```
#  original web

#%%writefile TiemposUrg2.py

# cargar librerias
import streamlit as
st
import types  # Importa types en lugar de builtins
import pandas as pd
import numpy as
np
import matplotlib.pyplot as plt
import seaborn as sns
from typing import List,
Tuple
from prophet import Prophet

def set_page_config():
    st.set_page_config(

page_title="Tiempos de Urgencias",
        page_icon=":bar_chart:",

  layout="wide",
        initial_sidebar_state="expanded",
    )

st.markdown("<style> footer {visibility: hidden;} </style>",
unsafe_allow_html=True)

set_page_config()

#Define una función de hash personalizada para
tu función
def my_hash_func(func):
    return
id(func)

@st.cache_resource(hash_funcs={types.FunctionType: my_hash_func})

def
load_data(url):
    # Cargamos los datos desde el archivo Excel
    return
pd.read_excel(url)

url =
"https://github.com/Vitotoju/Compensar/raw/main/tiempos_urgencias.xlsx"
dataset =
load_data(url)
df = dataset

# Asegúrate de que la columna 'ds' sea de tipo
datetime
df['FECHA_LLEGADA'] = pd.to_datetime(df['FECHA_LLEGADA'])

# Cadena Más Común
(Moda)  -  para reemplazar los datos vacios con el valor más frecuente o la moda
promedio =
df['Tiempo_Minutos_Total'].median()
df.loc[df['Tiempo_Minutos_Total'] > 420,
'Tiempo_Minutos_Total'] = promedio
df.loc[df['Tiempo_Minutos_Total'] < 0,
'Tiempo_Minutos_Total'] = promedio

# Filtros
st.sidebar.header("Opciones a filtrar:
")

# Filtros Laterales
filtro_centro = st.sidebar.selectbox('Filtrar por Centro',
['Todos'] + df['CENTRO_ATENCION'].unique().tolist())
filtro_mes =
st.sidebar.selectbox('Filtrar por Mes', ['Todos'] +
```

```
df['MES'].unique().tolist())
filtro_clasificacion = st.sidebar.selectbox('Filtrar por
Triague', ['Todos'] +
df['CLASIFICACION_TRIAGE'].unique().tolist())

st.sidebar.header("Prediccion en Dias:
")
opciones = ['Todos', 10, 15, 30, 90]
filtro_Prediccion =
st.sidebar.selectbox('Filtrar por Dias', opciones)

st.sidebar.info('Created by Victor -
Diana')

# Aplicar filtros a los datos
filtro_anos = df['AÑO'].unique().tolist()

if
filtro_centro == 'Todos':
    mask_centro = df['CENTRO_ATENCION'].notna()
else:

mask_centro = df['CENTRO_ATENCION'] == filtro_centro

if filtro_Prediccion == 'Todos':

mask_Prediccion = 90
else:
    mask_Prediccion = filtro_Prediccion

if filtro_mes ==
'Todos':
    mask_mes = df['MES'].notna()
else:
    mask_mes = df['MES'] == filtro_mes

if
filtro_clasificacion == 'Todos':
    mask_clasificacion =
df['CLASIFICACION_TRIAGE'].notna()
else:
    mask_clasificacion = df['CLASIFICACION_TRIAGE']
== filtro_clasificacion

# Crear gráficas de barras

with st.container():

st.subheader("Bienvenidos  :wave:")
  st.title("? Tiempos Atencion de
Urgencias")

  # Combinar las máscaras de filtro
  mask = mask_centro & mask_mes
& mask_clasificacion
  numero_resultados = df[mask].shape[0]

## KPIs


@st.cache_data
  def calculate_kpis(df: pd.DataFrame) -> List[float]:

total_minutos1 =(df[mask]['Tiempo_Minutos_Total'].sum())
        Total_minutos =
f"{total_minutos1:.1f}M"
        total_pacientes =
df[mask]['PACIENTE_#_DOCUMENTO'].nunique()
        Promedio_minutos = f"{total_minutos1 /
numero_resultados:.1f}Min"
        Promedio_minutos2
=(df[mask]['Tiempo_Minutos_Total'].min())
        promedio =
f"{Promedio_minutos2:.1f}Min"
        return [Promedio_minutos, total_pacientes,
```

```
promedio, numero_resultados]


