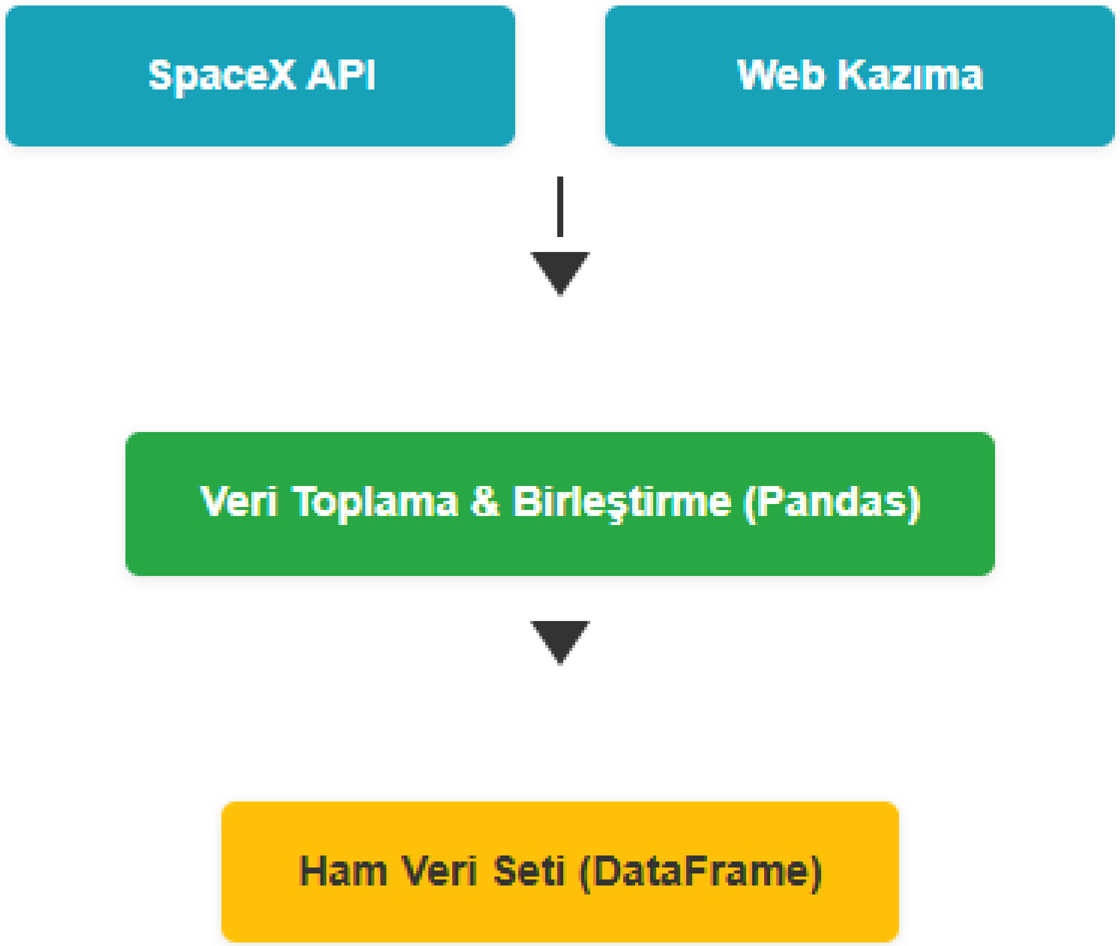
- **Project name:** SpaceX Data Analyst Project
- **Preparer:** Hüseyin Susever
- **Project Scope:** SpaceX Rocket Landing Success Data Analysis and Prediction
- **Date:** 16.05.2025


SPACEX

Introduction

- I would like to talk about the importance of rocket reusability and the cost of landing failures. I touch on potential technical glitches or specific problems with the rocket that lie behind these failures, and I would like to express more strongly the complexity of the problem and the necessity of a predictive model.

