Taller Streamlit

Análisis de ventas de Super Store

Hoy vamos a conocer una nueva herramienta para poder visualizar nuestros datos y crear nuestras primeras páginas web de forma sencilla.

En la galería de Streamlit podemos ver algunos ejemplos de lo que podemos llegar a hacer:

https://streamlit.io/gallery.

Lo primero que tenemos que hacer, como con cualquier librería, es instalarla.

pip install streamlit

Streamlit es una herramienta relativamente nueva y sus funcionalidades aún están creciendo por lo que estamos en un momento en el que todavía podemos abarcar la mayoría de sus características https://docs.streamlit.io/en/stable/api.html.

Análisis de ventas de Super Store

El objetivo del presente taller es utilizar Streamlit para analizar las ventas del grupo Super Store en los años 2013 a 2016. Para ello contamos con los datos de los pedidos recibidos por la empresa durante esos años, en formato Excel.

Ejecutar nuestro código

Para ejecutar nuestro programa debemos ejecutar en la terminal el comando:

streamlit run main.py

Configuración de la aplicación

Vamos a empezar configurando ciertos parámetros de nuestra aplicación, podéis ver todos los parámetros que podemos configurar en el siguiente enlace: https://docs.streamlit.io/en/stable/api.html#placeholders-help-and-options.

Nosotros configuraremos el tamaño de la pantalla, el título y el icono de la página.

Para ello utilizamos el método st.set_page_config().

```
st.set_page_config(page_title = "SUPERSTORE", page_icon=":chart:",
layout="wide")
```

Montamos la introducción

- Vamos a poner una cabecera (st.title()) con el texto “Grupo SuperStore”.

- Vamos a poner un subtítulo (st.subheader()) con el texto “Introducción”

- Vamos a cargar una imagen (st.image())

- Vamos a escribir una introducción (st.markdown()) con el texto:

  "SuperStore es el **grupo lider en el sector de tecnología, suministros y equipamiento de oficina** en Estados Unidos. El grupo nació hace más de 30 años en Detroit (EEUU). Fue la primera empresa en desarrollar una plataforma B2B de compra online de materiales para el entorno de trabajo en 1999. Su catálogo incluye productos tecnológicos, suministros y equipamiento de oficina. A fecha de hoy disponen de **más de 60.000 empresas clientes** en EEUU."

- Vamos a escribir unas líneas adicionales sobre sostenibilidad. Pero no me apetece que se vea siempre, sólo si el usuario quiere verlo (st.expander()), con el texto:

  "Sostenibilidad

  SuperStore garantiza que todos los pasos que da para satisfacer a sus clientes se realizan del modo más sostenible posible.

  - Más de la mitad de los productos son ecológicos.
  - El grupo ha reducido en una tercera parte sus emisiones de CO2 (desde 2010).
  - Asimismo, ha reducido al máximo los embalajes y optimizado sus rutas de transporte."

Cargamos y leemos los datos

El usuario debe cargar el archivo con los pedidos a analizar utilizando el método st.file_uploader().

Los datos se leen exactamente igual que en Pandas, con pd.read_excel()

Además, vamos a insertar un botón para que el usuario pueda ver los datos que ha cargado. Para ello usamos el método .button() y los mostraremos en formato dataframe

Para celebrar que hemos leído los datos, vamos a lanzar unos globos cuando el usuario puse el botón de ver los datos (st.balloons)

Ya tenemos los datos. Y ahora… vamos a parar un momento.

¿No os parece que la página está bastante desordenada? Además, se nos está acumulando el código en el script main. Esto no es una buena práctica.

Vamos a separar en distintas funciones los elementos que corresponden a la introducción sobre Super Store de los elementos para la carga de datos. Esas funciones las guardaremos en un script llamado functions.py.

Además, en main crearemos una estructura para que, dependiendo de la selección del menú, el usuario pueda ver la página de introducción, la página de carga de los datos o la página de análisis, que rellenaremos más adelante.

Para montar el menú utilizamos un selector. Para crear el selector utiliza el método st.selectbox() con el texto "Elige una sección:" y le damos las siguientes opciones: "Panorámica", "Carga de datos" y "Analiza las ventas".

Para que el selector sea siempre visible y que el usuario pueda cambiar de página cuando lo desee, lo vamos a colocar en el panel lateral utilizando el método st.sidebar().

Podéis explorar otras disposiciones de la aplicación en este enlace a la documentación de Streamlit: https://docs.streamlit.io/library/api-reference/layout

Ahora sí, vamos a analizar nuestros datos.

Análisis de los datos

En esta página vamos a ofrecer al usuario tres opciones de análisis, con el texto "Escoge lo que te interesa analizar".

Las opciones serán "Ventas por categoría", "Ventas por subcategoría" y "Ventas por Estado".

Para preparar este menú utilizaremos el método st.radio() y deberá aparecer en el panel lateral de la página.

Ventas por categoría

En esta página mostraremos una tabla y una gráfica de barras (bar plot) montado con Plotly.

- La tabla deberá mostrar el importe de las ventas por categoría de producto y el porcentaje que representan sobre el total de las ventas.

  Deberá aparecer en formato tabla en la página, no en formato dataframe. Busca en la documentación el método de streamlit que te permite hacerlo.

- Bar plot de ventas por año y categoría.

