Unidad 3 Entregable 1

Ernesto Martínez Silva

Fundación Universitaria Compensar

Programación para Ciencia de Datos II

Sebastián Rodríguez Muñoz

2024

**Justificación**

Las decisiones tomadas en este proyecto se basaron en asegurar la claridad, utilidad y facilidad de uso del análisis de datos de vuelos. Se eligió Dash por su capacidad para integrar gráficos interactivos y Pandas para la manipulación eficiente de datos, permitiendo la limpieza, transformación y agrupación de los mismos. La estructura del dashboard, con un Dropdown para seleccionar aeropuertos y un RangeSlider para ajustar el tiempo, facilita una navegación intuitiva y personalizada. Se incluyeron gráficos de barras, líneas y tortas por su claridad en la representación de datos clave como pasajeros y proporciones de vuelos nacionales e internacionales. Finalmente, los textos explicativos dentro del dashboard ayudan a guiar al usuario, proporcionando contexto para una interpretación fácil y efectiva de los datos.

**Explicación del Código por Línea**

### 1. ****Importación de bibliotecas****

import dash

from dash import dcc, html

from dash.dependencies import Input, Output

import plotly.express as px

import pandas as pd

Aquí estamos importando las bibliotecas necesarias para construir el dashboard:

* **Dash**: la biblioteca principal para crear aplicaciones web interactivas.
* **dcc (Dash Core Components)** y **html (Dash HTML Components)**: se usan para agregar componentes como gráficos, botones, menús desplegables, etc.
* **dash.dependencies.Input y Output**: son utilizados para conectar los elementos interactivos (entrada y salida) del dashboard.
* **Plotly Express (px)**: una herramienta para crear gráficos simples y rápidos.
* **Pandas (pd)**: biblioteca para la manipulación de datos en DataFrames.

### 2. ****Inicialización de Dash****

app = dash.Dash(\_\_name\_\_)

Inicializamos la aplicación de Dash. Esto es lo que permite que el código se convierta en una aplicación web interactiva.

### 3. ****Cargar los datos****

file\_path = r'D:\Asus\Documents\UCOMPENSAR\SEMESTRE 4\MOM 1\Programacion para base de datos 2\AEROCIVILProf EMS.csv'

df = pd.read\_csv(file\_path, delimiter=';')

Aquí se está cargando el archivo CSV que contiene los datos sobre vuelos en Colombia, utilizando Pandas para leer el archivo. delimiter=';' indica que las columnas del archivo están separadas por punto y coma.

### 4. ****Limpieza de datos****

df.columns = df.columns.str.strip()

Esta línea limpia los nombres de las columnas eliminando espacios en blanco al inicio o final. Es útil para evitar errores de nombres incorrectos.

df['Fecha'] = pd.to\_datetime(df['Fecha'], format='%d/%m/%Y', dayfirst=True, errors='coerce')

Convertimos la columna Fecha al formato de fecha correcto, utilizando el formato día/mes/año.

### 5. ****Filtrado de datos (vuelos desde Colombia)****

df\_colombia = df[df['Pais Origen'] == 'COLOMBIA']

Aquí se filtran los datos para mantener solo los vuelos cuyo país de origen sea Colombia.

### 6. ****Agrupar datos (por pasajeros y carga)****

df\_grouped\_pasajeros = df.groupby('Pais Origen').agg({'Pasajeros': 'mean'}).reset\_index()

df\_grouped\_carga = df\_colombia.groupby('Pais Destino').agg({'Carga + Correo (Kg)': 'mean'}).reset\_index()

Se agrupan los datos:

* El primer grupo calcula el promedio de pasajeros por país de origen.
* El segundo grupo filtra los vuelos de Colombia y calcula el promedio de carga transportada por país de destino.

### 7. ****Agrupar por fecha y calcular la distancia promedio****

df\_grouped\_fecha = df.groupby('Fecha').agg({'Distancia (KM)': 'mean'}).reset\_index()

Agrupamos los datos por fecha y calculamos la distancia promedio de los vuelos para cada fecha.