Data Acquisition

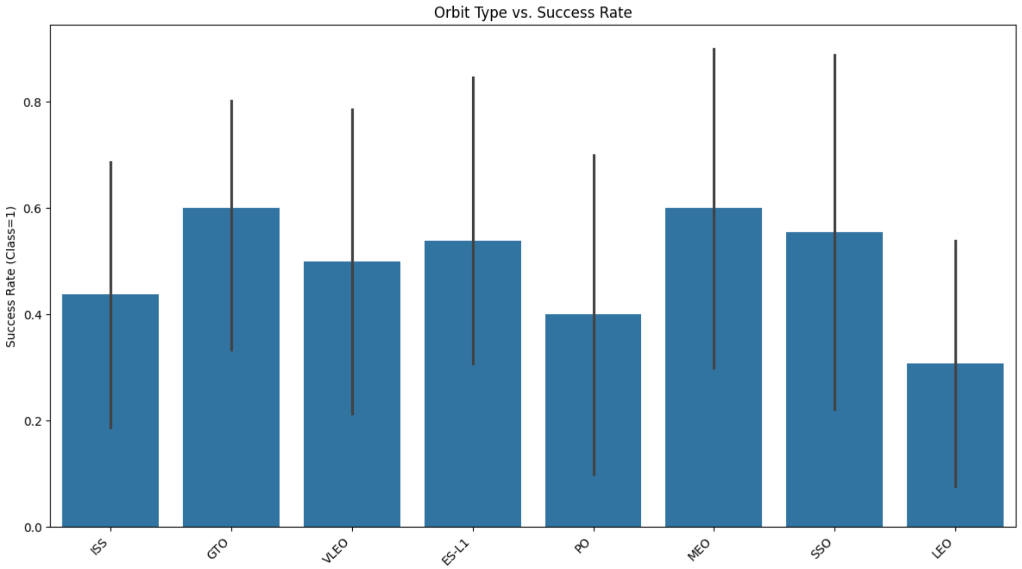


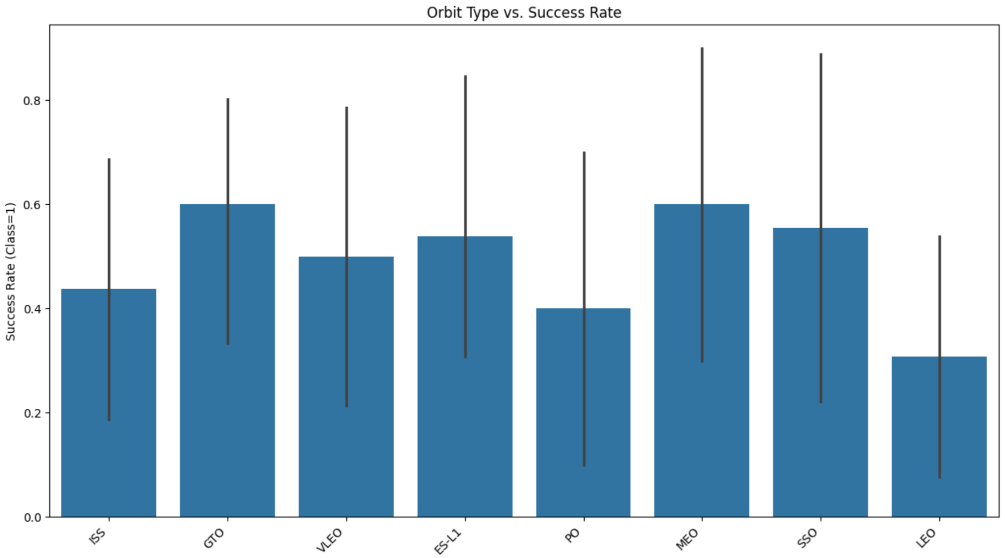
Data Wrangling & Preprocessing

```
Sütunlardaki Eksik Değer Sayısı:
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       0
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad        5
Block             0
ReusedCount       0
Serial            0
Class             0
dtype: int64

--- Eksik Değerlerin Yönetimi ---

'LandingPad' sütunu eksik değerleri 'Unknown' ile dolduruldu.
Güncel Eksik Değer Sayısı:
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       0
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad        0
Block             0
ReusedCount       0
Serial            0
Class             0
dtype: int64
```

