**✈️ MODULE 4: Weather & Wind Component Analysis – Full Python Code**

**🔧 Step 1: Import Libraries and Load Data**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

df = pd.read\_csv("air\_arabia\_flight\_operations\_data.csv")

# Check available weather/wind-related columns

print(df[['Route', 'ActualFuel\_kg', 'PlannedFuel\_kg', 'WindComponent\_kts', 'WeatherCondition']].head())

**🌬️ Step 2: Classify Wind Component as Headwind or Tailwind**

# Classify wind

df['WindType'] = df['WindComponent\_kts'].apply(lambda x: 'Tailwind' if x > 0 else ('Headwind' if x < 0 else 'Calm'))

# Display count

print(df['WindType'].value\_counts())

**📊 Step 3: Visualize Fuel Burn by Wind Type**

# Plot boxplot: Actual Fuel vs Wind Type

plt.figure(figsize=(8,6))

sns.boxplot(data=df, x='WindType', y='ActualFuel\_kg', palette='coolwarm')

plt.title('Fuel Burn Distribution by Wind Type')

plt.xlabel('Wind Type')

plt.ylabel('Actual Fuel Burn (kg)')

plt.tight\_layout()

plt.show()

**📈 Step 4: Analyze Correlation Between Wind Speed and Fuel Burn**

# Scatter plot

plt.figure(figsize=(8,6))

sns.scatterplot(data=df, x='WindComponent\_knots', y='ActualFuel\_kg', alpha=0.7)

plt.title('Wind Component vs Actual Fuel Burn')

plt.xlabel('Wind Component (knots)')

plt.ylabel('Actual Fuel Burn (kg)')

plt.tight\_layout()

plt.show()

# Correlation value

correlation = df['WindComponent\_knots'].corr(df['ActualFuel\_kg'])

print("Correlation between Wind Component and Fuel Burn:", correlation)

**🌦️ Step 5: Impact of Weather Conditions on Fuel Consumption**

# Check weather categories

print(df['WeatherCategory'].value\_counts())

# Boxplot by weather

plt.figure(figsize=(10,6))

sns.boxplot(data=df, x='WeatherCategory', y='ActualFuel\_kg', palette='viridis')

plt.title('Fuel Burn by Weather Category')

plt.xlabel('Weather')

plt.ylabel('Fuel Burn (kg)')

plt.tight\_layout()

plt.show()

**📍 Step 6: Route-wise Analysis Under Different Wind Types**

# Average fuel by Route and Wind Type

route\_wind = df.groupby(['Route', 'WindType'])['ActualFuel\_kg'].mean().reset\_index()

# Pivot for plotting

pivot = route\_wind.pivot(index='Route', columns='WindType', values='ActualFuel\_kg').fillna(0)

# Plot top 10 routes with different wind effects

pivot = pivot.sort\_values(by='Headwind', ascending=False).head(10)

pivot.plot(kind='bar', figsize=(12,6), title='Route-wise Avg Fuel Burn by Wind Type')

plt.ylabel('Avg Fuel Burn (kg)')

plt.tight\_layout()

plt.show()

**✅ Summary of Insights:**

* **Tailwind reduces fuel burn**, while **headwind increases it**, especially on long sectors
* Weather categories like **stormy or foggy** lead to **higher fuel usage**
* Certain **routes are more wind-sensitive** than others
* Payload\_kg
* WeatherCategory (categorical)

**🧠 Predicting Fuel Consumption using Linear Regression (Scikit-learn)**

**🔧 Step 1: Import Libraries**

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.metrics import mean\_squared\_error, r2\_score

import numpy as np

**📦 Step 2: Prepare the Features and Target**

# Define features and target

features = ['WindComponent\_knots', 'Route', 'Payload\_kg', 'WeatherCategory']

target = 'ActualFuel\_kg'

X = df[features]

y = df[target]

**🔀 Step 3: Split the Data**

# Split into train and test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42)

**⚙️ Step 4: Define the Preprocessing and Model Pipeline**

# Categorical columns

cat\_features = ['Route', 'WeatherCategory']

num\_features = ['WindComponent\_knots', 'Payload\_kg']

# Preprocessing: OneHotEncode categorical + passthrough numeric

preprocessor = ColumnTransformer([

('cat', OneHotEncoder(handle\_unknown='ignore'), cat\_features),

('num', 'passthrough', num\_features)

])

# Full pipeline

pipeline = Pipeline([

('preprocess', preprocessor),

('model', LinearRegression())

])

**🚀 Step 5: Train the Model**

pipeline.fit(X\_train, y\_train)

**📊 Step 6: Predict and Evaluate**

# Predict on test set

y\_pred = pipeline.predict(X\_test)

# Evaluation metrics

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("Mean Squared Error:", round(mse, 2))

print("R² Score:", round(r2, 3))

**📈 Plot Actual vs Predicted**

plt.figure(figsize=(8,6))

sns.scatterplot(x=y\_test, y=y\_pred)

plt.xlabel("Actual Fuel Burn (kg)")

plt.ylabel("Predicted Fuel Burn (kg)")

plt.title("Actual vs Predicted Fuel Burn")

plt.plot([y.min(), y.max()], [y.min(), y.max()], '--r') # Line of perfect prediction

plt.tight\_layout()

plt.show()

**✅ What this does:**

* Handles both numeric and categorical variables
* Trains a **regression model** to predict fuel usage based on real-world operational data
* Evaluates performance with MSE and R²
* Shows actual vs predicted to visualize accuracy

Bottom of Form