**Descripción:** Segundo elemento del modelo LIVE y es contenido desarrollado por el experto en formato electrónico Word. La finalidad es profundizar en los temas definidos en este bloque. Los elementos del modelo INSPIRA que se aplican son: **Nutre y Significa el contenido y Practica.**

**Instrucciones :** Se realizará como una secuencia de contenidos que se integrará y producirá en diferentes recursos formativos a publicar en la plataforma digital CANVAS. Por lo que se solicita que se desarrollen 18-20 cuartillas de contenido de autoría de experto en Arial 12 (incluyendo actividades de práctica)

**Duración: 2h + 30 min de evaluación**

**Es información tomada del Mapa de Contenidos**

|  |  |
| --- | --- |
| **Nombre del experto** | Jesús Aguilar González |
| **Nombre del programa** | Data Science A&I |
| **Nombre del módulo** | Aplicación Web de Ciencia de Datos |
| **Objetivos del elemento PROFUNDIZAR)**  **(trabajo asincrónico)** | **Relacionarás las características y atributos escenciales de la plataforma Streamlit para la construcción de dashboard** |
| **Contenido** | TEMA 1: Plataforma Streamlit  TEMA 2: Sidebar  TEMA 3: Estatutos slider, radio, selectbox  TEMA 4: Gráficas en Streamlit |
| **Take away** | Conocimientos par construir una aplicación web de analítica de datos |

1. **Bienvenida (video inicial)**

Bienvenido al quinto módulo del diplomado Data Science & AI: Del Concepto a desarrollo de aplicaciones. En el módulo anterior aprendiste a visualizar bases de datos por medio de la librería Pandas de Python y aplicaste técnicas y métodos para personalizar y estilizar dichas visualizaciones.

Las aplicaciones web son una de las formas más convenientes de mostrar el trabajo que se realiza en la ciencia de datos. La creación de aplicaciones web puede resultar abrumadora para muchos científicos de datos si no tienen experiencia en desarrollo web. Sin embargo, con el marco optimizado, las aplicaciones web ya no son difíciles para los científicos de datos; si se conoce el lenguaje Python, se pueden crear aplicaciones web interactivas por ejemplo con streamlit; este marco increíble hace el trabajo difícil para hacer y diseñar elementos web y dejar que el científico de datos simplemente se centre en la parte de los datos y en el análisis.

En esta evolución de marcos que permiten la creación de sitios web para la ciencia de datos existen varias opciones y a continuación se presentará una perspectiva histórica y como han ido evolucionando (ver figura 1). Todo comienza en el 2012 con el surgimiento de RStudio Shiny. Shiny es un paquete de R que facilita la creación de aplicaciones web interactivas directamente desde el lenguaje R. Puede alojar aplicaciones independientes en una página web o incrustarlas en documentos de R Markdown o crear paneles de control. También puede ampliar sus aplicaciones Shiny con temas CSS, htmlwidgets y acciones de JavaScript.

A finales del 2015 aparece Jupiter Notebook, que es una aplicación web de código abierto que permite crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo. Los usos incluyen: limpieza y transformación de datos, simulación numérica, modelado estadístico, visualización de datos, aprendizaje automático y mucho más. Su uso se volvió un referente y desde esa fecha ha incrementado su popularidad.

Posteriormente a finales del 2017 surge Plotly Dash que es un marco que permite construir aplicaciones con sencillos pasos y lo más importante es que puede ser en los lenguajes de programación R, Python o Julia. Ha sido el éxito tan grande detrás de este marco, que muchos científicos de datos han determinado que es la interfaz para los modelos de ciencia de datos y aprendizaje automático; su crecimiento y popularidad han ido en aumento.

Sin embargo, a finales de 2019 principios del 2020 surge Streamlit y su crecimiento y aceptación entre los científicos de datos ha sido meteórica pues en un corto tiempo ha tenido una aceptación tan grande que ahora se compara con Dash y Jupyter Notebook, incluso se ha determinado que pronto pudiera desbancar a Jupyter Notebook.

En este módulo aprenderemos a crear aplicaciones web utilizando Streamlit para poder generar tableros interactivos que permitan visualizar datos de una forma sencilla pero poderosa. Es importante recordar que es necesario tomar en cuenta los conceptos sobre visualización que estuvimos revisando en el aprende pues son importantes a considerar cuando se esta construyendo un tablero interactivo.

