ANÁLISIS DE DATOS CON STREAMLIT

Introducción:

Se describe una aplicación interactiva de análisis de datos desarrollada con Streamlit. La aplicación permite a los usuarios subir archivos de datos en formato CSV o Excel y realizar diversas operaciones de análisis y visualización de datos de manera sencilla y eficiente. Las principales funcionalidades de la aplicación incluyen:

- **Lectura de Datos**: Permite a los usuarios subir archivos de datos y muestra los datos cargados.
- **Resumen de Datos**: Proporciona un resumen estadístico básico de los datos, incluyendo medidas como media, mediana, varianza, y más.
- **Visualización de Datos**: Ofrece varias opciones para visualizar los datos, como histogramas, boxplots, diagramas de dispersión, ojivas (frecuencia acumulada) y gráficos de regresión lineal.
- **Análisis Estadístico**: Realiza análisis de regresión lineal y muestra los resultados, incluyendo el intercepto y el coeficiente, además de un resumen estadístico detallado.
- **Cálculos Estadísticos**: Calcula y muestra diversas estadísticas descriptivas para las columnas seleccionadas, como media, mediana, varianza, desviación estándar, y otros.

Librerías Utilizadas

La aplicación hace uso de varias librerías de Python para manejar datos, visualizarlos y realizar análisis estadísticos. A continuación se describen las principales librerías utilizadas y sus propósitos:

- **Streamlit**: Biblioteca para crear aplicaciones web interactivas. Permite desplegar la aplicación en la web y crear interfaces de usuario de forma sencilla.
- **Pandas**: Biblioteca para manipulación y análisis de datos. Es utilizada para leer, procesar y mostrar los datos cargados desde archivos CSV o Excel.
- **NumPy**: Biblioteca para operaciones matemáticas y manejo de matrices. Se usa en conjunto con Pandas para realizar cálculos numéricos y estadísticos.
- **Seaborn**: Biblioteca para visualización de datos basada en matplotlib. Facilita la creación de gráficos estadísticos atractivos y descriptivos, como histogramas, boxplots y diagramas de dispersión.
- **Matplotlib**: Biblioteca para crear gráficos en Python. Es la base sobre la cual Seaborn construye sus visualizaciones y se utiliza para personalizar y mostrar gráficos.
- **scikit-learn**: Biblioteca para aprendizaje automático. En este proyecto, se utiliza específicamente para realizar análisis de regresión lineal.
- **Statsmodels**: Biblioteca para análisis estadístico. Proporciona herramientas para estimar modelos estadísticos, realizar pruebas estadísticas y visualizar resultados, complementando las capacidades de scikit-learn en análisis de regresión.

Código Completo:

```
import streamlit as st
import pandas as pd
import numpy as np
import seaborn as sns
```

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm

def main():
    st.title("Análisis de Datos con Streamlit")

    uploaded_file = st.file_uploader("Sube un archivo CSV o Excel", type=["csv", "xlsx", "xls"])

    if uploaded_file is not None:
        try:
            if uploaded_file.name.endswith('.csv'):
                df = pd.read_csv(uploaded_file)
            else:
                df = pd.read_excel(uploaded_file)
            st.write("### Datos del archivo")
            st.write(df)

            st.write("### Resumen Estadístico")
            st.write(df.describe().transpose())

            st.write("### Visualización de Datos")

            chart_type = st.selectbox("Selecciona el tipo de gráfico", ["Histogram", "Boxplot", "D. de

            if chart_type == "Histogram":
                column = st.selectbox("Selecciona la columna para el histograma", df.columns)
                plt.figure(figsize=(10, 6))
                sns.histplot(df[column], kde=True)
                st.pyplot(plt)

            elif chart_type == "Boxplot":
                column = st.selectbox("Selecciona la columna para el boxplot", df.columns)
                plt.figure(figsize=(10, 6))
                sns.boxplot(y=df[column])
                st.pyplot(plt)

            elif chart_type == "D. de Dispersión":
                col1 = st.selectbox("Selecciona la columna para el eje X", df.columns)
                col2 = st.selectbox("Selecciona la columna para el eje Y", df.columns)
                plt.figure(figsize=(10, 6))
                sns.scatterplot(x=df[col1], y=df[col2])
                st.pyplot(plt)

            elif chart_type == "Ojiva":
                column = st.selectbox("Selecciona la columna para la ojiva", df.columns)
                plt.figure(figsize=(10, 6))
                sns.histplot(df[column], kde=False, cumulative=True, stat="density", bins=30)
                plt.xlabel(column)
                plt.ylabel("Frecuencia acumulada")
                plt.title("Ojiva")
                st.pyplot(plt)