  El bar plot deberá aparecer debajo de la tabla. Deberá reflejar las ventas totales por año, mostrando las categorías de producto a través del color. Para conseguir esta información, filtra el dataframe usando el método .groupby() de Pandas.

  En el anexo I de este documento tenéis el código para montar la gráfica. Veréis que al px.bar se le pasa el dataframe "df_ventas_año_cat", que es el resultado de aplicar al dataframe completo el groupby antes indicado.

  Investigad qué método necesitáis para para mostrar un objeto de plotly en Streamlit.

Ventas por Subcategoría

En esta página mostraremos un bar plot creado con Plotly y un line chart creado con un método propio de Streamlit.

- Bar plot mostrando las ventas por subcategoría

  El bar plot deberá reflejar las ventas por subcategoría de producto, mostrando las categorías a través del color de las barras. Utilizad un .groupby() de Pandas para preparar el dataframe.

  Además, debéis filtrar por años la información que muestra el bar plot, utilizando un

slider de Streamlit (st.slider()). Para filtrar por años los datos que le pasáis a la gráfica podéis utilizar una máscara.

Tenéis el código de esta gráfica de plotly en el anexo II.

- Line chart mostrando la evolución de las ventas por subcategoría de producto desde el 2013 al 2016.

  El line chart debéis crearlo con el método de Streamlit st.line_chart().

  Debéis filtrar por años la información que muestra el line chart, utilizando un slider de Streamlit (st.slider()). Para filtrar por años los datos que le pasáis a la gráfica podéis utilizar una máscara.

  Además, debéis utilizar un selector múltiple para mostrar en el line chart únicamente los subproductos que quiera ver el usuario. Usad para ello el método st.multiselect().

El orden de los elementos de esta página deberá ser el siguiente:

1. Slider del bar plot de ventas por subcategoría de producto.
2. Bar plot de ventas por subcategoría de producto.
3. Slider del line chart.
4. Multiselector de subcategorías.
5. Por último, el line chart.

Ventas por Estado

En esta última página vamos a mostrar un mapa que refleje las ventas por Estado.

Para ello, lo primero que tenéis que hacer es cargar las coordenadas de los estados con el método read_csv() de Pandas. El csv se llama “statelatlong.csv”.

Después debéis utilizar un groupby de Pandas para obtener las ventas por estado y año.

Acto seguido debéis dar al usuario la posibilidad de filtrar la información del mapa por años. Para ellos debéis usar un selectbox de Streamlit (st.selectbox()). El año a mostrar lo guardaréis en una variable denominada “`year_to_show`”.

El código para montar el mapa lo tenéis en el anexo III de este documento.

¡Esperamos que os guste el resultado de vuestro trabajo!

*****

Anexo I

```
fig1 = px.bar(df_ventas_año_cat,
              x="Order Date Year",
              y = "Sales",
             color='Category',
             template="plotly_white",
             labels={"Order Date Yea":''},
             color_discrete_sequence = px.colors.qualitative.Safe,
             height=500,
             width=600)

    fig1.update_layout(
        xaxis = dict(
            tickmode = 'linear',
            tick0 = 0,
            dtick = 1))

    fig1.update_layout(font=dict(size=9),title_text="Ventas de artículos por
categoría y año")
```

Anexo II

```
fig2 = px.bar(data,
          x="Sub-Category",
          y = "Sales",
         color='Category',
         template="plotly_white",
         labels={"Order Date Year":'',"Sub-Category":" "},
         color_discrete_sequence = px.colors.qualitative.Safe,
         height=500,
         width=600)

fig2.update_layout(
    xaxis = dict(
        tickmode = 'linear',
        tick0 = 0,
        dtick = 1))

fig2.update_layout(font=dict(size=11))
```

Anexo III

```
view = pdk.ViewState(latitude=37, longitude=-95, zoom=3,)

    tooltip = {
        "html":
            "<b>Estado:</b> {State} <br/>"
            "<b>Ventas:</b> {Sales} <br/>",
        "style": {
            "backgroundColor": "steelblue",
            "color": "black",
        }
    }
    salesLayer = pdk.Layer(
            type= "ScatterplotLayer",
            data=df_map,
            pickable=True,
            opacity=0.3,
            filled=True,
            onClick=True,
            radius_scale=10,
            radius_min_pixels=0,
            radius_max_pixels=30,
            line_width_min_pixels=1,
            get_position=["Longitude", "Latitude"],
            get_radius="Sales",
            get_fill_color=[252, 136, 3],
            get_line_color=[255,0,0],
        )
    layertext = pdk.Layer(
        type="TextLayer",
        data=df_map,
        pickable=False,
        get_position=["Longitude", "Latitude"],
        get_text="Sales",
        get_size=3000,
        sizeUnits='meters',
        get_color=[0, 0, 0],
        get_angle=0,
        getTextAnchor= '"middle"',
        get_alignment_baseline='"bottom"'
    )

    r = pdk.Deck(
        layers=[salesLayer, layertext,],
        initial_view_state=view,
```

```
        map_style="mapbox://styles/mapbox/light-v10",
        tooltip=tooltip,
    )
    map = st.pydeck_chart(r)
    salesLayer.data = df_map[df_map['Order Date Year'] == year_to_show]
    layertext.data = df_map[df_map['Order Date Year'] == year_to_show]

    r.update()
    map.pydeck_chart(r)
```