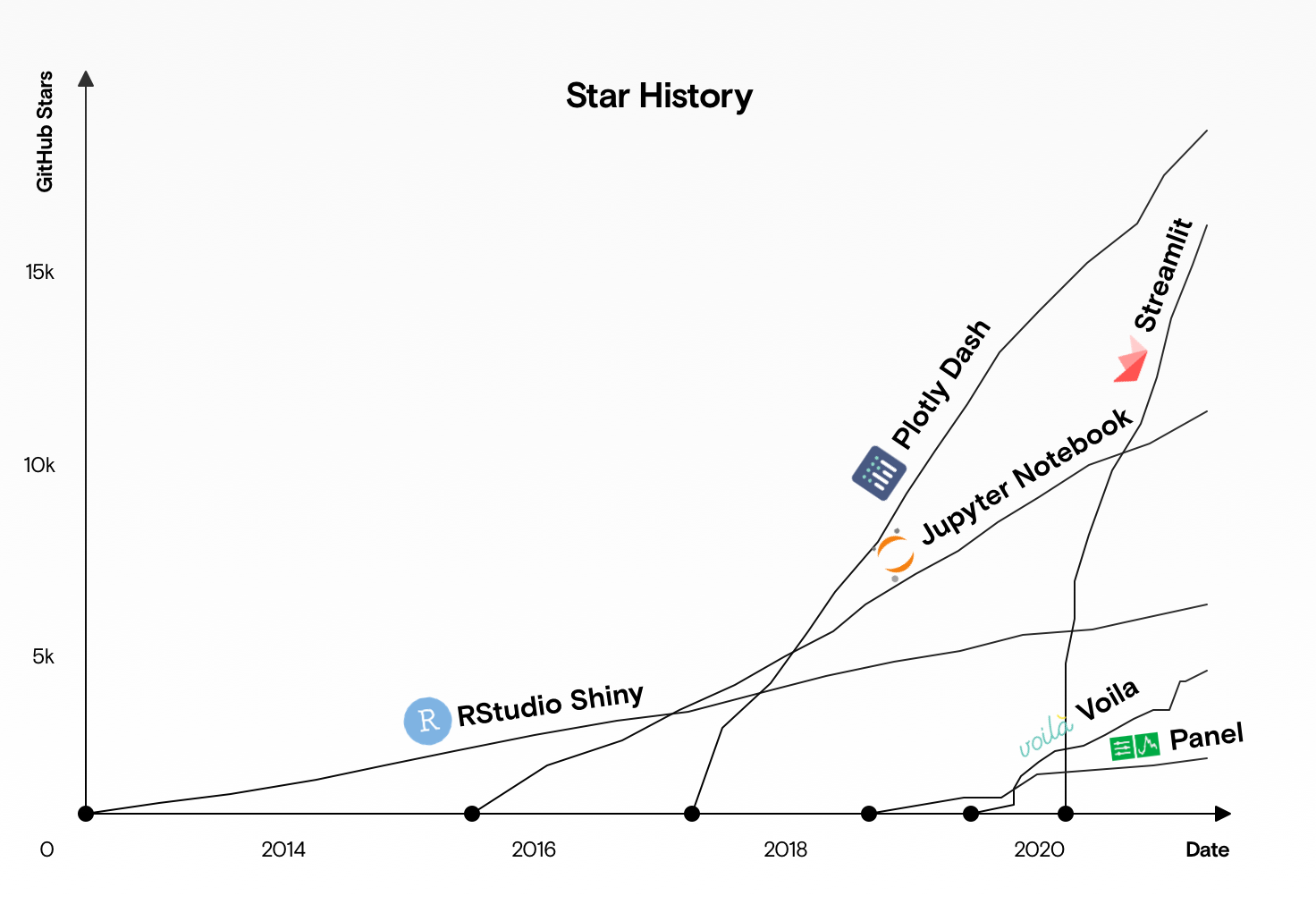


Figura 1. Historia del desarrollo de plataformas para aplicaciones web de analítica de datos

Se incluye el contexto de los contenidos a revisar en este bloque asincrónico (objetivos) y el resultado esperado, se incluyen preguntas de reflexión y se puede hacer una conexión entre lo revisado en la sesión sincrónica y lo que se verá en este bloque.

1. **Desarrollo del contenido por temas (esto se desarrolla por cada tema incluido en este bloque)**

TEMA 1: Plataforma Streamlit

TEMA 2: Sidebar

TEMA 3: Estatutos slider, radio, selectbox

TEMA 4: Gráficas en Streamlit

Hacer un colab dividido por tema y subtemas en donde se incluya todos los códigos **del nutre**. Este colab se pondrá al final del profundiza, en material adicional para que los participantes puedan consultarlo y ejecutarlo en caso de ellos no consigan replicar las indicaciones del nutre. **(INCLUIR COLAB GUIA NUTRE que va a hacer Grettel)**

GUIÁ PARA EL DESARROLLO DE CONTENIDOS:

**TEMA 1 Plataforma Streamlit**:

Streamlit es un paquete de Python abierto que le ayuda a crear aplicaciones web interactivas desplegables sin ningún conocimiento de HTML o CSS, etc. Python es todo lo que necesita. Lo mejor de Streamlit es que puede actualizar automáticamente sus aplicaciones web cada vez que se modifican las entradas de los códigos fuente.

Existen numerosas bibliotecas de visualización que Streamlit admite para construir cuadros de mando interactivos, como Plotly, Altair, Bokeh, etc. Con estos, contar una historia a través de gráficos de barras, circulares, lineales, etc., nunca ha sido más sencillo. Es posible con pocas líneas de código crear un tablero interactivo simple pero encantador con solo algunas librerías fundamentales como Pandas, Numpy y Plotly.

Aquí hay una demostración simple de un tablero creado con Streamlit, y está implementado en Heroku para una referencia de fácil acceso, ver el siguiente enlace: <https://sheltered-ridge-30828.herokuapp.com/>

**NUTRE**

El primero paso que debemos hacer es abrir un archivo en Google Colab, como se muestra en la siguiente figura (ver figura 2)

Graphical user interface, text, application, email

Description automatically generated

Figura 2. Archivo inicial de Google Colab

Para usar Streamlit, primero debe instalarlo usando el comando:

!pip install streamlit

Cuando se ejecuta esa instrucción al final aparecerá un mensaje de alerta en color rojo que indica que es necesario realizar una reinicialización pues se encontrarón problemas (ver figura 3)

Graphical user interface, text, application, email

Description automatically generated

Figura 3. Aviso de alerta de la instalación de la librería streamlit.

Para corregir este problema, solo es necesario dar clic en el botón que dice “Restart Runtime” y aparecerá un mensaje (ver figura 4) que pregunta si se desea reiniciar el sistema.

Graphical user interface, text, application

Description automatically generated

Figura 4. Ventana de reinicialización del entorno de ejecución.

Al dar clic en la opción de “Si” se reinicializará de nuevo el sistema. Si se quiere comprobar que ya se corrigió el problema, solo basta volver a ejecutar la instrucción de instalación de la librería de streamlit y se verá como el mensaje de alerta desaparece.

Posteriormente vamos a instalar ngrok; ngrok proporciona una interfaz de usuario web en tiempo real donde se puede reproducir una aplicación web para determinar como se vería en un servidor de web real.

En la siguiente instrucción de nuestro Google Colab vamos a cargar una máquina virtual que que contiene el instalador de ngrok (ver figura 5). Esto lo hacemos con la siguiente instrucción:

!wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip

Graphical user interface, text, application, email

Description automatically generated

Figura 5. Máquina virtual con el instalador de Ngrok.

Ya que tenemos la maquina virtual en nuestro Google Colab lo que tenemos que hacer es ejecutar una instrucción para descomprimir dicho archivo en el entorno de trabajo de Google Colab (ver figura 6). Y esto lo podemos hacer con la siguiente instrucción:

!unzip ngrok-stable-linux-amd64.zip

Graphical user interface, text, application, Teams

Description automatically generated

Figura 6. Descomprimiendo la máquina virtual en el entorno de trabajo de Google Colab.

