!Pip install gradio

!pip install gradio pandas matplotlib

From mpl\_toolkits.mplot3d import Axes3D

From sklearn.preprocessing import StandardScaler

Import matplotlib.pyplot as plt # plotting

Import numpy as np # linear algebra

Import os # accessing directory structure

Import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

Df=pd.read\_csv(‘sample.csv’)

Df.head()

# Distribution graphs (histogram/bar graph) of column data

Def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):

    Nunique = df.nunique()

    Df = df[[col for col in df if nunique[col] > 1 and nunique[col] < 50]] # For displaying purposes, pick columns that have between 1 and 50 unique values

    nRow, nCol = df.shape

    columnNames = list(df)

    nGraphRow = (nCol + nGraphPerRow – 1) // nGraphPerRow

    plt.figure(num = None, figsize = (6 \* nGraphPerRow, 8 \* nGraphRow), dpi = 80, facecolor = ‘w’, edgecolor = ‘k’)

    for I in range(min(nCol, nGraphShown)):

        plt.subplot(nGraphRow, nGraphPerRow, I + 1)

        columnDf = df.iloc[:, i]

        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):

            valueCounts = columnDf.value\_counts()

            valueCounts.plot.bar()

        else:

            columnDf.hist()

        plt.ylabel(‘counts’)

        plt.xticks(rotation = 90)

        plt.title(f’{columnNames[i]} (column {i})’)

    plt.tight\_layout(pad = 1.0, w\_pad = 1.0, h\_pad = 1.0)

    plt.show()

# Correlation matrix

Def plotCorrelationMatrix(df, graphWidth):

    Filename = df.dataframeName

    Df = df.dropna(‘columns’) # drop columns with NaN

    Df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique values

    If df.shape[1] < 2:

        Print(f’No correlation plots shown: The number of non-NaN or constant columns ({df.shape[1]}) is less than 2’)

        Return

    Corr = df.corr()

    Plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, facecolor=’w’, edgecolor=’k’)

    corrMat = plt.matshow(corr, fignum = 1)

    plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)

    plt.yticks(range(len(corr.columns)), corr.columns)

    plt.gca().xaxis.tick\_bottom()

    plt.colorbar(corrMat)

    plt.title(f’Correlation Matrix for {filename}’, fontsize=15)

    plt.show()

    # Scatter and density plots

Def plotScatterMatrix(df, plotSize, textSize):

    Df = df.select\_dtypes(include =[np.number]) # keep only numerical columns

    # Remove rows and columns that would lead to df being singular

    Df = df.dropna(‘columns’)

    Df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique values

    columnNames = list(df)

    if len(columnNames) > 10: # reduce the number of columns for matrix inversion of kernel density plots

        columnNames = columnNames[:10]

    df = df[columnNames]

    ax = pd.plotting.scatter\_matrix(df, alpha=0.75, figsize=[plotSize, plotSize], diagonal=’kde’)

    corrs = df.corr().values

    for I, j in zip(\*plt.np.triu\_indices\_from(ax, k = 1)):

        ax[I, j].annotate(‘Corr. Coef = %.3f’ % corrs[I, j], (0.8, 0.2), xycoords=’axes fraction’, ha=’center’, va=’center’, size=textSize)

    plt.suptitle(‘Scatter and Density Plot’)

    plt.show()

nRowsRead = 1000 # specify ‘None’ if want to read whole file

df1 = pd.read\_csv(‘sample.csv’, delimiter=’,’, nrows = nRowsRead)

df1.dataframeName = ‘sample.csv’

nRow, nCol = df1.shape

print(f’There are {nRow} rows and {nCol} columns’)

df1.head(5)

plotPerColumnDistribution(df1, 10, 5)

#dly

Import gradio as gr

Import pandas as pd

Import matplotlib.pyplot as plt

Import numpy as np

# ----------- Plot Functions -----------

Def plotPerColumnDistribution(df, nGraphShown=10, nGraphPerRow=5):

    Nunique = df.nunique()

    Df = df[[col for col in df if nunique[col] > 1 and nunique[col] < 50]]

    nRow, nCol = df.shape

    columnNames = list(df)

    nGraphRow = (nCol + nGraphPerRow – 1) // nGraphPerRow

    fig, axs = plt.subplots(nGraphRow, nGraphPerRow, figsize=(6 \* nGraphPerRow, 5 \* nGraphRow))

    axs = axs.flatten()

    for I in range(min(nCol, nGraphShown)):

        columnDf = df.iloc[:, i]

        ax = axs[i]

        if not np.issubdtype(type(columnDf.iloc[0]), np.number):

            columnDf.value\_counts().plot.bar(ax=ax)

        else:

            columnDf.hist(ax=ax)

        ax.set\_title(f’{columnNames[i]}’)

        ax.tick\_params(axis=’x’, rotation=90)

    for j in range(i+1, len(axs)):

        fig.delaxes(axs[j])

    plt.tight\_layout()

    return fig

def plotCorrelationMatrix(df):

    df = df.dropna(axis=1)

    df = df.loc[:, df.nunique() > 1]

    if df.shape[1] < 2:

        raise ValueError(“Not enough numeric columns for correlation matrix.”)

    corr = df.corr()

    fig, ax = plt.subplots(figsize=(10, 8))

    cax = ax.matshow(corr, cmap=’coolwarm’)

    plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)

    plt.yticks(range(len(corr.columns)), corr.columns)

    plt.colorbar(cax)

    plt.title(“Correlation Matrix”, y=1.15)

    return fig

def plotScatterMatrix(df):

    df = df.select\_dtypes(include=[np.number])

    df = df.dropna(axis=1)

    df = df.loc[:, df.nunique() > 1]

    columnNames = list(df)

    if len(columnNames) > 6:

        columnNames = columnNames[:6]

    df = df[columnNames]

    fig = pd.plotting.scatter\_matrix(df, figsize=(12, 12), diagonal=’kde’, alpha=0.7)

    plt.suptitle(“Scatter Matrix”, y=1.02)

    return plt.gcf()

# ----------- Gradio Interface Function -----------

Def analyze\_file(file, plot\_type):

    Df = pd.read\_csv(file)

    If plot\_type == “Column Distribution”:

        Fig = plotPerColumnDistribution(df)

    Elif plot\_type == “Correlation Matrix”:

        Fig = plotCorrelationMatrix(df)

    Elif plot\_type == “Scatter Matrix”:

        Fig = plotScatterMatrix(df)

    Else:

        Raise ValueError(“Invalid plot type selected.”)

    Return fig

# ----------- Gradio Setup -----------

Plot\_options = [“Column Distribution”, “Correlation Matrix”, “Scatter Matrix”]

Gr.Interface(

    Fn=analyze\_file,

    Inputs=[

        Gr.File(label=”Upload CSV File”),

        Gr.Radio(choices=plot\_options, label=”Select Plot Type”)

    ],

    Outputs=gr.Plot(label=”Output Plot”),

    Title=”Interactive EDA Tool”,

    Description=”Upload a CSV file and choose the type of data visualization you’d like to generate.”

).launch()