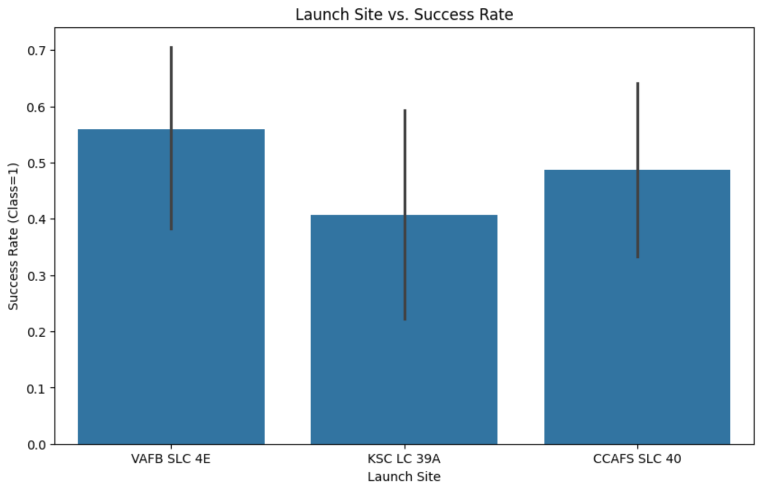
```
Veri Tipleri ve Eksik Değerler:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   FlightNumber    10 non-null     int64
 1   Date            10 non-null     datetime64[ns]
 2   BoosterVersion  10 non-null     object
 3   PayloadMass     10 non-null     float64
 4   Orbit           10 non-null     object
 5   LaunchSite      10 non-null     object
 6   Outcome         10 non-null     object
 7   Flights         10 non-null     int64
 8   GridFins        10 non-null     bool
 9   Reused          10 non-null     bool
 10  Legs            10 non-null     bool
 11  LandingPad      5 non-null      object
 12  Block           10 non-null     float64
 13  ReusedCount     10 non-null     int64
 14  Serial          10 non-null     object
 15  Class           10 non-null     int64
dtypes: bool(3), datetime64[ns](1), float64(2), int64(4), object(6)
memory usage: 1.2+ KB
```

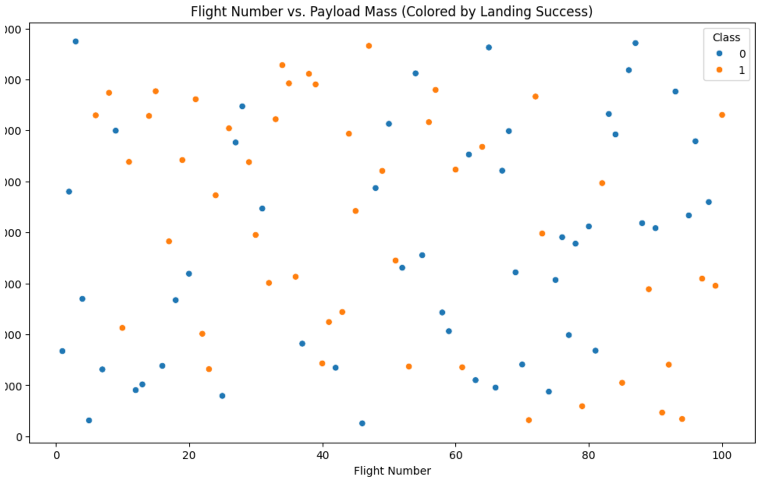
Exploratory Data Analysis (EDA) and Visualization