Una vez que se ha descomprimido el archivo es necesario generar el servidor temporal en donde se ejecutará nuestra aplicación web y en donde se le indica a Ngrok que vamos a utilizar el puerto 8501 (ver figura 7) y para lo cual es necesario ejecutar la siguiente instrucción.

get\_ipython().system\_raw('./ngrok http 8501 &')

Graphical user interface, text, application, email

Description automatically generated

Figura 7. Instrucción para activar puerto 8501 para la aplicación ngrok.

Adicionalmente es necesario incluir una instrucción que le indique a ngrok que deberá generar un túnel para que el servidor que estamos creando se pueda ejecutar en el entorno de Google Colab (ver figura 8). Y la instrucción sería la siguiente:

!curl -s http://localhost:4040/api/tunnels | python3 -c \

'import sys, json; print("Execute the next cell and the go to the following URL: " +json.load(sys.stdin)["tunnels"][0]["public\_url"])'

Graphical user interface, text, application, email

Description automatically generated

Figura 8. Creación del túnel en ngrok para generar el url de la aplicación web.

Una vez que ejecutaron la instrucción anterior, verán que aparece un URL que es el que vamos a dar clic para visualizar nuestra aplicación, pero primero tenemos que ejecutar nuestra aplicación.

Antes de explicar como carga y ejecutar la aplicación web de nuestro proyecto, primero tenemos que aprender a crear un archivo para que pueda ser interpretado por streamlit. Así que vamos a explicar como funciona esto.

En un editor de texto, vamos a empezar a codificar el archivo que contendrá nuestra aplicación. La primera instrucción es:

import streamlit as st

Esta instrucción nos permite importar la librería streamlit para empezar a crear la aplicación.

Una de las instrucciones más útiles es st.write que nos permite escribir información que se visualizará en nuestra aplicación. Vamos a utilizar esa instrucción para crear un título. A continuación se muestra como quedaría dicha instrucción:

st.write("""

# Ejemplo Simple de una App de Prediccion

Esta app predice mis datos!

""")

Con esas dos instrucciones podemos generar nuestro primer archivo que vamos a probar en Google Colab. Vamos a grabar ese archivo con el nombre primero.py y una vez hecho esto, ahora vamos a colocarlo en nuestro Google Colab. A continuación se muestra como quedaría el código completo.

import streamlit as st

st.write("""

# Simple Example Prediction App

This app predicts my data!

""")

Hay dos formas de ejecutar este código en Google Colab. La primera opción es cargar el archivo primero.py en nuestro Google Colab, para hacerlo vamos a dar clic en el icono del lado derecho que tiene la imagen de un folder, al hacerlo se abrirá un espacio como el que se muestra en la figura 9.

Graphical user interface, text, application, email

Description automatically generated

Figura 9. Sección para cargar archivos complementarios para Google Colab.

Vamos a dar clic en el icono que tiene una hoja con una flecha hacia arriba y ahí nos permitirá subir el archivo que acabamos de hacer.

Una vez que se ha seleccionado el archivo, este deberá aparecer en la sección de archivos como se muestra en la figura 10.

Graphical user interface, text, application, email

Description automatically generated

Figura 10. Sección de archivos con el archivo primero.py

Finalmente, tenemos que agerar una última instrucción, que es la llamada a streamlit para que ejecute el archivo de Python que hicimos anteriormente y que contiene nuestra aplicación web (en este ejercicio el archivo de denomina “primero.py”) (ver figura 11). La instrucción quedaría de la siguiente forma:

!streamlit run /content/primero.py

Graphical user interface, text, application, email

Description automatically generated

Figura 11. Instrucción para ejecutar el archivo primero.py

Para ejecutar el servidor temporal en donde se podrá visualizar la aplicación web, deberá dar clic en el url que aparece en la instrucción anterior.

Execute the next cell and the go to the following URL: [http://a9fe7569e5e1.ngrok.io](http://a9fe7569e5e1.ngrok.io/" \t "_blank)

Se cargará una pestaña en su navegador que este predeterminado y podrá visualizar la aplicación web que hemos creado (ver figura 12).

Graphical user interface, application, Word

Description automatically generated

Figura 12. Vista de la aplicación web con Streamlit y Google Colab

La segunda opción es insertar el código en nuestro Google Colab y solo se le antepone el siguiente comando:

%%writefile first.py

Y se escribe todo el código. Para fines prácticas en los ejemplos que se anexan, así es como se presentarán cada uno de las prácticas de los temas sobre Streamlit.

* **SIGNIFICA**

Ahora que ya aprendiste como visualizar una aplicación web utilizando el marco de referencia de streamlit, como podrías construir un sitio web que contenga lo siguiente:

* + - ¿Cómo se podría poner un título y un subtítulo?
    - ¿Cuál sería la instrucción para poder visualizar un breve párrafo introductorio que describa el proyecto?
    - ¿Cómo se podría incluir la fecha en la que se realizó dicho proyecto?
* **ACTIVATE**
  + - Te invito a que revises el siguiente enlace en donde encontrarás las instrucciones que se requieren para contestar a las interrogantes planteadas anteriormente. <https://docs.streamlit.io/en/stable/api.html#display-text>
* **PRACTICA TEMA 1** 
  + **Es momento de poner en práctica lo que aprendiste en este tema.**

**Ejemplo 1 Actívate:**

***Descripción breve:***

**Deberás construir la estructura básica de una aplicación web que de la bienvenida al proyecto de visualización de analítica de datos para WalMart USA. En el siguiente enlace podrás encontrar el enlace al conjunto de datos que pertencen a WaltMart USA. <https://raw.githubusercontent.com/jeaggo/tc3068/master/Superstore.csv>**

***Instrucciones generales***

**Utilizando las instrucciones para desplegar texto en streamlit se deberá construir la estructura general de una aplicación web para analizar los datos de WalMart USA.**

***Ejercicio:***

**La estructura de la aplicación web deberá tener lo siguiente:**

* + **Un título para el proyecto.**
  + **Un subtitulo (este puede ser opcional) para el proyecto.**
  + **Un breve párrafo que describa el objetivo de este proyecto de analítica de datos; por ejemplo: predicción de ventas de productos de línea blanca en el noroeste de los Estados Unidos.**
  + **La fecha en la que se realizó dicho proyecto.**

**TEMA 2: Sidebar**

Hay varios marcos de trabajo de Python versátiles para el desarrollo web, incluidos Flask y Django. Sin embargo, cuando se trata del mundo de la ciencia de datos, los científicos de datos o los ingenieros de aprendizaje automático no son desarrolladores web y no están interesados en pasar semanas aprendiendo a usar estos marcos para crear aplicaciones web. En cambio, quieren una herramienta que sea más fácil de aprender y usar, siempre que pueda mostrar datos y recopilar los parámetros necesarios para el modelado.

