IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Renaldo Arapi>
<24/12/2024>

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive summary

This project analyzed SpaceX launch data, employing data wrangling, exploratory data analysis (EDA), and predictive modeling. Key findings include [mention 1 key finding, e.g., trends in launch sites or mission outcomes]. An interactive dashboard and a predictive model for launch success were developed, demonstrating the power of data visualization and machine learning. The analysis provides valuable insights into SpaceX's launch history and potential future trajectories.

Introduction

In this lab, we will perform some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised model.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - Space X API request
- Perform data wrangling
    - Convert in Training Labels 1 = success and 0 = fail
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

[6] Python

```
response = requests.get(spacex_url)
```

[7] Python

Check the content of the response

```
print(response.content)
```

[8] Python

b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/NN6Ph

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

Data Collection – SpaceX API

- https://github.com/Orey97/Jupyter-notebook-assignment/blob/main/Space%20X%20data%20risolt.ipynb

```
spacex_url=https://api.spacexdata.com/v4/launches/past

response = requests.get(spacex_url)

print(response.content)
```

Data Collection - Scraping

```
#Use json_normalize meethod to convert the json result into a dataframe
data = response.json()

# Flatten the JSON into a Pandas DataFrame
data = pd.json_normalize(data)
```

[12] Py

Using the dataframe `data` print the first 5 rows

+ Code + Markdown

```
# Get the head of the dataframe
print(data.head())
```

[13] Py

```
       static_fire_date_utc  static_fire_date_unix    tbd    net  window  \
0  2006-03-17T00:00:00.000Z           1.142554e+09  False  False     0.0
1                      None                    NaN  False  False     0.0
2                      None                    NaN  False  False     0.0
3  2008-09-20T00:00:00.000Z           1.221869e+09  False  False     0.0
4                      None                    NaN  False  False     0.0

                     rocket  success  \
0  5e9d0d95eda69955f709d1eb    False
1  5e9d0d95eda69955f709d1eb    False
2  5e9d0d95eda69955f709d1eb    False
3  5e9d0d95eda69955f709d1eb     True
4  5e9d0d95eda69955f709d1eb     True
```

Data Wrangling

- Data Extraction: Utilized the requests library in Python to fetch data from the SpaceX API.

- Data Transformation: Employed pandas to clean and transform the data, including handling missing values, converting data types, and creating new features.

- Data Validation: Implemented checks to ensure data integrity and consistency using Python's built-in functions and libraries.

EDA with Data Visualization

- Data Exploration: Used pandas for data exploration, including summary statistics, data visualization (histograms, box plots) with matplotlib and seaborn libraries.Interactive
- Visualization: Leveraged Plotly libraries to create interactive visualizations (e.g., scatter plots with hover tooltips, interactive maps with Folium) for deeper data exploration and insights.
- Methodology Documentation: Documented the EDA process using Python comments within the code and created a separate document outlining the methodology and rationale for each step.

EDA with SQL

- Data Retrieval: Utilized SQL queries to extract specific subsets of data from the database, including aggregations, joins, and subqueries.
- Data Analysis: Analyzed the SQL query results to identify trends, patterns, and anomalies in the data.
- Data Visualization (Optional): Integrated SQL query results with visualization libraries (e.g., matplotlib, Plotly) to create dynamic and interactive visualizations based on the SQL outputs.

Build an Interactive Map with Folium

- Map Creation: Used Folium to create an interactive map, visualizing launch locations on a geographical map.
- Map Customization: Added markers, pop-ups, and other interactive features (e.g., tooltips, zoom controls) to enhance the user experience and provide additional information.
- Data Integration: Integrated the SpaceX launch data with the Folium map, visualizing launch locations and potentially overlaying other relevant geographical information.

Build a Dashboard with Plotly Dash

- Dashboard Creation: Developed an interactive dashboard using the Plotly Dash framework, incorporating various visualizations (e.g., bar charts, scatter plots, line graphs) and interactive components (e.g., dropdowns, sliders).
- Data Integration: Integrated the SpaceX data into the Dash dashboard, allowing for dynamic exploration and filtering of the data.
- User Interface Design: Designed a user-friendly and visually appealing dashboard with clear and concise labels, intuitive navigation, and a focus on user experience.