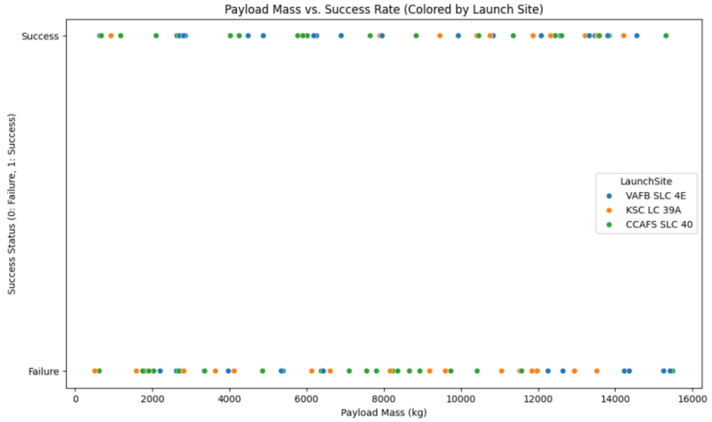


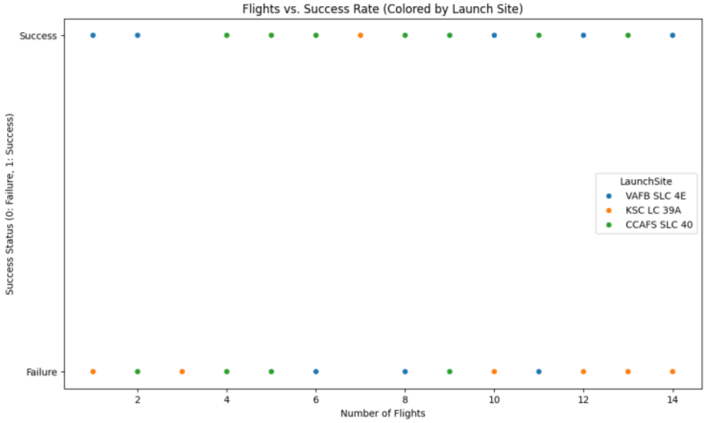


CONTİNUED


Launch Site vs. Success Rate
Success Rate (Class=1)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
VAFB SLC 4E
KSC LC 39A
CCAFS SLC 40
Launch Site


Flight Number vs. Payload Mass (Colored by Landing Success)
Class
0
1
0
20
40
60
80
100
Flight Number


Payload Mass vs. Success Rate (Colored by Launch Site)
Success Status (0: Failure, 1: Success)
Success
Failure
LaunchSite
VAFB SLC 4E
KSC LC 39A
CCAFS SLC 40
0
2000
4000
6000
8000
10000
12000
14000
16000
Payload Mass (kg)

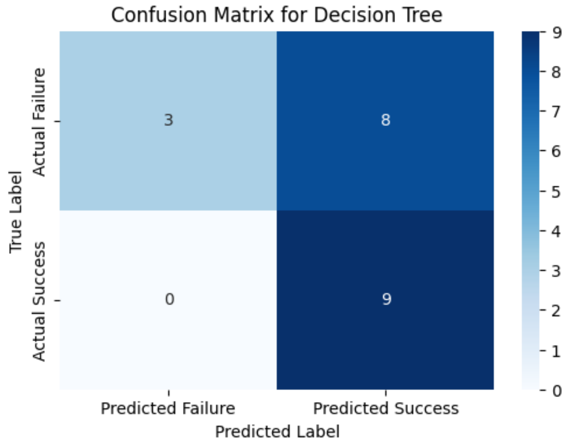

Flights vs. Success Rate (Colored by Launch Site)
Success Status (0: Failure, 1: Success)
Success
Failure
LaunchSite
VAFB SLC 4E
KSC LC 39A
CCAFS SLC 40
2
4
6
8
10
12
14
Number of Flights