Aquí es donde entra en juego Streamlit: un marco web fácil de usar que nos permite crear aplicaciones web interactivas basadas en datos. Si ya se conoce bien el lenguaje de programación Python, tal vez sea necesario unas pocas horas antes de que se pueda crear una aplicación web significativa. Ciertamente, aprender a usarlo bien requiere algunas pruebas y errores. En esta sección empezaremos a revisar una de las formas básicas para ir construyendo una aplicación web usando Streamlit.

**NUTRE**

Si la aplicación web debe mostrar datos y cifras relacionadas según la entrada del usuario, es una buena idea colocar widgets relevantes en la barra lateral (sidebar). Esta separación deja en claro la intención de la interactividad para los usuarios: la barra lateral (sidebar) es para proporcionar parámetros (es decir, la entrada) mientras que la pantalla principal es para proporcionar comentarios (es decir, la salida) en respuesta a la entrada de los usuarios.

La siguiente figura (ver figura 13) solo le da una idea. Si tiene más parámetros, el uso de la barra lateral (sidebar) puede mejorar la limpieza de la aplicación. Sin embargo, debe tenerse en cuenta que el ancho de la barra lateral es limitado, así que hay que evitar los textos largos y los widgets grandes, que pueden hacer que el flujo de información se sienta roto. Además, si se tienen demasiados parámetros que exceden la altura de la pantalla, es un buen momento para pensar en reorganizar los widgets.

Graphical user interface, application

Description automatically generated

Figura 13. Uso de la barra lateral (sidebar)

Vamos a construir el archivo que contendrá una barra lateral que servirá para alojar los datos de entrada del proyecto de ciencia de datos a realizar; así mismo dicho archivo contendrá la información que se desplegará en la página principal.

import streamlit as st

# Crear el título para la aplicación web

st.title("Mi Primera App con Streamlit")

sidebar = st.sidebar

sidebar.title("Esta es la barra lateral.")

sidebar.write("Aquí van los elementos de entrada.")

st.header("Información sobre el Conjunto de Datos")

st.header("Descripción de los datos ")

st.write("""

Este es un simple ejemplo de una app para predecir

¡Esta app predice mis datos!

""")

La primera instrucción que tendrá nuestro archivo es la importación de la librería streamlit.

import streamlit as st

La segunda instrucción es crear un título para nuestra aplicación.

st.title("Mi Primera App con Streamlit")

Posteriormente vamos a crear una sección del lado derecho que se denomina barra lateral o sidebar en inglés. Para esto se utiliza la siguiente instrucción:

sidebar = st.sidebar

Ya que tenemos creada la barra lateral, lo que tenemos que agregar es un título y un párrafo que explique que es lo que vamos a encontrar en dicha sección. Y las instrucciones para hacer eso se muestran a continuación:

sidebar.title("Esta es la barra lateral.")

sidebar.write("Aquí van los elementos de entrada.")

Ahora bien, vamos a generar unos subtitulos que nos ayuden a dar más contexto sobre nuestra aplicación y para hacerlo vamos a utilizar la instrucción st.header. A continuación, se muestran las dos instrucciones que se utilizaron:

st.header("Información sobre el Conjunto de Datos")

st.header("Descripción de los datos ")

Finalmente vamos a poner un pequeño párrafo que nos ayude a mostrar una breve descripción de nuestra aplicación web. Para lo cual vamos a utilizar la instrucción que ya habíamos visto en el tema anterior, st.write. A continuación se muestra dicha instrucción:

st.write("""

Este es un simple ejemplo de una app para predecir

¡Esta app predice mis datos!

""")

Este archivo lo puede editar en cualquier editor de texto (notepad; sublime, etc) y deberá guardarlo con el nombre de segundo.py. Hay que recordar que este archivo se deberá colocar en la sección de archivos del Google Colab principal que se tiene que hacer para gener el url que nos permitirá visualizar la aplicación web.

En el ejemplo de este tema encontrarán el código inserto en el Google Colab para que lo puedan ejecutar.

Se cargará una pestaña en su navegador que este predeterminado y podrá visualizar la aplicación web que hemos creado (ver figura 14).

Graphical user interface, text, application

Description automatically generated

Figura 14. Vista de la aplicación web con Streamlit y Google Colab usando una barra lateral (sidebar).

* **SIGNIFICA**

Ahora que ya aprendiste como generar una barra horizontal para una aplicación web utilizando el marco de referencia de streamlit, como podrías construir un sitio web que contenga lo siguiente:

* + - Una sección de bienvenida que contenga una barra lateral
    - Una sección que contenga un apartado para tener checkboxes
    - Una sección que contenga un apartado para tener selectboxes
* **ACTIVATE**
  + - Te invito a que revises el siguiente enlace en donde encontrarás las instrucciones que se requieren para contestar a las interrogantes planteadas anteriormente. <https://docs.streamlit.io/en/stable/getting_started.html>
* **PRACTICA TEMA 2** 
  + **Es momento de poner en práctica lo que aprendiste en este tema.**

**Ejemplo 1 Actívate:**

***Descripción breve:***

**Deberás construir la estructura básica de una aplicación web que contenga una sección principal y además un apartado denominado barra lateral, el cual contendrá algunos controles para que se pueda manipular los datos sobre el proyecto de visualización de analítica de datos para WalMart USA.**

**En el siguiente enlace podrás encontrar el enlace al conjunto de datos que pertencen a WaltMart USA. <https://raw.githubusercontent.com/jeaggo/tc3068/master/Superstore.csv>**

***Instrucciones generales***

**Utilizando las instrucciones para desplegar una barra lateral en streamlit se deberá construir la estructura básica general de una aplicación web para analizar los datos de WalMart USA.**

***Ejercicio:***

**La estructura de la aplicación web deberá tener lo siguiente:**

* + **Un título que describa el proyecto de analítica de datos.**
  + **Un subtítulo (este puede ser opcional) que de más contexto sobre el proyecto de analítica de datos.**
  + **Un breve párrafo que describa el objetivo de este proyecto de analítica de datos; por ejemplo: predicción de ventas de productos de línea blanca en el noroeste de los Estados Unidos.**
  + **Una sección del lado derecho a manera de una barra lateral que contenga dos secciones una para los controles de checkboxes y otra para los controles selectboxes. Adicionalmente deberá tener una descripción explicando el porque de la barra lateral. Nota: Los apartados para los controles solo se deberán indicar, no se espera que se incluyan los controles, solo indicar que ahí estarán.**

**TEMA 3: Estatutos slider, radio, selectbox**

Con los widgets, Streamlit permite crear interactividad directamente en las aplicaciones con casillas de verificación, botones, controles deslizantes y más. Esto es una de las cosas más significativas pues permite al científico de datos crear elementos que permitan que el análisis que está realizando sea más profundo.

