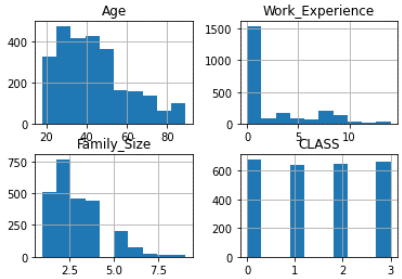
Module 6 Assignment - SOLUTIONS

This assignment is now an optional 50pt bonus (extra credit) assignment. It must be submitted by the assignment deadline.

1. Using the data in "M5.csv" (should have been M6.csv), write the code to perform the following characterization & preprocessing using Python. Do not include the CLASS column in any of your preprocessing as this is the output column. For your solution, provide the script code or Jupyter notebook and any documentation asked for in each subpart (this can be manually provided or output as part of your script)
    a. Characterize each input feature. What data type is it (string: categorical, ordinal, factor, Boolean; numeric: binary, integer, real), how many missing, NaN, or na values, number of distinct values, histogram or similar graphic showing distribution of values.

```
import pandas as pd
import os
os.chdir('c:/users/gknapp/desktop')  #change path to location of data files on your pc
df =  pd.read_csv('M5.csv')
print('Column types:')
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()
#from above and viewing the df dataframe after loading (2627 rows and 9 columns):
   #Gender: string, categorical but basically binary; 2 unique values; 0 null values
   #Ever_Married: string (object), categorical but basically binary; 2 unique values; 50 null values
   #Age: integer (imported as float); 67 unique values; 1 null values
   #Graduated: string, categorical but basically binary; 2 unique values; 26 null values
   #Profession: string, categorical; 9 unique values; 38 null values
   #Work_Experience: integer (imported as float); 15 unique values; 270 null values
   #Spending_Score: string, ordinal; 3 unique values; 0 null values
   #Family_Size: integer (imported as float); 9 unique values; 113 null values
   #CLASS: integer; 4 unique values; 0 null values
#See plots for output of .hist()
```



Age, Work_Experience, and Family_Size all show some skew to the right

    b. If a row has 2 or more missing/NaN, or na values, remove it from the dataset. Document the number of rows removed.

```
df=df.drop(df[df.isnull().sum(axis=1)>=2].index)
#44 rows removed
```

c. From the remaining data, replace all missing, NaN, or na values with for numeric values, the average value for that feature; and for string type, replace with the most frequent value in the column. Document the number of values replaced for each feature.

```
#Get count of remaining na by column
df.isna().sum()
#replace age, work_experience, family_size na with column mean
df.Age.fillna(value=df.Age.mean(), inplace=True)
df.Work_Experience.fillna(value=df.Work_Experience.mean(), inplace=True)
df.Family_Size.fillna(value=df.Family_Size.mean(), inplace=True)
#for categorical, first get frequency of values. Could just
#review data, or can use .value_counts() method.
#Most frequent: For ever married, "Yes"; for Graduated: "Yes" ; for Profession: "Artist"
df.Ever_Married.value_counts()
df.Graduated.value_counts()
df.Profession.value_counts()
df.Ever_Married.fillna(value='Yes', inplace=True)
df.Graduated.fillna(value='Yes', inplace=True)
df.Profession.fillna(value='Artist', inplace=True)
#from earlier isna sum: 0 age, 40 ever_married, 18 graduated,
#29 profession, 231 work experience, 88 family size
```

d. Convert string-based Boolean features ("YES"/"NO", "T"/"F"/, etc.) to numeric binary features (0/1). Document what features are converted.

```
df.Gender=df.Gender.replace({"Female":1,"Male":0}).astype(int)
df.Ever_Married=df.Ever_Married.replace({"Yes":1,"No":0}).astype(int)
df.Graduated=df.Graduated.replace({"Yes":1,"No":0}).astype(int)
```

e. Convert any ordinal features into numeric features; make sure to retain the ordering in the numbering assigned. Document what features are converted.

```
df.Spending_Score=df.Spending_Score.replace({"Low":1,"Average":2,"High":3}).astype(int)
```

f. Dummy code any remaining categorical data fields; remove the original feature columns once completed. Document what variables are added/removed in the process.

```
df=pd.concat([pd.get_dummies(df[['Profession']]),df],axis=1)
df=df.drop(columns=['Profession'])
```

g. Normalize all numeric features using Z-score normalization.

```
from sklearn.preprocessing import StandardScaler
#numeric columns to scale (don't generally include output class or binary variables):
col_names = ['Age','Work_Experience','Spending_Score','Family_Size']
df[col_names] = pd.DataFrame(StandardScaler().fit_transform(pd.DataFrame(df[col_names])))
```

h. For numeric features, check for outliers (use +/- 4 Z for cutoff); document but do not remove.

```
#Recall StandardScaler sets values to number of Zs from column mean,
#so just need to check the column values directly.
import numpy as np
#execute the following statements individually to see results
#for each feature
df[np.abs(df.Age) >=4]
df[np.abs(df.Work_Experience) >=4]
df[np.abs(df.Spending_Score) >=4]
df[np.abs(df.Family_Size) >=4]
#Only family_size has any +/-4 Zs from mean (16 rows)
```

i. Use a filter-based feature selection method to determine which features to keep and drop. Document the method used and results.

I should have modified this question as we did not cover feature selection as much as was planned for last fall. You could just use one of the methods from Module 6 tutorial script

```
from xgboost import XGBClassifier
model = XGBClassifier()
X = df.drop(columns="CLASS")
Y = df.CLASS
model.fit(X,Y)
#Extract importance for each feature (Attribute) from trained model
importances = pd.DataFrame(data={
   'Attribute': X.columns,
   'Importance': model.feature_importances_
})
print(importances)
```