Executive Summary – Falcon 9 Launch Prediction Project

Project Goal:
To predict whether the first stage of a Falcon 9 rocket will successfully land, enabling cost-saving assessments and competitive benchmarking against SpaceX.

Business Context:
SpaceX significantly reduces launch costs through first-stage reuse. Accurate prediction of landings supports financial forecasting and helps competitors propose more competitive bids.

Key Activities:

Explored and cleaned SpaceX launch data

Engineered features and created training labels

Applied and compared multiple machine learning models (Logistic Regression, SVM, Decision Tree, KNN)

Used hyperparameter tuning and model evaluation techniques

Visualized performance with confusion matrices and accuracy scores

Key Outcome:
Developed a predictive model that accurately classifies landings, offering valuable insights into mission success probability and helping stakeholders make data-driven decisions.

Introduction – Falcon 9 First Stage Landing Prediction

Introduction – Falcon 9 First Stage Landing Prediction

Background:
SpaceX revolutionized the space industry by reusing the first stage of its Falcon 9 rockets, drastically reducing launch costs. Predicting the success of these landings is essential for evaluating mission reliability and financial efficiency.

Objective:
To build machine learning models that can predict whether the first stage of a Falcon 9 rocket will land successfully, based on features from past launch data.

Tools & Techniques:

Python, Pandas, NumPy for data manipulation

Scikit-learn for machine learning modeling and evaluation

Data visualization with Matplotlib and Seaborn

Jupyter Notebooks and GitHub for development and sharing

Business Impact:
A reliable prediction model provides insights for cost estimation, competitive analysis, and strategic planning in the commercial spaceflight industry.

Applied Data Science Capstone

In this capstone project, we aim to predict whether the Falcon 9 first stage will land successfully. SpaceX offers significant savings by reusing the first stage of the Falcon 9 rocket, reducing the launch cost to $62 million, compared to over $165 million for other providers. By predicting first-stage landing success, we can estimate launch costs and offer valuable insights to companies considering competing with SpaceX.

Learning Objectives:

Data Manipulation with Pandas: Develop Python code to clean and manipulate data within a Pandas DataFrame.

Converting JSON to DataFrame: Learn to convert a JSON file into a Pandas DataFrame for analysis.

Sharing Work via GitHub: Create a Jupyter notebook and make it shareable using GitHub for collaboration.

Data Science Methodologies: Apply data science techniques to define a business problem and analyze relevant data.

Data Loading and Analysis: Load datasets, clean the data, and extract valuable insights.

Data Collection API Lab

```python
# Import Libraries
import requests
import pandas as pd
import numpy as np
import datetime

# Pandas display settings
pd.set_option('display.max_columns', None)
pd.set_option('display.max_colwidth', None)

# Helper Functions
def getBoosterVersion(data):
    for x in data['rocket']:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/rockets/" + str(x)).json()
            BoosterVersion.append(response['name'])

def getLaunchSite(data):
    for x in data['launchpad']:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/launchpads/" + str(x)).json()
            Longitude.append(response['longitude'])
            Latitude.append(response['latitude'])
            LaunchSite.append(response['name'])
```

