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)*

print(os.listdir('../input'))

*# 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

# Tweets.csv has 14640 rows in reality, but we are only loading/previewing the first 1000 rows

df1 = pd.read\_csv('../input/Tweets.csv', delimiter=',', nrows = nRowsRead)

df1.dataframeName = 'Tweets.csv'

nRow, nCol = df1.shape

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