbit`, `Customer`, `Launch outcome`, `Version Booster`
  - `Booster landing`, `Date`, `Time`

Data Collection - SpaceX Public API

REST API

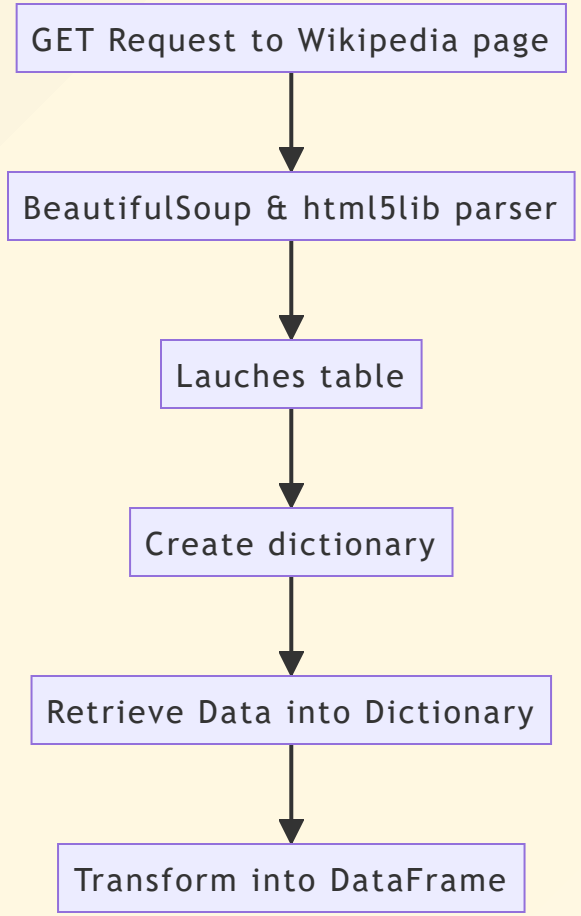
- This project involves integrating with the SpaceX API to gather comprehensive data on their launches. The process begins by making a GET request to the SpaceX REST API to obtain detailed information about various aspects of each launch. This includes data on the rocket, the payload, launch and landing specifics, and the result of the landing. The received data is then converted into a JSON format and subsequently into a Pandas DataFrame for ease of manipulation and analysis.



Data Collection - Scraping

Wikipedia

- In this project, the primary goal is to perform web scraping for collecting historical launch records of Falcon 9 from the Wikipedia page titled "List of Falcon 9 and Falcon Heavy launches." The process begins by requesting the HTML page from its static URL and storing the response in a designated object.



Data Wrangling Process

1. **Data Loading and Inspection:**
    - Load the SpaceX dataset.
    - Inspect the dataset for structure and columns.
2. **Number of Launches on Each Site:**
    - Count the number of launches at each site using `.value_counts()` on 'LaunchSite'.
3. **Number and Occurrence of Each Orbit:**
    - Count the occurrences of each orbit type using `.value_counts()` on 'Orbit'.
4. **Number and Occurrence of Landing Outcome per Orbit Type:**
    - Filter the data for each orbit type and count landing outcomes within each subset.
5. **Create the Training Label 'class':**
    - Create a new 'class' column based on specified conditions:
        - If 'Mission Outcome' is True, set 'class' to 1.
        - For other cases, set 'class' to 0.

Exploratory data analysis (EDA) with Data Visualization

Exploratory Data Analysis Summary

- **Variables Analyzed**: Flight Number, Payload Mass, Launch Site, Orbit, Class, Year.
- **Plots Utilized**:
  - Scatter plots: Flight Number vs. Payload Mass, Payload Mass vs. Launch Site, Payload vs Orbit.
  - Line charts: Orbit vs. Success Rate, Success Yearly Trend.
  - Bar plots: Flight Number vs. Launch Site, Flight Number vs. Orbit.
- **Purpose**: To examine relationships between variables for potential use in training machine learning models.

EDA with SQL Queries

EDA using SQL in IBM DB2

- **Database Integration**: Loaded dataset into IBM DB2; utilized SQL and Python integration for querying.
- **Data Exploration**:
  - Obtained information on launch site names, mission outcomes, payload sizes, booster versions, and landing outcomes.
- **Key SQL Queries**:
  - Identified unique space mission launch sites.
  - Retrieved 5 records of launch sites starting with 'CCA'.
  - Calculated total payload mass by NASA (CRS) boosters.
  - Computed average payload mass for booster version F9 v1.1.
  - Determined date of first successful ground pad landing.
  - Listed boosters with successful drone ship landings and payloads of 4000-6000 kg.
  - Counted total successful vs. failed mission outcomes.
  - Identified boosters with maximum payload mass.
  - Analyzed failed drone ship landings, including booster and launch site details (2015).
  - Ranked landing outcomes from 2010-06-04 to 2017-03-20 in descending order.