            elif chart_type == "Regresión Lineal":
```

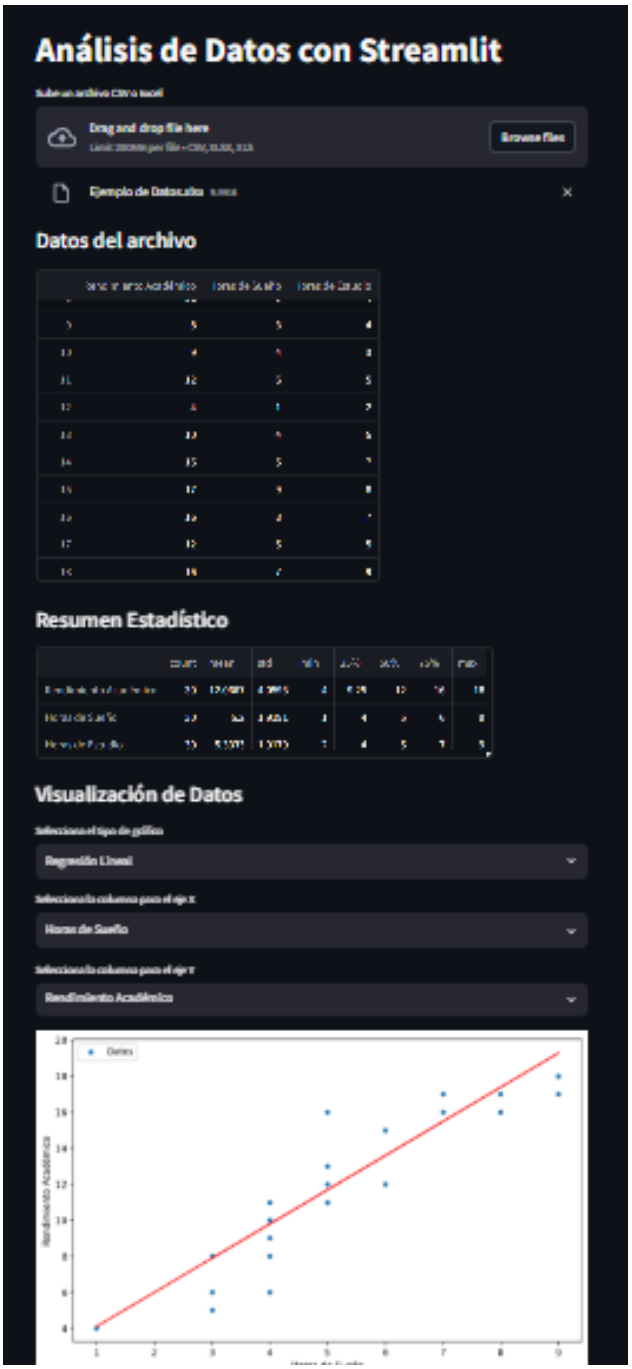
```
        col1 = st.selectbox("Selecciona la columna para el eje X", df.columns)
        col2 = st.selectbox("Selecciona la columna para el eje Y", df.columns)

        X = df[[col1]]
        Y = df[col2]
        model = LinearRegression()
        model.fit(X, Y)
        Y_pred = model.predict(X)

        plt.figure(figsize=(10, 6))
        sns.scatterplot(x=df[col1], y=df[col2], label='Datos')
        plt.plot(df[col1], Y_pred, color='red', label='Regresión Lineal')
        st.pyplot(plt)

        st.write("### Resultados de la Regresión Lineal")
        st.write(f"Intercepto: {model.intercept_}")
        st.write(f"Coeficiente: {model.coef_[0]}")

        X2 = sm.add_constant(X)
        est = sm.OLS(Y, X2)
        est2 = est.fit()
        st.write(est2.summary())

    st.write("### Cálculos Estadísticos")
    stats_col = st.selectbox("Selecciona la columna para cálculos estadísticos", df.columns)
    if df[stats_col].dtype in ['float64', 'int64']:
        mean = df[stats_col].mean()
        median = df[stats_col].median()
        var = df[stats_col].var()
        std = df[stats_col].std()
        min_val = df[stats_col].min()
        max_val = df[stats_col].max()
        range_val = max_val - min_val
        iqr = df[stats_col].quantile(0.75) - df[stats_col].quantile(0.25)
        skewness = df[stats_col].skew()
        kurtosis = df[stats_col].kurt()