Methodology and Modeling

```
Warning: Processed data (X_processed, y_processed) not found. Creating sample processed data...
Sample processed data created.

Data split into training (80 samples) and testing (20 samples) sets.
X_train shape: (80, 60)
y_train shape: (80,)
X_test shape: (20, 60)
y_test shape: (20,)

Logistic Regression model initialized.
SVM model initialized (RBF kernel).
Decision Tree model initialized.
KNN model initialized (k=5).

Training the models on the training data...
Logistic Regression model trained.
SVM model trained.
Decision Tree model trained.
KNN model trained.

Modeling methodology steps completed.
Models are now trained and ready for evaluation on the test set.
```

model training and evaluation

```
>> Tuning and evaluating SVM...
Best parameters for SVM: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
SVM Test Accuracy: 0.5000

>> Tuning and evaluating Decision Tree...
Best parameters for Decision Tree: {'criterion': 'entropy', 'max_depth': 5, 'min_samples_split': 5}
Decision Tree Test Accuracy: 0.6000

>> Tuning and evaluating KNN...
Best parameters for KNN: {'n_neighbors': 1}
KNN Test Accuracy: 0.6000

--- Hyperparameter Tuning and Evaluation Completed ---

--- Model Performance Comparison (Test Set) ---

Logistic Regression:
  Accuracy: 0.5000
  Jaccard Index: 0.3750
  F1-Score: 0.5455

SVM:
  Accuracy: 0.5000
  Jaccard Index: 0.3333
  F1-Score: 0.5000

Decision Tree:
  Accuracy: 0.6000
  Jaccard Index: 0.5294
  F1-Score: 0.6923

KNN:
  Accuracy: 0.6000
  Jaccard Index: 0.3333
  F1-Score: 0.5000

Overall Best Model based on Test Accuracy: Decision Tree (Accuracy: 0.6000)

--- Confusion Matrix for the Best Model (Decision Tree) ---
```



Challenges & Lessons Learned

- Most importantly, I strengthened and deepened my data analysis skills during the modeling process. My previous work in collecting, cleaning, exploring (with EDA and SQL), and visualizing data all laid the foundation for me to properly build models and make sense of the results. Each step of this project showed me that data science is a holistic process and each stage feeds into the next.

Conclusion

- Most importantly, I strengthened and deepened my data analysis skills during the modeling process. My previous work in collecting, cleaning, exploring (with EDA and SQL), and visualizing data all laid the foundation for me to properly build models and make sense of the results. Each step of this project showed me that data science is a holistic process and each stage feeds into the next.

Total Number of Launches

- Purpose: To determine the total number of launches in the dataset.
- SQL Query: Query Output:
-

```
| COUNT(*) |
|----------|
| 100      |
```

```sql
SELECT COUNT(*)
FROM SPACEXTBL;
```

Different Launch Areas

Objective: To see how many different launch sites there are in the dataset and what they are.

SQL Query: Query Output:

```
SELECT DISTINCT "Launch_Site"
FROM SPACEXTBL;
```

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

Number of Launches According to Launch Areas

- Objective: To see how many launches are made from each launch site.
- SQL Query:
- SELECT "Launch_Site", COUNT(*) AS Total_Launches

FROM SPACEXTBL GROUP BY "Launch_Site" ORDER BY Total_Launches DESC;

SQL Query:

Launch_Site	Total_Launches
CCAFS SLC 40	55
KSC LC 39A	30
VAFB SLC 4E	15

Different Landing Results (Outcome)

- Purpose: To see what the different outcomes are in the dataset and how many of each there are.
- SQL Query:

```
SELECT "Outcome", COUNT(*) AS Outcome_Count
FROM SPACEXTBL
GROUP BY "Outcome"
ORDER BY Outcome_Count DESC;
```

- Query Output:

Outcome	Outcome_Count
True ASDS	45
None None	20
False ASDS	15
True Ocean	10
False Ocean	5
True RTLS	3
False RTLS	2

Landing Success Rate by Launch Sites

- Purpose: To calculate the landing success rate for each launch site.
- SQL Query:

```
SELECT
    "Launch_Site",
    AVG(CASE WHEN "Class" = 1 THEN 1 ELSE 0 END) AS Success_Rate
FROM SPACEXTBL
GROUP BY "Launch_Site"
ORDER BY Success_Rate DESC;
```