Es recomedable indicar que existen muchos widgets pero nos vamos a concentrar en los 3 más importantes: controles deslizantes o sliders; botones de opciones ó radio buttons y casillas de verificación o selectbox.

**NUTRE**

Para recopilar las respuestas de los usuarios, tenemos varias opciones para crear widgets de opción múltiple. El primer widget es el botón de opción (radio button). La instrucción st.radio se debe aplicar a una variable que para este ejemplo se denominará selected\_class; la instrucción tiene dos secciones, una que indica el título del botón de opción y la otra que indica la información que alimentará dicho botón y para este ejemplo es la columna “clase” del conjunto de datos del Titanic. Adicionalment es necesario utilizar la instrucción st.write para plasmar el botón de opción en nuestra aplicación y además capturar la opción que el usuario haya seleccionado.

|  |
| --- |
| selected\_class = st.radio("Select Class", titanic\_data['class'].unique()) |
|  |

|  |
| --- |
| st.write("Selected Class:", selected\_class) |
|  |

Verá algo como se muestra a continuación (ver figura 15). Tenga en cuenta que la opción seleccionada será el valor real de las opciones mostradas en lugar del índice numérico seleccionado. En este caso, será el texto.

A picture containing background pattern

Description automatically generated

Figura 15. Selección de opciones con el botón de opción (radio button).

El segundo widget es la casilla de verificación (selectbox), que proporciona una lista desplegable desde la que puede hacer su elección. El código será así:

selected\_sex = st.selectbox("Select Sex", titanic\_data['sex'].unique())  
st.write(f"Selected Option: {selected\_sex!r}")

La casilla de verificación es mucho más sencilla, pues solo requiere de dos instrucciones. La primera instrucción es la asignación a una variable, que para este ejercicio se denomina “selected\_sex”; a la instrucción st.selectbox. La instrucción st.selectbox tiene dos parámetros: el nombre de la casilla de verificación y el conjunto de datos de donde se utilizará para generar dicha casilla; en este caso, se utiliza la columna denominada “sex” del conjunto de datos del Titanic. Posteriormente la segunda instrucción es utilizar st.write para desplegar la casilla de verificación. Nota: En la instrucción st.write hay que notar que para el titulo se antepone la letra “f” que indica que se capturara información y en la sección donde se indica la variable al final se pone el signo de admiración y la letra “r” esto indica que se escribirá la opción que se haya elegido.

La interacción será como la siguiente (ver figura 16).

A picture containing background pattern

Description automatically generated

Figura 16. Selección de opciones con la casilla de verificación (selectbox).

Además de este widget, streamlit también proporciona un control deslizante (slider) que permite al usuario especificar una entrada de número sin teclear nada. A continuación, se muestra el código completo.

optionals = st.beta\_expander("Optional Configurations", True)

fare\_select = optionals.slider(

"Select the Fare",

min\_value=float(titanic\_data['fare'].min()),

max\_value=float(titanic\_data['fare'].max())

)

subset\_fare = titanic\_data[(titanic\_data['fare'] >= fare\_select)]

st.write(f"Number of Records With this Fare {fare\_select}: {subset\_fare.shape[0]}")

La primera instrucción establece una variable que contendrá el expansor (se explica posteriormente este) y que lo activará.

Posteriormente se establece una variable que contendrá el valor seleccionado en la barra deslizante, para este ejemplo se utilizará las tarifas del conjunto de datos del Titanic. Esta variable utiliza la instrucción nombre\_expansor.slider para generar los máximos y mínimos. Cabe mencionar que esta instrucción utiliza datos que son númericos, como precios, costos, temperatura, etc.

Adicionalmente hay que definir una variable que para este ejemplo se denomina “subset\_fare” que es la contendrá el conjunto de datos máximos y mínimos de los datos que se utilizarán para generar las barras deslizantes.

Finalmente se tiene que utilizar la instrucción st.write que nos ayudará a plasmar las barras deslizantes en nuestra aplicación web. Nota: Observe que se debe de anteponer la letra “f” al titulo que aparecerá debajo de la barra deslizante.

Con el widget de control deslizante (slider), especifica el nombre del widget. Mientras tanto, establece los valores mínimo y máximo. Además del control deslizante, el código anterior incluye otra característica llamada expansor (st.beta\_expander). Esta función es útil cuando desea ocultar algunos widgets opcionales. Puede especificar si desea que el expansor se expanda de forma predeterminada. La siguiente figura (ver figura 17) muestra un control deslizante interactivo de este tipo.

A picture containing line chart

Description automatically generated

Figura 17 – Expansor y Control deslizante

A continuación, se presenta el código completo y se explicará como ejecutarlo en Google Colab.

import pandas as pd

import streamlit as st

titanic\_link = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.csv'

titanic\_data = pd.read\_csv(titanic\_link)

# Create the title for the web app

st.title("My First Streamlit App")

sidebar = st.sidebar

sidebar.title("This is the sidebar.")

sidebar.write("You can add any elements to the sidebar.")

st.header("Dataset Overview")

st.dataframe(titanic\_data)

st.header("Data Description")

selected\_class = st.radio("Select Class", titanic\_data['class'].unique())

st.write("Selected Class:", selected\_class)

st.markdown("\_\_\_")

selected\_sex = st.selectbox("Select Sex", titanic\_data['sex'].unique())

st.write(f"Selected Option: {selected\_sex!r}")

st.markdown("\_\_\_")

optionals = st.beta\_expander("Optional Configurations", True)

fare\_select = optionals.slider(

"Select the Fare",

min\_value=float(titanic\_data['fare'].min()),

max\_value=float(titanic\_data['fare'].max())

)

subset\_fare = titanic\_data[(titanic\_data['fare'] >= fare\_select)]

st.write(f"Number of Records With this Fare {fare\_select}: {subset\_fare.shape[0]}")

st.markdown("\_\_\_")

Este archivo lo puede editar en cualquier editor de texto (notepad; sublime, etc) y deberá guardarlo con el nombre de tercero.py. Hay que recordar que este archivo se deberá colocar en la sección de archivos del Google Colab principal que se tiene que hacer para gener el url que nos permitirá visualizar la aplicación web. Vamos a recordar como se tiene que hacer.