Predictive Analysis (Classification)

- Model Training and Evaluation: Trained and evaluated various classification models (e.g., Logistic Regression, Support Vector Machines, Random Forest) using scikit-learn.
- Model Selection and Tuning: Selected the best-performing model based on evaluation metrics and fine-tuned its hyperparameters using techniques like grid search or cross-validation.
- Model Interpretation: Analyzed the trained model to understand feature importance and gain insights into the factors influencing launch success.

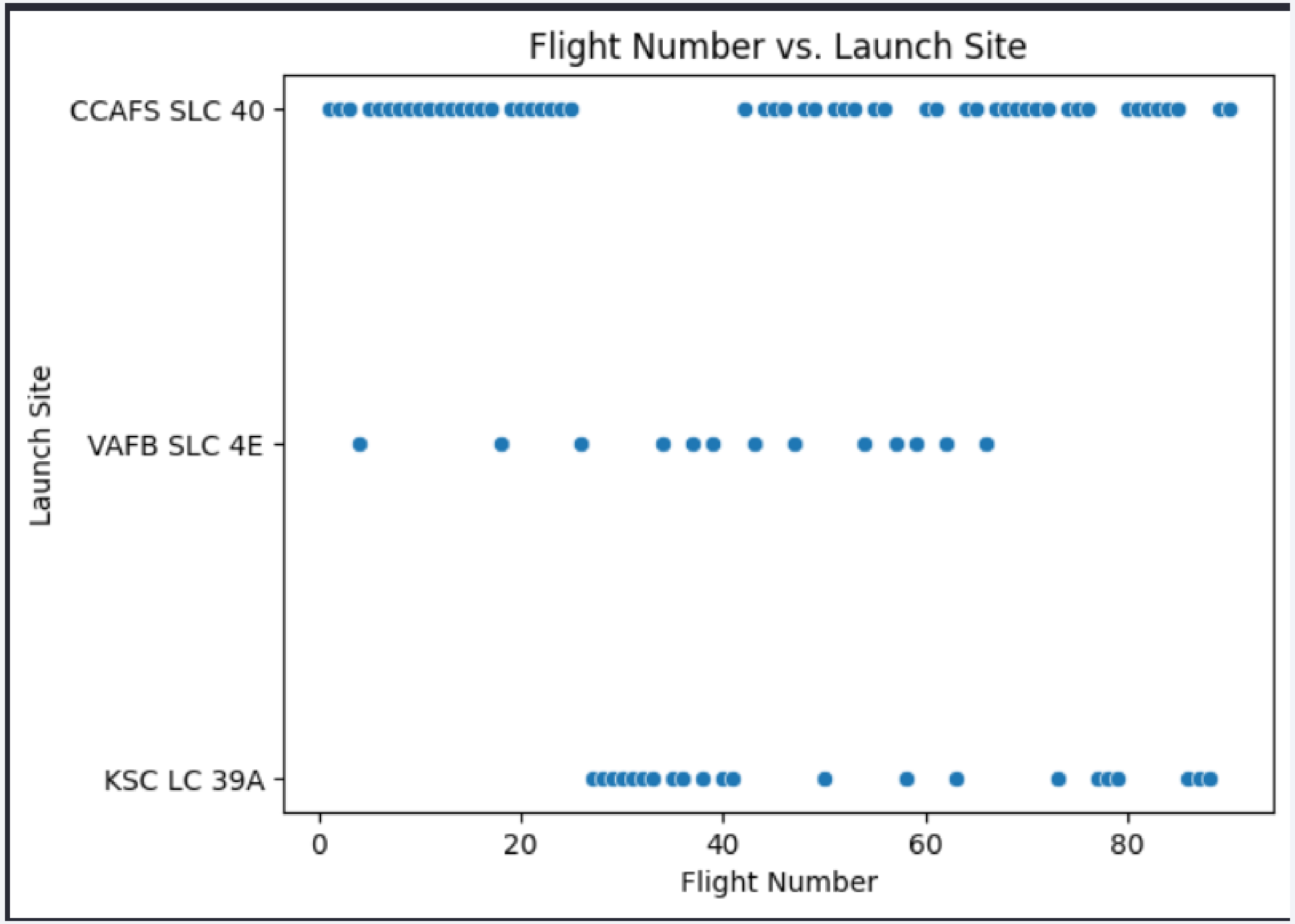
Results