  def display_kpi_metrics(kpis: List[float], kpi_names:
List[str]):
        st.header("KPI Metrics")
        for i, (col, (kpi_name,
kpi_value)) in enumerate(zip(st.columns(4), zip(kpi_names, kpis))):

col.metric(label=kpi_name, value=kpi_value)


  @st.cache_data
  def
calculate_kpisp(forecast: pd.DataFrame) -> List[float]:
        total_minutos2
=(forecast['yhat'].sum())
        Total_minutos = f"{total_minutos2:.1f}M"

total_pacientes = forecast['yhat'].nunique()
        Promedio_minutos = f"{total_minutos2
/ total_pacientes:.1f}Min"
        Promedio_minutos2 =(forecast['yhat'].median())

 promedio = f"{Promedio_minutos2:.1f}Min"
        return [promedio, total_pacientes,
Promedio_minutos, numero_resultados]

  def display_kpi_metricsp(kpis: List[float],
kpi_names: List[str]):
        st.header("KPI Metrics")
        for i, (col,
(kpi_name, kpi_value)) in enumerate(zip(st.columns(4), zip(kpi_names, kpis))):

col.metric(label=kpi_name, value=kpi_value)

with st.container():

st.write("---")

   # st.sidebar.image(load_image("sur.png"),
use_column_width=True)
    st.header("Historico de Tiempo de Espera de Atencion de
Pacientes")

    kpis = calculate_kpis(df)
    kpi_names = ["Promedio
Minutos", "Cantidad Pacientes", "Minutos Minimo", "Cantidad
Atenciones"]
    display_kpi_metrics(kpis, kpi_names)

    left_column ,
right_column = st.columns(2)

    with left_column:
        st.header("DIA DE LA
SEMANA")
        st.write("Esta imagen muestra Por dias de la semana del
Dia")

        # Ahora puedes acceder al día de la semana usando el atributo
'dayofweek'
        df['day_of_week'] = df[mask]['FECHA_LLEGADA'].dt.dayofweek

       #
promedio = df[mask]['Tiempo_Minutos_Total'].median()
       #
df[mask].loc[df[mask]['Tiempo_Minutos_Total'] > 420, 'Tiempo_Minutos_Total'] = promedio

   # df[mask].loc[df[mask]['Tiempo_Minutos_Total'] < 0, 'Tiempo_Minutos_Total'] =
promedio

        # Calcula el promedio de las predicciones para cada día de la semana

   average_predicted_minutes =
```

```
df[mask].groupby('day_of_week')['Tiempo_Minutos_Total'].mean()

        # Establece los
índices explícitamente
        average_predicted_minutes.index = ['Lunes', 'Martes',
'Miércoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo']

        # Trazar el gráfico de
barras
        fig, ax = plt.subplots(figsize=(10, 6))

sns.barplot(x=average_predicted_minutes.index, y=average_predicted_minutes.values,
color='blue', ax=ax)
        ax.set_xlabel('Día de la Semana')

ax.set_ylabel('Promedio del Tiempo (minutos)')
        ax.set_title('Promedio del Tiempo por
Día de la Semana')

        # Añade etiquetas a las barras
        for i, bar in
enumerate(ax.patches):
            yval = bar.get_height()
            xval = bar.get_x() +
bar.get_width() / 2
            ax.text(xval, yval, f"{round(yval, 2)}",
ha='center', va='bottom')

        # Muestra la figura en Streamlit

st.pyplot(fig)

    with right_column:
        st.header("HORAS DEL DIA")

st.write("Esta imagen muestra Por Horas del Dia")

        # Ahora puedes
acceder al día de la semana usando el atributo 'dayofweek'
        df['Hora_del_dia'] =
df[mask]['FECHA_LLEGADA'].dt.hour

        promedio =
df[mask]['Tiempo_Minutos_Total'].median()

df[mask].loc[df[mask]['Tiempo_Minutos_Total'] > 420, 'Tiempo_Minutos_Total'] = promedio

    df[mask].loc[df[mask]['Tiempo_Minutos_Total'] < 0, 'Tiempo_Minutos_Total'] =
promedio