        st.write(f"Media: {mean}")
        st.write(f"Mediana: {median}")
        st.write(f"Varianza: {var}")
        st.write(f"Desviación Estándar: {std}")
        st.write(f"Mínimo: {min_val}")
        st.write(f"Máximo: {max_val}")
        st.write(f"Rango: {range_val}")
        st.write(f"Rango Intercuartílico: {iqr}")
        st.write(f"Simetría: {skewness}")
        st.write(f"Curtosis: {kurtosis}")

        numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
        if len(numeric_cols) > 1:
            st.write("### Matriz de Correlación")
            corr_matrix = df[numeric_cols].corr()
            st.write(corr_matrix)
            plt.figure(figsize=(10, 6))
```

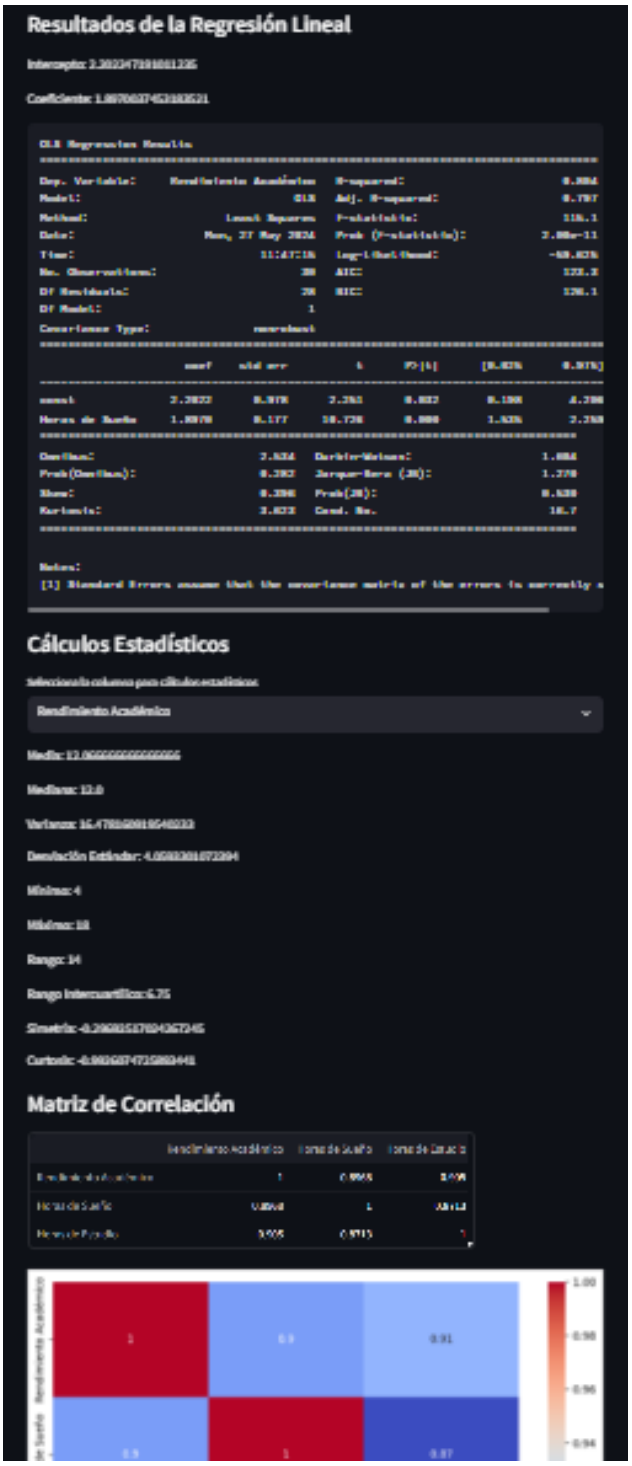
```
                    sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
                    st.pyplot(plt)
            except Exception as e:
                st.error(f"Error al leer el archivo: {e}")

if __name__ == "__main__":
    main()
```

Web Streamlit:

Link del Repositorio y Código QR:

https://github.com/Lucc4z/Programming-Language-II/blob/main/Python/stream1it.py


Análisis de Datos con Streamlit
Drag and drop file here
Browse files
Ejemplo de Datos.xlsx
Datos del archivo
Resumen Estadístico
Visualización de Datos
Regresión Lineal
Horas de Sueño
Rendimiento Académico
Datos


Resultados de la Regresión Lineal
Cálculos Estadísticos
Matriz de Correlación