En el ejemplo de este tema encontrarán el código inserto en el Google Colab para que lo puedan ejecutar.

Se cargará una pestaña en su navegador que este predeterminado y podrá visualizar la aplicación web que hemos creado (ver figura 18).

Graphical user interface, table

Description automatically generated

Figura 18. Vista de la aplicación web con Streamlit y Google Colab usando una barra lateral (sidebar) y los controles botón de opción; casilla de verificación y control deslizante.

* **SIGNIFICA**

Ahora que ya aprendiste como generar los controles más comunes en Streamlit para una aplicación web, como podrías construir un sitio web que contenga lo siguiente:

* + - ¿Cuál sería el código para generar un control de checkbox para las categorías: 1, 2 y 3?
    - ¿Cuál sería el código para generar un control de selectbox para las regiones: Norte y Sur?
    - ¿Cuál sería el código para generar un control de slider para máximo precio: 10.00 y mínimo precio: 5:00?
* **ACTIVATE**
  + - Te invito a que revises el siguiente enlace en donde encontrarás las instrucciones que se requieren para contestar a las interrogantes planteadas anteriormente. <https://docs.streamlit.io/en/stable/getting_started.html>
* **PRACTICA TEMA 3** 
  + **Es momento de poner en práctica lo que aprendiste en este tema.**

**Ejemplo 1 Actívate:**

***Descripción breve:***

**Desarrollar un código sobre la estructura de una aplicación web que contenga 3 controles (radio, selectbox y slider) sobre el proyecto de visualización de analítica de datos para WalMart USA.**

**En el siguiente enlace podrás encontrar el enlace al conjunto de datos que pertencen a WaltMart USA. <https://raw.githubusercontent.com/jeaggo/tc3068/master/Superstore.csv>**

***Instrucciones generales***

**El conjunto de datos de WalMart USA contiene los siguientes campos:**

**Row ID - Indice del registro de la orden**

**Order ID - Indice de la orden**

**Order Date - Fecha de la orden**

**Ship Date - Fecha de embarque**

**Ship Mode - Modo de embarque**

**Customer ID - Clave del cliente**

**Customer Name - Nombre del cliente**

**Segment - Segmento**

**Country - Pais**

**City - Ciudad**

**State - Estado**

**Postal Code - Código postal**

**Region - Region**

**Product ID - Clave del producto**

**Category - Categoría**

**Sub-Category - Sub categoría**

**Product Name - Nombre del producto**

**Sales - Ventas**

**Quantity - Cantidad vendida**

**Discount - Descuento aplicado**

**Profit - Ganancia**

***Ejercicio:***

**La estructura de la aplicación web deberá tener lo siguiente:**

* + **Deberá incluir un titulo**
  + **Deberá incluir un subtitulo (este puede ser opcional)**
  + **Deberá incluir un breve párrafo que describa el objetivo de este proyecto de analítica de datos; por ejemplo: predicción de ventas de productos de línea blanca en el noroeste de los Estados Unidos.**
  + **Una sección que contenga los controles antes mencionados. Se sugiere los siguientes campos para cada control: El campos Ship Mode para el control radio. El campo Category para el control selectbox. El campo Discount para el control slider.**

**TEMA 4: Gráficas en Streamlit**

Streamlit admite varias bibliotecas de gráficos diferentes, como Altair, Bokeh, Matplotlib e incluso Plotly. También soporta bibliotecas de gráficos interactivos como Vega Lite (gráficos en 2D) y deck.gl (mapas y gráficos en 3D). Sin embargo, también proporciona algunos tipos de gráficos que son nativos de Streamlit como st.line\_chart que te permite crear un gráfico de líneas o st.area\_chart que te permite crear un gráfico de área.

Por ejemplo si vamos a utilizar la librería Seaborn la forma de hacerlo en Streamlit es utilizando la instrucción st.pyplot (figura).

Vamos a analizar un ejemplo para que se pueda comprender como se utiliza esta librería en Streamlit.

Se tiene la base de datos del Titanic

df = pd.read\_csv(titanic.csv)

Se genera la figura que vamos a visualizar con Seaborn, para este ejemplo vamos a generar una gráfica de barras.

figura = sns.barplot(x="class", y="survived", data=titanic\_data)

st.pyplot(figura)

La variable figura tiene la instrucción tal y como se maneja en la librería Seaborn para generar una gráfica de barra de los valores de la base de datos de Titanic.

Posteriormente, se llama a la instrucción st.pyplot para pasarle la variable figura e indicarle la gráfica que se desea visualizar.

Streamlit también puede generar gráficas de la librería Matplotib. Esta librería es una de las más populares y la primera librería de Python para la graficación de datos. Puede encontrar más información sobre esta librería es: <https://matplotlib.org/stable/index.html>

La estructura de las instrucciones para graficar con Matplotlib en Streamlit es la siguiente:

st.pyplot(fig)

Como podrán observae es la misma instrucción que se utiliza para graficar en la librería Seaborn. A continuación, veremos como se puede graficar en Matplotlib en Streamlit.

**NUTRE**

En esencia, cada aplicación web optimizada es una secuencia de comandos de Python. En la parte superior del archivo, importamos las dependencias necesarias. En este caso, utilizaremos las 3 principales, como se muestra a continuación.

import pandas as pd  
import streamlit as st  
import matplotlib.pyplot as ptl

Las 3 librerías que vamos a usar serán pandas para la manipulación de los datos; streamlit para construir la aplicación web y matplotlib.pyplot para visualizar las gráficas.