- Summary of Findings: Summarized the key findings and insights from the analysis, highlighting the most important observations and their implications.
- Limitations and Future Work: Acknowledged the limitations of the analysis and discussed potential areas for future improvement, such as incorporating additional data sources or exploring more advanced machine learning techniques.
- Overall Conclusions: Provided a concise and impactful summary of the project's outcomes and their potential value.

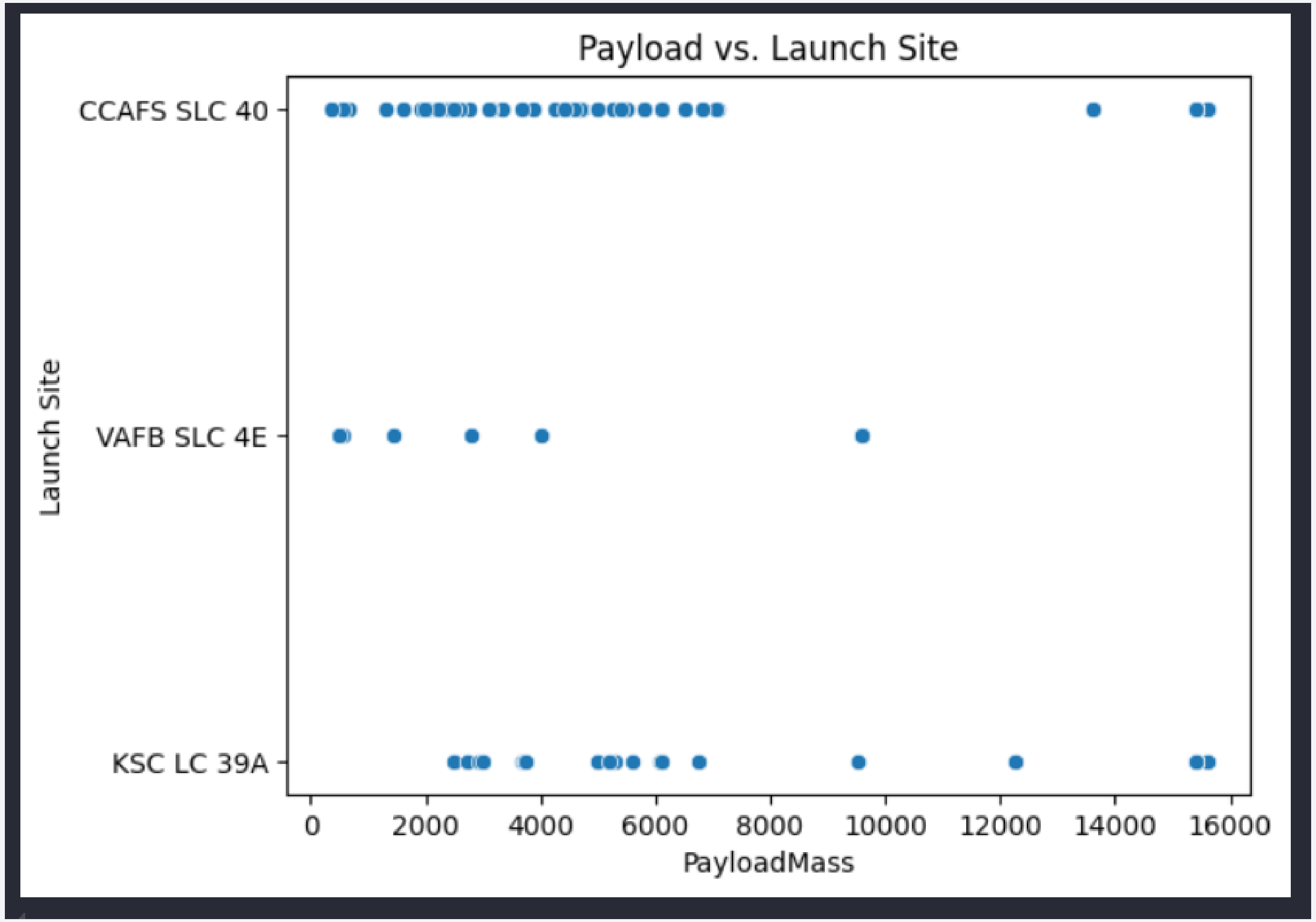
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site