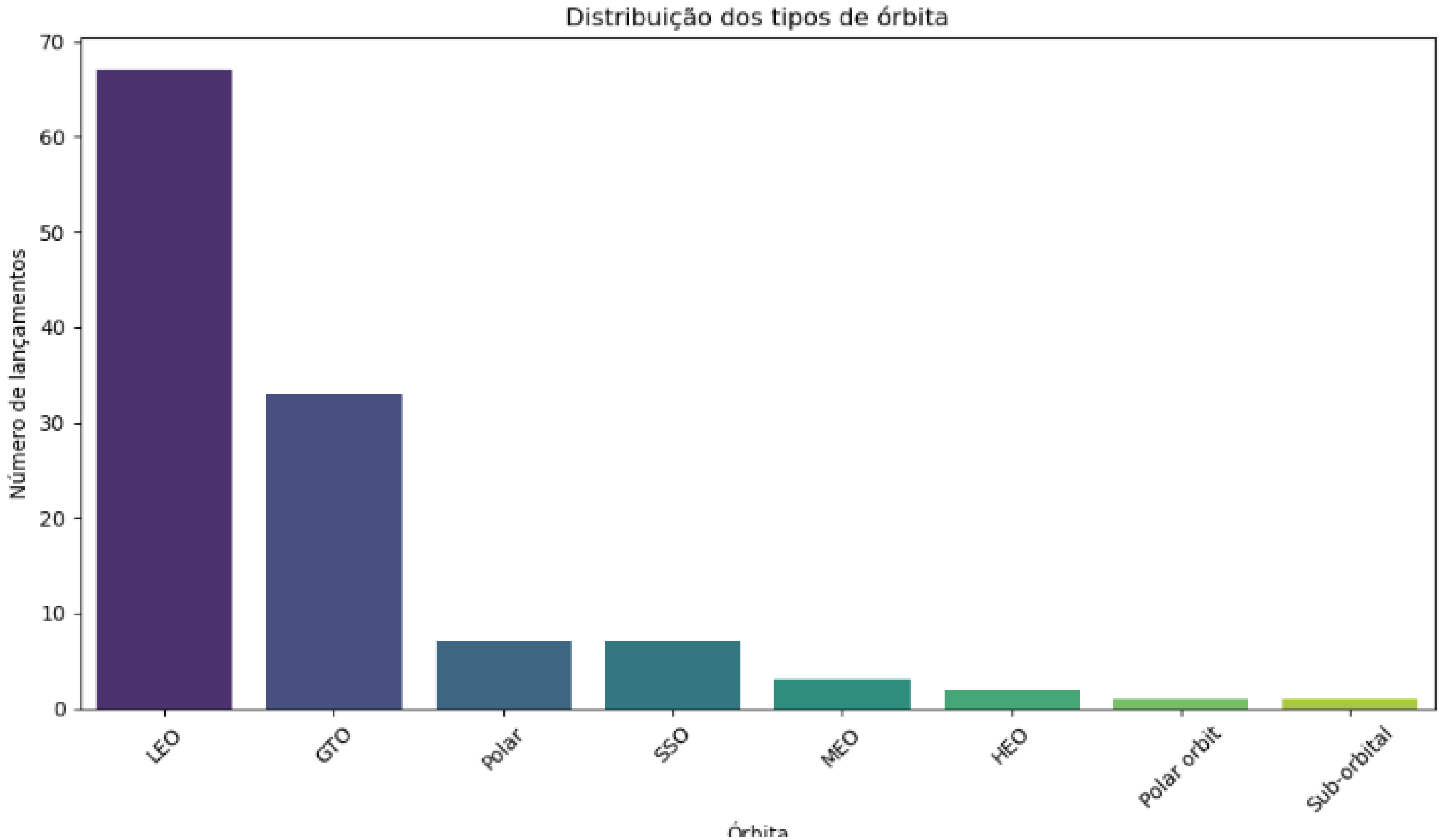
Data Collection API Lab cont.

```
   FlightNumber        Date BoosterVersion  PayloadMass Orbit    LaunchSite  \
4             1  2010-06-04       Falcon 9          NaN   LEO  CCSFS SLC 40
5             2  2012-05-22       Falcon 9        525.0   LEO  CCSFS SLC 40
6             3  2013-03-01       Falcon 9        677.0   ISS  CCSFS SLC 40
7             4  2013-09-29       Falcon 9        500.0    PO   VAFB SLC 4E
8             5  2013-12-03       Falcon 9       3170.0   GTO  CCSFS SLC 40

       Outcome  Flights  GridFins  Reused   Legs LandingPad  Block  \
4    None None        1     False   False  False       None    1.0
5    None None        1     False   False  False       None    1.0
6    None None        1     False   False  False       None    1.0
7  False Ocean        1     False   False  False       None    1.0
8    None None        1     False   False  False       None    1.0

   ReusedCount Serial   Longitude   Latitude
4            0  B0003  -80.577366  28.561857
5            0  B0005  -80.577366  28.561857
6            0  B0007  -80.577366  28.561857
7            0  B1003 -120.610829  34.632093
8            0  B1004  -80.577366  28.561857
```