En cuanto al conjunto de datos, usaremos el conjunto de datos del Titanic, que descargaremos de Internet.

titanic\_link = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.csv'

titanic\_data = load\_dataset(titanic\_link)

La primera sección que va a tener nuestra aplicación web será la correspondiente al título y a un encabezado de nuestro sitio web.

# Create the title for the web app

st.title("My First Streamlit App")

st.header("Dataset Overview")

Posteriormente vamos a desplegar el contenido de nuestro conjunto de datos (dataframe) para que el usuario pueda visualizar la información que vamos a graficar.

st.dataframe(titanic\_data)

st.header("Data Description")

st.markdown("\_\_\_")

Y se verá algo como a continuación (ver figura 19).

Table

Description automatically generated

Figura 19. Resumen de los datos.

Nota importante: la instrucción st.markdown ayuda a dividir las secciones con una línea y es útil para determinar como se dividirá la información dentro de la aplicación web.

Ahora que ya tenemos la parte introductoria, vamos a construir nuestra primera gráfica. Vamos a graficar un histograma, para lo cual las instrucciones se muestran a continuación:

fig, ax = plt.subplots()

ax.hist(titanic\_data.fare)

st.header("Histograma del Titanic")

st.pyplot(fig)

st.markdown("\_\_\_")

La primera instrucción inicializa la variable fig que contendrá el histograma y la variable ax que nos servirá para construirlo. La siguiente instrucción nos permite construir el histograma con el campo “fare” de la base de datos del Titanic. La instrucción st.header nos sirve para poner un encabezado para nuestra gráfica. La instrucción st.pyplot nos ayude a graficar nuestro histograma en streamlit. Finalmente la instrucción st.markdown nos ayuda a generar una línea que sirva de división para nuestra gráfica.

El siguiente paso será generar una gráfica de barras. Para lo cual vamos a utilizar dos campos de nuestro conjunto de datos, el campo “class” que tiene que ver con la clase de pasajeros del Titanic y el campo “fare” que indica la tarifa que pago el pasajero. Para construir nuestra gráfica solo necesitamos las siguientes instrucciones:

fig2, ax2 = plt.subplots()

y\_pos = titanic\_data['class']

x\_pos = titanic\_data['fare']

ax2.barh(y\_pos, x\_pos)

ax2.set\_ylabel("Class")

ax2.set\_xlabel("Fare")

ax2.set\_title('¿Cuanto pagaron las clases del Titanic')

st.header("Grafica de Barras del Titanic")

st.pyplot(fig2)

st.markdown("\_\_\_")

La primera instrucción inicializa la variable fig que contendrá la gráfica de barras y la variable ax que nos servirá para construirlo. Las variables x\_pos y y\_pos nos sirven para cargar los datos que corresponden a los campos “class” y “fare” respectivamente y que vamos a pasarle a la gráfica de barras. Posteriormente la instrucción ax2.barh nos sirve para generar una gráfica de barras horizontal que contendrá los campos antes mencionados. Las instrucciones ax2.set\_xlabel y ax2.set\_ylabel nos sirven para poner etiquetas para el eje de las x y el de las y. La instrucción ax2.set\_title nos ayuda a poner un título para nuestra gráfica. La instrucción st.header como ya hemos visto, nos sirve para colocar un título para nuestra gráfica. La instrucción st.pyplot es la que le indica a Streamlit la gráfica que va a visualizar. Finalmente la instrucción st.markdown nos ayuda a generar una línea que sirva de división para nuestra gráfica.

La última gráfica que vamos a realizar es una gráfica de dispersión, para lo cual utilizaremos los campos “age” que corresponde a la edad de los pasajeros y “fare” que corresponde a la tarifa que pagaron los pasajeros del Titanic. A continuación se muestra el código para hacer dicha gráfica:

fig3, ax3 = plt.subplots()

ax3.scatter(titanic\_data.age, titanic\_data.fare)

ax3.set\_xlabel("Edad")

ax3.set\_ylabel("Tarifa")

st.header("Grafica de Dispersión del Titanic")

st.pyplot(fig3)

La primera instrucción inicializa la variable fig que contendrá la gráfica de dispersión y la variable ax que nos servirá para construirlo. La instrucción ax3.scatter es la instrucción de Matplotlib que nos sirve para construir una gráfica de dispersión y esta recibe dos parámetros que en este caso serán los campos “age” y “fare”. Las instrucciones ax3.set\_xlabel y ax3.set\_ylabel nos permiten colocar etiquetas en el eje de las x y en el eje de las y. Vamos a utilizar la instrucción st.header para colocar un título para nuestra gráfica. Finalmente usamos la instrucción st.pyplot para indicarle a Streamlit que visualice la gráfica que deseamos.

Una vez que se ha entendido el funcionamiento de nuestra aplicación web en donde se pueden graficar diferentes tipos de gráficos, se muestra el código completo para su mejor comprensión.

import pandas as pd

import matplotlib.pyplot as plt

import streamlit as st

titanic\_link = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.csv'

titanic\_data = pd.read\_csv(titanic\_link)

fig, ax = plt.subplots()

ax.hist(titanic\_data.fare)

st.header("Histograma del Titanic")

st.pyplot(fig)

st.markdown("\_\_\_")

fig2, ax2 = plt.subplots()

y\_pos = titanic\_data['class']

x\_pos = titanic\_data['fare']

ax2.barh(y\_pos, x\_pos)

ax2.set\_ylabel("Class")

ax2.set\_xlabel("Fare")

ax2.set\_title('¿Cuanto pagaron las clases del Titanic')

st.header("Grafica de Barras del Titanic")

st.pyplot(fig2)

st.markdown("\_\_\_")

fig3, ax3 = plt.subplots()

ax3.scatter(titanic\_data.age, titanic\_data.fare)

ax3.set\_xlabel("Edad")

ax3.set\_ylabel("Tarifa")

st.header("Grafica de Dispersión del Titanic")

st.pyplot(fig3)

Este archivo lo puede editar en cualquier editor de texto (notepad; sublime, etc) y deberá guardarlo con el nombre de mi\_app.py. Hay que recordar que este archivo se deberá colocar en la sección de archivos del Google Colab principal que se tiene que hacer para generar el url que nos permitirá visualizar la aplicación web.

En el ejemplo de este tema encontrarán el código inserto en el Google Colab para que lo puedan ejecutar.

Se cargará una pestaña en su navegador que este predeterminado y podrá visualizar la aplicación web que hemos creado (ver figura 20).