        # Calcula el promedio de las predicciones para cada día de la semana

   average_predicted_minutes = df.groupby('Hora_del_dia')['Tiempo_Minutos_Total'].mean()


    # Trazar el gráfico de barras
        fig, ax = plt.subplots(figsize=(10, 6))

sns.barplot(x=average_predicted_minutes.index, y=average_predicted_minutes.values,
color='#00FF00', ax=ax)
        ax.set_xlabel('Hora del Dia')
        ax.set_ylabel('Promedio
del Tiempo (minutos)')
        ax.set_title('Promedio del Tiempo en Horas')

        #
Añade etiquetas a las barras
        for i, bar in enumerate(ax.patches):
            yval =
bar.get_height()
            xval = bar.get_x() + bar.get_width() / 2

ax.text(xval, yval, f"{round(yval, 2)}", ha='center', va='bottom')

        #
```

```
Muestra la figura en Streamlit
        st.pyplot(fig)

with st.container():

st.write("---")

    left_column , right_column = st.columns(2)

    with
left_column:
        st.header("TURNO DEL DIA")
        st.write("Esta imagen
muestra Total Tiempo x Turno del Dia")

        # Condiciones para asignar turnos

      conditions = [
            (df['FECHA_LLEGADA'].dt.time >=
pd.to_datetime('07:00:00').time()) & (df['FECHA_LLEGADA'].dt.time <
pd.to_datetime('14:00:00').time()),  # Mañana
            (df['FECHA_LLEGADA'].dt.time >=
pd.to_datetime('14:00:00').time()) & (df['FECHA_LLEGADA'].dt.time <
pd.to_datetime('19:00:00').time()),  # Tarde
            (df['FECHA_LLEGADA'].dt.time >=
pd.to_datetime('19:00:00').time()) & (df['FECHA_LLEGADA'].dt.time <
pd.to_datetime('23:59:59').time()),  # Noche
            (df['FECHA_LLEGADA'].dt.time >=
pd.to_datetime('00:00:00').time()) & (df['FECHA_LLEGADA'].dt.time <
pd.to_datetime('07:00:00').time())     # Madrugada
        ]

        # Valores
correspondientes a cada turno
        values = ['Mañana', 'Tarde', 'Noche', 'Madrugada']


      # Asignar el turno según las condiciones
        df['Turno'] = np.select(conditions,
values, default='Error')

        # Calcular el promedio de las predicciones para cada
turno
        average_predicted_minutes =
df.groupby('Turno')['Tiempo_Minutos_Total'].mean()

        # Trazar el gráfico de barras

      fig, ax = plt.subplots(figsize=(10, 6))

sns.barplot(x=average_predicted_minutes.index, y=average_predicted_minutes.values,
color='cornflowerblue', ax=ax)
        ax.set_xlabel('Turno del Día')

ax.set_ylabel('Promedio del Tiempo (minutos)')
        ax.set_title('Promedio del Tiempo en
Turnos')

        # Añade etiquetas a las barras
        for i, bar in
enumerate(ax.patches):
            yval = bar.get_height()
            xval = bar.get_x() +
bar.get_width() / 2
            ax.text(xval, yval, f"{round(yval, 2)}",
ha='center', va='bottom')

        # Muestra la figura en Streamlit

st.pyplot(fig)

    with right_column:
        st.header("MES")

st.write("Esta imagen muestra Por Timepo x Mes")
```

```
        # Ahora puedes
acceder al día de la semana usando el atributo 'dayofweek'
        df['Mes1'] =
df[mask]['FECHA_LLEGADA'].dt.month

        promedio =
df[mask]['Tiempo_Minutos_Total'].median()

df[mask].loc[df[mask]['Tiempo_Minutos_Total'] > 420, 'Tiempo_Minutos_Total'] = promedio

    df[mask].loc[df[mask]['Tiempo_Minutos_Total'] < 0, 'Tiempo_Minutos_Total'] =
promedio

        # Calcula el promedio de las predicciones para cada día de la semana

   average_predicted_minutes = df.groupby('Mes1')['Tiempo_Minutos_Total'].mean()

        #