Build an Interactive Map with Folium

Interactive Map Creation with Folium

This presentation outlines the process of developing an interactive map using Folium to analyze rocket launch sites. The map highlights launch sites, differentiates between successful and unsuccessful landings, and examines proximity to critical infrastructures like railways, highways, coasts, and cities. This approach provides insights into the strategic placement of launch sites and the success rates of landings based on their locations.

Key steps in the map creation include:

- Marking all launch sites with folium.Circle and folium.Marker on a Folium Map.
- Visualizing successful and failed launches by clustering launch points with similar coordinates and marking them with color-coded markers (green for success, red for failure).
- Calculating and displaying distances from each launch site to nearby essential facilities. This is achieved by using latitude and longitude values to place markers and drawing folium.PolyLine to represent distance lines.

Through these steps, the map serves as an interactive tool for comprehensively understanding launch site selections and their operational outcomes.

Build a Dashboard with Plotly Dash

Dashboard Visualization Summary

- **Pie Chart**:
  - Displays distribution of successful landings across all launch sites.
  - Offers individual launch site success rate visualization.
  - Built with Plotly's `px.pie()`.
  - Features a filter for site-specific success/failure ratio using `dcc.Dropdown()`.
- **Scatter Plot**:
  - Illustrates the relationship between launch outcomes and payload mass.
  - Allows selection between all sites or individual sites.
  - Payload mass adjustable via a slider (0 - 10,000 kg).
  - Created using `px.scatter()`.
  - Includes filters for payload mass ranges (`RangeSlider()`) and booster version.

These interactive elements enhance the Plotly Dash dashboard, enabling dynamic data exploration.

Predictive Analysis (Classification)

1. Model Development

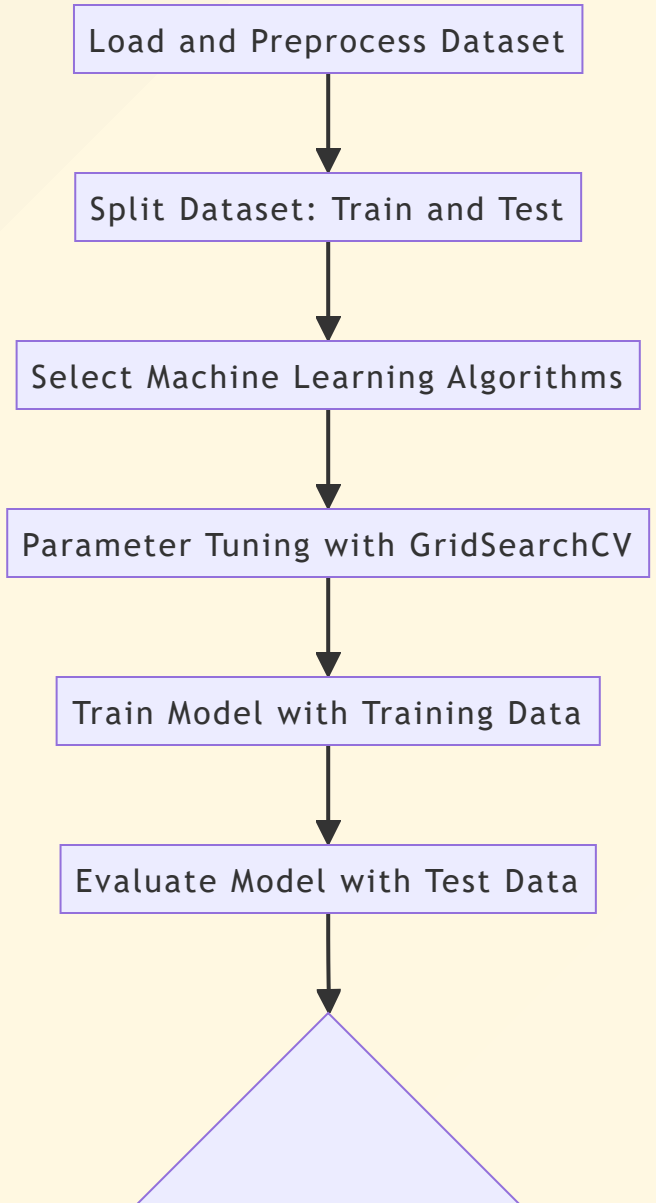
- **Data Preparation**
  - Load and preprocess dataset (standardization included).
  - Divide dataset into training and test sets using `train_test_split()`.
- **Algorithm Selection**
  - Identify suitable machine learning algorithms.
- **Parameter Tuning**
  - For each algorithm:
    - Implement `GridSearchCV` for hyperparameter optimization.
    - Train the model with training data.

2. Model Evaluation

- For each algorithm:
  - Utilize `GridSearchCV` results to:
    - Determine optimal hyperparameters (`best_params_`).
    - Assess model accuracy (`score` and `best_score_`).
  - Analyze performance using Confusion Matrix.

3. Selecting the Best Model

- Compare accuracy scores across all algorithms.



Results

Section 2

Insights drawn from the EDA