Chart, histogram

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, scatter chart

Description automatically generated

Figura 20. Vista de la aplicación web con Streamlit y Google Colab con la visualización de las gráficas.

* **SIGNIFICA**

Ahora que ya aprendiste como graficar en Streamlit para una aplicación web, como podrías construir un sitio web que conteste a las siguientes interrogantes:

* + - ¿Cuál sería el código para hacer una gráfica de barras que muestre las ventas de las categorías de WalMart USA?
    - ¿Cuál sería el código para hacer una gráfica de pastel que muestre las ventas por región de WaltMart USA?
    - ¿Cuál sería el código para hacer un histograma de los discuentos de los datos de WaltMart USAR?
* **ACTIVATE**
  + - Te invito a que revises el siguiente enlace en donde encontrarás las instrucciones que se requieren para contestar a las interrogantes planteadas anteriormente. <https://docs.streamlit.io/en/stable/api.html#display-charts>
* **PRACTICA TEMA 4** 
  + **Es momento de poner en práctica lo que aprendiste en este tema.**

**Ejemplo 1 Actívate:**

***Descripción breve:***

**Desarrollar un código sobre la estructura de una aplicación web que contenga 3 gráficas (barras, pastel y un histograma) sobre el proyecto de visualización de analítica de datos para WalMart USA.**

**En el siguiente enlace podrás encontrar el enlace al conjunto de datos que pertencen a WaltMart USA. <https://raw.githubusercontent.com/jeaggo/tc3068/master/Superstore.csv>**

***Instrucciones generales***

**El conjunto de datos de WalMart USA contiene los siguientes campos:**

**Row ID - Indice del registro de la orden**

**Order ID - Indice de la orden**

**Order Date - Fecha de la orden**

**Ship Date - Fecha de embarque**

**Ship Mode - Modo de embarque**

**Customer ID - Clave del cliente**

**Customer Name - Nombre del cliente**

**Segment - Segmento**

**Country - Pais**

**City - Ciudad**

**State - Estado**

**Postal Code - Código postal**

**Region - Region**

**Product ID - Clave del producto**

**Category - Categoría**

**Sub-Category - Sub categoría**

**Product Name - Nombre del producto**

**Sales - Ventas**

**Quantity - Cantidad vendida**

**Discount - Descuento aplicado**

**Profit - Ganancia**

***Ejercicio:***

**La estructura de la aplicación web deberá tener lo siguiente:**

* + **Deberá incluir un titulo**
  + **Deberá incluir un subtitulo (este puede ser opcional)**
  + **Deberá incluir un breve párrafo que describa el objetivo de este proyecto de analítica de datos; por ejemplo: predicción de ventas de productos de línea blanca en el noroeste de los Estados Unidos.**
  + **Una sección que contenga las gráficas antes mencionadas.**

1. **Cierre (Ideas para llevar)**

En este profundiza se aprendio acerca de los elementos principales con los que se puede construir una aplicación web sencilla que permita visualizar el análisis de datos de una organización o empresa. Hay que recordar que en una aplicación web usando el marco de referencia Streamlit se debe de tener los siguientes elementos:

1. Titulo; Subtitulo y una descripción del proyecto
2. Una barra lateral (sidebar)
3. Controles o widgets
4. Y el uso de gráficas como barras, gráficas de pastel o histogramas.
5. **Material adicional (se incluye por tema o al final del contenido)**

Se recomienda revisar el siguiente enlace en donde vienen una serie de tutoriales que ayudaran a profundizar sobre este marco de referencia: https://docs.streamlit.io/en/stable/

* **Aquí se debe de incluir el Colab Guia Nutre**

El primer código es el que sirve para llamar a Streamlit desde Google Colab. Este archivo lo puedes encontrar como zero.ipynb

Instalación de paquetes

!pip install streamlit

Vamos a cargar una máquina virtual que tenga instalado Ngrok

!wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip

Ahora cargaremos esta máquina virtual en nuestro archivo de Google Colab

!unzip ngrok-stable-linux-amd64.zip

Posteriormente le indicaremos a Ngrok que estará utilizando el puerto 8501 para visualizar nuestra aplicación

get\_ipython().system\_raw('./ngrok http 8501 &')

Una vez inicializado el puerto, ahora vamos a crear el tunel para que pueda generarse un enlace que nos permita visualizar nuestra aplicación

!curl -s http://localhost:4040/api/tunnels | python3 -c \

'import sys, json; print("Execute the next cell and the go to the following URL: " +json.load(sys.stdin)["tunnels"][0]["public\_url"])'

Una vez que ejecutaron la instrucción anterior, verán que aparece un URL que es el que vamos a dar clic para visualizar nuestra aplicación, pero primero tenemos que ejecutar nuestra aplicación

Tenemos que cargar primero la aplicación en nuestro archivo de Google Colab, para eso vamos a dar clic en el icono del lado derecho que tiene la imagen de un folder

Una vez cargado el archivo ahora vamos a proceder a ejecutarlo con el comando streamlit para que se visualice el contenido del mismo.

!streamlit run /content/mi\_app.py

El segundo código es el que se hace en un editor de texto y se guarda con la extensión .py y ese es el que se debe de cargar en el archivo de inicio que se tiene en Google Colab.

import pandas as pd

import matplotlib.pyplot as plt

import streamlit as st

titanic\_link = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.csv'

titanic\_data = pd.read\_csv(titanic\_link)

fig, ax = plt.subplots()

ax.hist(titanic\_data.fare)

st.header("Histograma del Titanic")

st.pyplot(fig)

st.markdown("\_\_\_")

fig2, ax2 = plt.subplots()

y\_pos = titanic\_data['class']

x\_pos = titanic\_data['fare']

ax2.barh(y\_pos, x\_pos)

ax2.set\_ylabel("Class")

ax2.set\_xlabel("Fare")

ax2.set\_title('¿Cuanto pagaron las clases del Titanic')

st.header("Grafica de Barras del Titanic")

st.pyplot(fig2)

st.markdown("\_\_\_")

fig3, ax3 = plt.subplots()

ax3.scatter(titanic\_data.age, titanic\_data.fare)

ax3.set\_xlabel("Edad")

ax3.set\_ylabel("Tarifa")

st.header("Grafica de Dispersión del Titanic")

st.pyplot(fig3)