Trazar el gráfico de barras
        fig, ax = plt.subplots(figsize=(10, 6))

sns.barplot(x=average_predicted_minutes.index, y=average_predicted_minutes.values,
color='skyblue', ax=ax)
        ax.set_xlabel('Mes')
        ax.set_ylabel('Promedio del
Tiempo (minutos)')
        ax.set_title('Promedio del Tiempo en Horas')

        # Añade
etiquetas a las barras
        for i, bar in enumerate(ax.patches):
            yval =
bar.get_height()
            xval = bar.get_x() + bar.get_width() / 2

ax.text(xval, yval, f"{round(yval, 2)}", ha='center', va='bottom')

        #
Muestra la figura en Streamlit
        st.pyplot(fig)


with st.container():

st.write("---")
    st.header("Predicción de Tiempo de Espera de Atencion de
Pacientes")

    dfp = dataset[['FECHA_LLEGADA', 'Tiempo_Minutos_Total']].copy()

dfp.rename(columns={'FECHA_LLEGADA': 'ds', 'Tiempo_Minutos_Total': 'y'}, inplace=True)

dfp["y"] = pd.to_numeric(dfp["y"],errors='coerce')

    median =
dfp['y'].median()
    dfp.loc[dfp['y'] > 420, 'y'] = median
    dfp.loc[dfp['y'] < 0,
'y'] = median

    m = Prophet()
    m.fit(dfp[mask])
    future =
m.make_future_dataframe(periods=mask_Prediccion)
    forecast = m.predict(future)

    kpis
= calculate_kpisp(forecast)
    kpi_names = ["Promedio Minutos", "Cantidad
Pacientes", "Minutos Minimo", "Cantidad Atenciones"]

display_kpi_metricsp(kpis, kpi_names)

    left_column , right_column = st.columns(2)
```

```
with left_column:
        st.header("DIA DE LA SEMANA")
        st.write("Esta
imagen muestra Prediccion Por dias de la semana")

        # Añade el día de la
semana a las predicciones
        forecast['day_of_week'] = forecast['ds'].dt.dayofweek


    # Calcula el promedio de las predicciones para cada día de la semana

average_predicted_minutes = forecast.groupby('day_of_week')['yhat'].mean()


total_minutos2 = (forecast['yhat'].sum())

        # Establece los índices explícitamente

       average_predicted_minutes.index = ['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes',
'Sábado', 'Domingo']

        # Trazar el gráfico de barras
        fig, ax =
plt.subplots(figsize=(10, 6))
        sns.barplot(x=average_predicted_minutes.index,
y=average_predicted_minutes.values, color='lightseagreen', ax=ax)
        ax.set_xlabel('Día
de la Semana')
        ax.set_ylabel('Prediccion Promedio del Tiempo (minutos)')

ax.set_title('Prediccion Promedio del Tiempo por Día de la Semana')

        # Añade
etiquetas a las barras
        for i, bar in enumerate(ax.patches):
            yval =
bar.get_height()
            xval = bar.get_x() + bar.get_width() / 2

ax.text(xval, yval, f"{round(yval, 2)}", ha='center', va='bottom')

        #
Muestra la figura en Streamlit
        st.pyplot(fig)

    with right_column:


st.header("HORA DEL DIA")
        st.write("Esta imagen muestra Prediccion Por
dias de la semana")

        # Añade el día de la semana a las predicciones

  forecast['Hora_dia'] = forecast['ds'].dt.hour

        # Calcula el promedio de las
predicciones para cada día de la semana
        average_predicted_minutes =
forecast.groupby('Hora_dia')['yhat'].mean()

        # Trazar el gráfico de barras

fig, ax = plt.subplots(figsize=(10, 6))
        sns.barplot(x=average_predicted_minutes.index,
y=average_predicted_minutes.values, color='skyblue', ax=ax)
        ax.set_xlabel('Hora del
Dia')
        ax.set_ylabel('Prediccion Promedio del Tiempo (minutos)')