### 8. ****Estructura del dashboard (Layout)****

app.layout = html.Div([

html.H1("Dashboard - Análisis de Vuelos"),

Se define el layout del dashboard, es decir, la estructura y los componentes que se verán en la aplicación web. Cada componente está dentro de un html.Div() (contenedor).

#### **Selector de Aeropuerto (Dropdown)**

html.Label("Seleccionar Aeropuerto Origen:"),

dcc.Dropdown(

id='dropdown-aeropuerto',

options=[{'label': i, 'value': i} for i in df['Aeropuerto Origen'].unique()],

value=df['Aeropuerto Origen'].unique()[0]

),

Este bloque crea un menú desplegable que muestra todos los aeropuertos de origen únicos en el conjunto de datos. El usuario puede seleccionar uno y el dashboard se actualizará según la selección.

#### **Gráficos y otros elementos interactivos**

dcc.Graph(id='bar-chart'),

Este es un gráfico de barras que se actualizará dinámicamente.

html.Label("Seleccionar Rango de Tiempo (minutos):"),

dcc.RangeSlider(

id='slider-tiempo',

min=1, max=15000,

step=100,

marks={...},

value=[1, 1000]

),

dcc.Graph(id='bar-chart-tiempo'),

Aquí tenemos un **RangeSlider** que permite seleccionar un rango de tiempo en minutos, y el gráfico asociado mostrará la cantidad de vuelos para ese rango de tiempo.

#### **Gráfico de Torta**

dcc.Graph(id='pie-chart'),

Este gráfico de torta muestra la distribución de vuelos nacionales e internacionales.

#### **Mapas de pasajeros y carga**

html.Div([

dcc.Graph(id='mapa-pasajeros'),

dcc.Graph(id='mapa-carga')

], style={'width': '48%', 'display': 'inline-block'})

Estos dos mapas muestran los países de origen de los pasajeros y los destinos de los vuelos que llevan carga desde Colombia.

#### **Gráfico de líneas**

dcc.Graph(id='line-chart'),

Este es un gráfico de líneas que muestra cómo evoluciona la distancia promedio de los vuelos a lo largo del tiempo.

#### **Tabla de datos**

html.Div(id='data-table')

Finalmente, una tabla que muestra los primeros 10 registros de los vuelos filtrados.

### 9. ****Callback para actualizar los gráficos y la tabla****

@app.callback(

[Output('bar-chart', 'figure'),

Output('data-table', 'children'),

...],

[Input('dropdown-aeropuerto', 'value'),

Input('slider-tiempo', 'value')]

)

def update\_dashboard(aeropuerto\_seleccionado, selected\_range):

df\_filtrado = df[df['Aeropuerto Origen'] == aeropuerto\_seleccionado]

Aquí comienza el **callback**, que es la lógica que hace que el dashboard sea interactivo. En este caso:

* Se utiliza la selección del aeropuerto y el rango de tiempo para filtrar los datos.
* Luego, el callback actualiza todos los gráficos y la tabla con base en la selección del usuario.

El **Output** son los gráficos que se actualizarán, mientras que el **Input** son los valores seleccionados por el usuario (el aeropuerto y el rango de tiempo).

#### **Gráfico de barras**

fig\_bar = px.bar(...)

Este gráfico muestra la cantidad total de pasajeros por aeropuerto.

#### **Gráfico de torta**

fig\_pie = px.pie(...)

Este gráfico muestra la proporción de vuelos nacionales e internacionales.

#### **Mapas y gráficos adicionales**

fig\_map\_pasajeros = px.scatter\_geo(...)

fig\_map\_carga = px.scatter\_geo(...)

fig\_line = px.line(...)

Estos gráficos y mapas se actualizan, utilizando las funciones de Plotly Express.