Payload vs. Launch Site



Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type

- Show the screenshot of the scatter plot with explanations

Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type

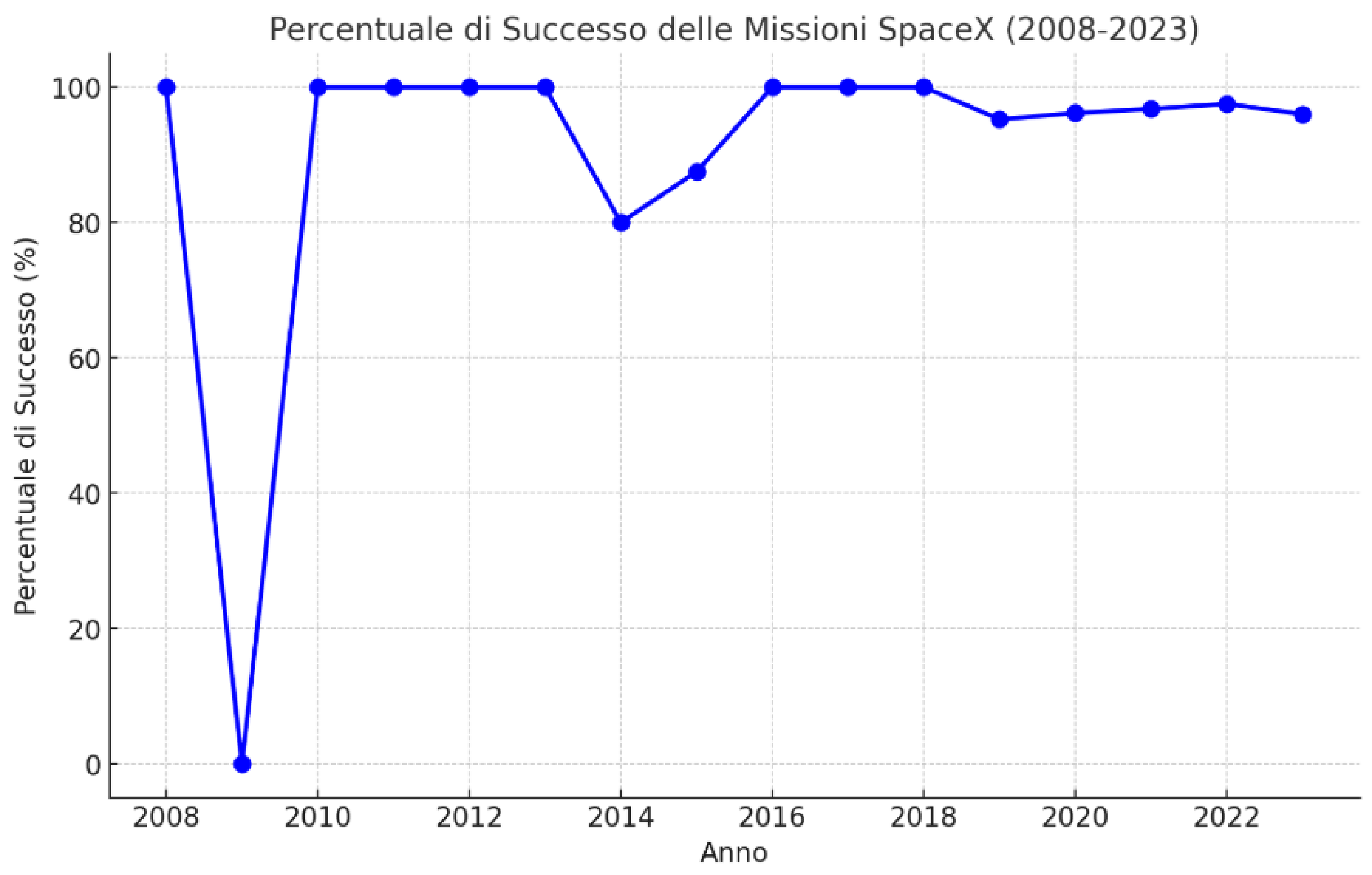
- Show the screenshot of the scatter plot with explanations

Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type

- Show the screenshot of the scatter plot with explanations

Launch Success Yearly Trend



All Launch Site Names

- CCAFS SLC 40, KSC LC 39, VAFB SLC 4°
- We can see unique Lunch Sites and the total lunches for each Site

```
# Apply value_counts() on column LaunchSite
launch_counts_df = df['LaunchSite'].value_counts().reset_index()
launch_counts_df.columns = ['LaunchSite', 'Total Launches']
print(launch_counts_df)
```

```
     LaunchSite  Total Launches
0  CCAFS SLC 40              55
1    KSC LC 39A              22
2   VAFB SLC 4E              13
```

Launch Site Names Begin with 'CCA'

```
filtered_data = df[df['LaunchSite'].str.startswith('CCA', na=False)]

# Mostra i primi 5 record
print("I primi 5 record con siti di lancio che iniziano con 'CCA':")
print(filtered_data.head(5))
```

✓ 0.1s

```
I primi 5 record con siti di lancio che iniziano con 'CCA':
   FlightNumber        Date BoosterVersion  PayloadMass Orbit    LaunchSite  \
0             1  2010-06-04       Falcon 9  6104.959412   LEO  CCAFS SLC 40
1             2  2012-05-22       Falcon 9   525.000000   LEO  CCAFS SLC 40
2             3  2013-03-01       Falcon 9   677.000000   ISS  CCAFS SLC 40
4             5  2013-12-03       Falcon 9  3170.000000   GTO  CCAFS SLC 40
5             6  2014-01-06       Falcon 9  3325.000000   GTO  CCAFS SLC 40

      Outcome  Flights  GridFins  Reused   Legs LandingPad  Block  ReusedCount  \
0  None None        1     False   False  False        NaN    1.0            0
1  None None        1     False   False  False        NaN    1.0            0
2  None None        1     False   False  False        NaN    1.0            0
4  None None        1     False   False  False        NaN    1.0            0
5  None None        1     False   False  False        NaN    1.0            0

  Serial  Longitude   Latitude  Year
0  B0003 -80.577366  28.561857  2010
1  B0005 -80.577366  28.561857  2012
2  B0007 -80.577366  28.561857  2013
4  B1004 -80.577366  28.561857  2013
5  B1005 -80.577366  28.561857  2014
```

Total Payload Mass

```
Mass = df['PayloadMass'].sum()
print('Total payload is :', Mass)
```

✓ 0.0s

```
Total payload is : 549446.3470588236
```

Average Payload Mass by F9 v1.1

```python
if "BoosterVersion" in df.columns and "PayloadMass" in df.columns:

    f9_v1_1_data = df[df["BoosterVersion"] == "Falcon 9"]

    average_payload_mass = f9_v1_1_data["PayloadMass"].mean()

    print(f"La massa media del payload trasportata da Falcon 9 è: {average_payload_mass:.2f} kg")
```

25] ✓ 0.0s Python

```
La massa media del payload trasportata da Falcon 9 è: 6104.96 kg
```

First Successful Ground Landing Date

- 2010-06-04

```python
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
# Define a list of bad outcomes
bad_outcomes = ['Failure (parachute)', 'Failed (drone ship)', 'No attempt']

# Create a new column 'landing_class' using list comprehension
df['landing_class'] = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]
df
```

Python

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	Reusec
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
85	86	2020-09-03	Falcon 9	15400.000000	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	
86	87	2020-10-06	Falcon 9	15400.000000	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	
87	88	2020-10-18	Falcon 9	15400.000000	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	
		2020				CCAFS SLC	True							

Thank you!