#Steve Austin

#<Module 6 Assignment>

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#\*\*\*\*\*\*\*\*\* PREPROCESSING TASKS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Standard imports for preprocessing

import pandas as pd

import numpy as np

from sklearn import preprocessing as pre

#----------------------------------------------------------------------------

# READ IN DATA SOURCES

df = pd.read\_excel('C:/Users/saust/OneDrive/Desktop/M5.xlsx', sheet\_name='M5')

#----------------------------------------------------------------------------

# CHARACTERIZE THE DATA

print('Basic info about the dataframe and its rows and columns:')

print(df.info())

print('Summary statistics for all numerical columns:')

print(df.describe())

print("Number of null/missing values in each column:")

print(df.isna().sum())

print("Number of unique values for all columns:")

print(df.nunique(axis=0))

print("Duplicate rows:")

print(df[df.duplicated(keep=False)])

df.hist()

print("Total number of null/missing values")

df.isna().sum().sum()

#----------------------------------------------------------------------------

# REMOVE ROWS WITH >1 MISSING VALUES

df = df[df.isnull().sum(axis=1) < 2]

#NUMBER OF NULLS AFTER CULL

print("Total number of null/missing values")

print(df.isna().sum().sum())

print(df.isna().sum())

#----------------------------------------------------------------------------

# UPDATE ROWS WITH MISSING VALUES AS MEAN OF COLUMN

AMe=round(df['Age'].mean())

print(AMe)

WEMe=round(df['Work\_Experience'].mean())

print(WEMe)

FSMe=round(df['Family\_Size'].mean())

print(FSMe)

replaceNansMean = {'Age':AMe,'Work\_Experience':WEMe,'Family\_Size':FSMe}

df.fillna(value=replaceNansMean, inplace=True)

#NUMBER OF NULLS AFTER UPDATE

print("Total number of null/missing values")

print(df.isna().sum().sum())

print(df.isna().sum())

#----------------------------------------------------------------------------

# UPDATE ROWS WITH MISSING VALUES AS MODE OF COLUMN

df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])

df['Ever\_Married'] = df['Ever\_Married'].fillna(df['Ever\_Married'].mode()[0])

df['Graduated'] = df['Graduated'].fillna(df['Graduated'].mode()[0])

df['Profession'] = df['Profession'].fillna(df['Profession'].mode()[0])

df['Spending\_Score'] = df['Spending\_Score'].fillna(df['Spending\_Score'].mode()[0])

#NUMBER OF NULLS AFTER UPDATE

print("Total number of null/missing values")

print(df.isna().sum().sum())

print(df.isna().sum())

#----------------------------------------------------------------------------

#TRANSFORM CATEGORIES TO BINARY

print("Binary Conversions")

from sklearn.preprocessing import LabelEncoder

df['Gender'] = LabelEncoder().fit\_transform(df['Gender'])

print("Female=0, Male=1")

df['Ever\_Married'] = LabelEncoder().fit\_transform(df['Ever\_Married'])

print("No=0, Yes=1")

df['Graduated'] = LabelEncoder().fit\_transform(df['Graduated'])

print("No=0, Yes=1")

#----------------------------------------------------------------------------

#TRANSFORM ORDINAL TO NUMERICAL

print("Ordinal Conversions")

scale\_mapper = {"Low":1, "Average":2, "High":3}

df["Spending\_Score"] = df["Spending\_Score"].replace(scale\_mapper)

print(scale\_mapper)

#----------------------------------------------------------------------------

#ENCODE PROFESSION FIELD

print("Original Columns")

list(df.columns)

df=pd.concat([df,pd.get\_dummies(df[['Profession']])],axis=1)

df=df.drop(columns=['Profession'])

print("Columns after 'Profession' Encoded")

list(df.columns)

#----------------------------------------------------------------------------

#Scale all numeric columns to Z-zcore normalization

dfNoCLASS = df.loc[:, df.columns != 'CLASS']

col\_names =  dfNoCLASS.columns

dfNoCLASS[col\_names] = pd.DataFrame(pre.StandardScaler().fit\_transform(pd.DataFrame(dfNoCLASS[col\_names])))

dfCLASS = df['CLASS']

df = dfNoCLASS.join(dfCLASS)

col\_names\_scaled =  df

df[col\_names\_scaled] = pd.DataFrame(pre.StandardScaler().fit\_transform(pd.DataFrame(df[col\_names\_scaled])))

import seaborn as sns

sns.boxplot(df['Age'])

sns.boxplot(df['Work\_Experience'])

sns.boxplot(df['Family\_Size'])

import pandas as pd

import matplotlib.pyplot as plt

from matplotlib import rcParams

rcParams['figure.figsize'] = 14, 7

rcParams['axes.spines.top'] = False

rcParams['axes.spines.right'] = False

#Can use correlation matrix to look at relations of variables

#create correlation matrix

corrM = df.corr()

#(note encoded variables will always have relatively high negative correlation)

#lets look at how strongly features are correlated with output class variable

out\_class=corrM[["Health index"]]

out\_class=out\_class.apply(abs)

out\_class.sort\_values(by="Health index",inplace=True, ascending=False)

#(it looks like none of the features are highly correlated with the class variable)

#lets look at correlation of "DIST" feature with other features

#(repeat following for each feature)

f\_dist=corrM[["Dist"]]

f\_dist=f\_dist.apply(abs)

f\_dist.sort\_values(by="Dist",inplace=True, ascending=False)

#Drop feature that have < 0.01 correlation

df=df.drop(columns=['Profession\_Marketing','Gender','Graduated','Family\_Size','Profession\_Doctor'])

print("Drop Features that have < 0.01 correlation using correlation matrix")

print("Features Droped from df are: 'Profession\_Marketing','Gender','Graduated','Family\_Size','Profession\_Doctor' ")