### 10. ****Ejecución del servidor****

if \_\_name\_\_ == '\_\_main\_\_':

app.run\_server(debug=True)

Este bloque final ejecuta la aplicación en modo local, abriendo el dashboard en el navegador web.

### ****11. Layout con texto descriptivo****

app.layout = html.Div([

html.H1("Dashboard - Análisis de Vuelos"),

html.P("Este dashboard presenta un análisis de los vuelos en Colombia..."),

...

])

En este bloque final, se agregan descripciones a las secciones del dashboard para proporcionar un contexto más claro al usuario. Estas explicaciones ayudan a entender mejor cómo usar los filtros y cómo interpretar los gráficos.

**Conclusiones**

El dashboard de análisis de vuelos ofrece una visión clara del tráfico aéreo en Colombia, permitiendo explorar aspectos clave como la distribución de vuelos entre destinos nacionales e internacionales, así como el número de pasajeros y la carga transportada desde distintos aeropuertos. Al ver cómo ha cambiado la distancia promedio de los vuelos a lo largo del tiempo, se pueden detectar patrones que podrían indicar cambios en las preferencias de los pasajeros o en la conectividad, ya sea hacia vuelos de corta o larga distancia.

El análisis por aeropuerto de origen también es útil para identificar qué terminales manejan más tráfico, tanto de pasajeros como de carga, lo cual podría sugerir áreas donde se necesita mejorar la infraestructura o los servicios. Además, la división del tráfico entre vuelos nacionales e internacionales es información valiosa para que las aerolíneas y las autoridades puedan ajustar sus operaciones según la demanda o encontrar formas de mejorar la conectividad internacional.

Por otra parte, la interactividad del dashboard, con sus filtros y visualizaciones personalizables, permite un análisis más detallado y flexible, lo que ayuda a tomar decisiones mejor fundamentadas.

En resumen, este dashboard no solo ayuda a comprender el tráfico aéreo de manera descriptiva, sino que también puede ser una herramienta estratégica para identificar tendencias y mejorar las decisiones dentro de la industria de la aviación de forma mas interactiva.

**Referencias**

* Oehlert, G. (2010). A First Course in Design and Analysis of Experiments. Plotly. Dash Documentation. (August 30, 2023). https://dash.plotly.com/
* The Turing Way. (2021). Zero to Binder. https://the-turing-way.netlify.app/communication/binder/zero-to-binder.html
* Kohavi, R., Longbotham, R., Sommerfield, D., & Henne, R. M. (2008). Controlled Experiments on the Web: Survey and Practical Guide
* Plotly Technologies Inc. (2023). Plotly Python Graphing Library. https://plotly.com/python/
* Dash Developers. (2023). Dash User Guide & Documentation. <https://dash.plotly.com/introduction>
* Pandas Development Team. (2023). pandas Documentation. https://pandas.pydata.org/docs/
* NumPy Developers. (2023). NumPy User Guide. <https://numpy.org/doc/stable/user/index.html>
* Python Software Foundation. (2023). Python 3.10 Documentation. https://docs.python.org/3/
* scikit-learn Developers. (2023). scikit-learn: Machine Learning in Python. https://scikit-learn.org/stable/
* Matplotlib Developers. (2023). Matplotlib: Python plotting. <https://matplotlib.org/stable/users/index.html>
* SciPy Developers. (2023). SciPy library documentation. https://docs.scipy.org/doc/scipy/
* McKinney, W. (2023). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media.
* VanderPlas, J. (2023). Python Data Science Handbook: Essential Tools for Working with Data. https://jakevdp.github.io/PythonDataScienceHandbook/
* Quarto. (2023). Getting Started with Jupyter. Quarto Documentation. <https://quarto.org/docs/get-started/computations/jupyter.html>
* Dash Bootstrap Components. (2023). Bootstrap components for Plotly Dash. https://dash-bootstrap-components.opensource.faculty.ai/