ax.set_title('Prediccion Promedio del Tiempo por Día de la Semana')
```

```
        # Añade
etiquetas a las barras
        for i, bar in enumerate(ax.patches):
            yval =
bar.get_height()
            xval = bar.get_x() + bar.get_width() / 2

ax.text(xval, yval, f"{round(yval, 2)}", ha='center', va='bottom')

        #
Muestra la figura en Streamlit
        st.pyplot(fig)


with st.container():

st.write("---")

    left_column , right_column = st.columns(2)

    with
left_column:
        st.header("TURNO DEL DIA")
        st.write("Esta imagen
muestra Prediccion Por Total Tiempo por Turno")

        # Condiciones para asignar
turnos
        conditions = [
            (forecast['ds'].dt.time >=
pd.to_datetime('07:00:00').time()) & (forecast['ds'].dt.time <
pd.to_datetime('14:00:00').time()),  # Mañana
            (forecast['ds'].dt.time >=
pd.to_datetime('14:00:00').time()) & (forecast['ds'].dt.time <
pd.to_datetime('19:00:00').time()),  # Tarde
            (forecast['ds'].dt.time >=
pd.to_datetime('19:00:00').time()) & (forecast['ds'].dt.time <
pd.to_datetime('23:59:59').time()),  # Noche
            (forecast['ds'].dt.time >=
pd.to_datetime('00:00:00').time()) & (forecast['ds'].dt.time <
pd.to_datetime('07:00:00').time())     # Madrugada
        ]

        # Valores
correspondientes a cada turno
        values = ['Mañana', 'Tarde', 'Noche', 'Madrugada']


      # Asignar el turno según las condiciones
        forecast['Turno'] =
np.select(conditions, values, default='Error')

        # Calcula el promedio de las
predicciones para cada día de la semana
        average_predicted_minutes =
forecast.groupby('Turno')['yhat'].mean()

        # Trazar el gráfico de barras

fig, ax = plt.subplots(figsize=(10, 6))
        sns.barplot(x=average_predicted_minutes.index,
y=average_predicted_minutes.values, color='lightseagreen', ax=ax)
        ax.set_xlabel('Turno
del Dia')
        ax.set_ylabel('Prediccion Promedio del Tiempo (minutos)')

ax.set_title('Prediccion Promedio del Tiempo por Turno Diario')

        # Añade etiquetas a
las barras
        for i, bar in enumerate(ax.patches):
            yval = bar.get_height()

           xval = bar.get_x() + bar.get_width() / 2
            ax.text(xval, yval,
```

```
f"{round(yval, 2)}", ha='center', va='bottom')

        # Muestra la figura en
Streamlit
        st.pyplot(fig)

    with right_column:


st.header("MES")
        st.write("Esta imagen muestra Prediccion TIEMPO por
mes")

        # Añade el día de la semana a las predicciones

forecast['Mes2'] = forecast['ds'].dt.month

        # Calcula el promedio de las predicciones
para cada día de la semana
        average_predicted_minutes =
forecast.groupby('Mes2')['yhat'].mean()

        # Trazar el gráfico de barras
        fig,
ax = plt.subplots(figsize=(10, 6))
        sns.barplot(x=average_predicted_minutes.index,
y=average_predicted_minutes.values, color='skyblue', ax=ax)
        ax.set_xlabel('Tiempo x
Mes')
        ax.set_ylabel('Prediccion Promedio del Tiempo (minutos)')

ax.set_title('Prediccion Promedio del Tiempo por Mes')

        # Añade etiquetas a las
barras
        for i, bar in enumerate(ax.patches):
            yval = bar.get_height()

       xval = bar.get_x() + bar.get_width() / 2
            ax.text(xval, yval,
f"{round(yval, 2)}", ha='center', va='bottom')

        # Muestra la figura en
Streamlit
        st.pyplot(fig)


# %%
```