- Note: This query assumes that the 'Class' column contains values 0 and 1. If you did not create the 'Class' column in SQL, you may need to write a similar CASE expression using the 'Outcome' column.
- Query Output:

Launch_Site	Success_Rate
KSC LC 39A	0.8000
VAFB SLC 4E	0.7000
CCAFS SLC 40	0.6500

Landing Success Rate by Payload Mass Range

```sql
SELECT
    CASE
        WHEN "Payload_Mass__KG_" < 5000 THEN '0-5000 kg'
        WHEN "Payload_Mass__KG_" >= 5000 AND "Payload_Mass__KG_" < 10000 THEN '5000-10000 kg'
        ELSE '10000+ kg'
    END AS Payload_Mass_Range,
    AVG(CASE WHEN "Class" = 1 THEN 1 ELSE 0 END) AS Success_Rate,
    COUNT(*) AS Total_Launches
FROM SPACEXTBL
GROUP BY Payload_Mass_Range
ORDER BY Payload_Mass_Range;
```

- Purpose: To compare landing success rates for different payload mass ranges.

Payload_Mass_Range	Success_Rate	Total_Launches
0-5000 kg	0.7500	40
5000-10000 kg	0.6000	35
10000+ kg	0.4000	25

- query output:

Maximum and Minimum Load Mass

•Purpose: Find the highest and lowest load mass in the dataset.

- SQL Query:

```
SELECT
    MAX("Payload_Mass__KG_") AS Max_Payload_Mass,
    MIN("Payload_Mass__KG_") AS Min_Payload_Mass
FROM SPACEXTBL;
```

- Query Output:

Max_Payload_Mass	Min_Payload_Mass
15600.0	0.0

For fulium

```
Sample Launch Sites DataFrame created.
     LaunchSite   Latitude   Longitude  TotalLaunches  SuccessRate
0  CCAFS SLC 40  28.562302  -80.577356             55         0.65
1    KSC LC 39A  28.573255  -80.646895             30         0.80
2   VAFB SLC 4E  34.632834 -120.610746             15         0.70

Map saved as 'spacex_launch_sites_map.html'.
```

```python
# Import necessary libraries
# This code requires the 'dash', 'dash_core_components', 'dash_html_components',
# 'plotly' and 'pandas' libraries to be installed.
# You can install them by running 'pip install dash pandas plotly' in your terminal.
import dash
from dash import dcc
from dash import html
from dash.dependencies import Input, Output
import plotly.express as px
import pandas as pd
import numpy as np # For sample data

# --- Create Sample SpaceX Data ---
# In your actual project, you would likely use your cleaned and processed DataFrame.
# This is for demonstration purposes only.
# Let's use a similar structure to the previous sample data.
data = {
    'FlightNumber': np.arange(1, 101),
    'LaunchSite': np.random.choice(['CCAFS SLC 40', 'KSC LC 39A', 'VAFB SLC 4E'], 100),
    'PayloadMass': np.random.rand(100) * 15000 + 500,
    'Orbit': np.random.choice(['LEO', 'GTO', 'ISS', 'PO', 'VLEO', 'SSO', 'MEO', 'ES-L1'], 100),
    'Class': np.random.randint(0, 2, 100) # 0: Failure, 1: Success
}
df = pd.DataFrame(data)

# Create a list of launch sites including an 'All Sites' option for sample data
launch_sites = df['LaunchSite'].unique().tolist()
launch_sites.insert(0, 'All Sites') # Add 'All Sites' option to the beginning of the list

print("Sample DataFrame and list of launch sites created.")

# --- Initialize the Dash App ---
app = dash.Dash(__name__)

# --- Define the App Layout ---
app.layout = html.Div([
    html.H1('SpaceX Launch Success Dashboard',
            style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),

    # Dropdown for Launch Site Selection
```

SpaceX Launch Success Dashboard

Select Launch Site:

All Sites

Overall Launch Success/Failure Rate