Data Wrangling

```
# Importar bibliotecas necessárias
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Carregar o dataset (caso não esteja carregado ainda)
df = pd.read_csv('spacex_web_scraped.csv')

# Ver as primeiras linhas do DataFrame
print(df.head())

# Verificar os tipos de órbita únicos
print("Tipos únicos de órbita:")
print(df['Orbit'].unique())

# Contar a frequência de cada tipo de órbita
orbit_counts = df['Orbit'].value_counts()
print("\nContagem de cada tipo de órbita:")
print(orbit_counts)

# Visualizar graficamente a distribuição de órbitas
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Orbit', order=orbit_counts.index, palette='viridis')
```

Data Wrangling cont.



Interactive Dashboard with Ploty Dash

```
import folium
import pandas as pd
from folium.features import DivIcon

# Load the dataset with launch sites, assuming it's available in the current environment
# URL of the dataset
URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv'

# Import the dataset
spacex_df = pd.read_csv(URL)

# Select relevant columns and group by 'Launch Site' to avoid duplicates
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]

# Initializing the map centered on NASA Johnson Space Center (Houston, Texas)
nasa_coordinate = [29.559684888503615, -95.0830971930759]
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)

# Loop through the launch sites DataFrame and add Circle and Marker for each site
for _, row in launch_sites_df.iterrows():
    # Extract the coordinates and site name
    coordinate = [row['Lat'], row['Long']]
    site_name = row['Launch Site']

    # Add a Circle for the launch site
    folium.Circle(
        coordinate,
        radius=1000,  # Radius of the circle
        color='#000000',
        fill=True
    ).add_child(folium.Popup(site_name)).add_to(site_map)
```

Interactive Dashboard with Ploty Dash cont.

Machine Learning

```
def plot_confusion_matrix(y,y_predict):
    "this function plots the confusion matrix"
    from sklearn.metrics import confusion_matrix

    cm = confusion_matrix(y, y_predict)
    ax= plt.subplot()
    sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells
    ax.set_xlabel('Predicted labels')
    ax.set_ylabel('True labels')
    ax.set_title('Confusion Matrix');
    ax.xaxis.set_ticklabels(['did not land', 'land']); ax.yaxis.set_ticklabels(['did not land', 'landed'])
    plt.show()
```

```
import requests
import pandas as pd
from io import StringIO

# URL of the dataset
URL1 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv"

# Fetch the data
response = requests.get(URL1)

# Read the CSV data into a pandas dataframe
data = pd.read_csv(StringIO(response.text))

# Show the first few rows of the dataset
data.head()
```

Machine Learning

```
# Compare all model test accuracies
logreg_acc = logreg_cv.score(X_test, Y_test)
svm_acc = svm_cv.score(X_test, Y_test)
tree_acc = tree_cv.score(X_test, Y_test)
knn_acc = knn_cv.score(X_test, Y_test)

# Print test accuracies
print("Logistic Regression Test Accuracy: ", logreg_acc)
print("SVM Test Accuracy: ", svm_acc)
print("Decision Tree Test Accuracy: ", tree_acc)
print("KNN Test Accuracy: ", knn_acc)

# Determine best model
accuracies = {
    "Logistic Regression": logreg_acc,
    "SVM": svm_acc,
    "Decision Tree": tree_acc,
    "KNN": knn_acc
}

best_model = max(accuracies, key=accuracies.get)
print(f"\nThe best performing model is: {best_model} with accuracy of {accuracies[best_model]:.2f}")
```

```
Logistic Regression Test Accuracy:  0.8333333333333334
SVM Test Accuracy:  0.8333333333333334
Decision Tree Test Accuracy:  0.7777777777777778
KNN Test Accuracy:  0.8333333333333334

The best performing model is: Logistic Regression with accuracy of 0.83
```