Automating Bivariate Statistics Homework

The function we created in the chapter to calculate bivariate statistics based on the data types of the feature and label is copied below:

```
def unistats(df):
  import pandas as pd

  output_df = pd.DataFrame(columns=['Count', 'Unique', 'Type', 'Min',
'Max', '25%', '50%', '75%', 'Mean', 'Median', 'Mode', 'Std', 'Skew',
'Kurt'])

  for col in df.columns:
    # these are the outputs that apply to every variable regardless of
data type
    count = df[col].count()
    unique = df[col].nunique()
    dtype = str(df[col].dtype)

    if pd.api.types.is_numeric_dtype(dtype):
      # perform additional calculations for numeric variables
      min = round(df[col].min(), 2)
      max = round(df[col].max(), 2)
      quar_1 = round(df[col].quantile(.25), 2)
      quar_2 = round(df[col].quantile(.50), 2)
      quar_3 = round(df[col].quantile(.75), 2)
      mean = round(df[col].mean(), 2)
      median = round(df[col].median(), 2)
      mode = round(df[col].mode().values[0], 2)  # Use the .values[0]
to prevent the return of an extra printed datatype
      std = round(df[col].std(), 2)
      skew = round(df[col].skew(), 2)
      kurt = round(df[col].kurt(), 2)

      output_df.loc[col] = (count, unique, dtype, min, max, quar_1,
quar_2, quar_3, mean, median, mode, std, skew, kurt)
    else:
      output_df.loc[col] = (count, unique, dtype, '-', '-', '-', '-',
'-', '-', '-', '-', '-', '-', '-', '-')

  return output_df

def bivariate_stats(df, label, roundto=4):
  import pandas as pd
  from scipy import stats

  output_df = pd.DataFrame(columns=['missing', 'p', 'r', 'y = m(x) +
b', 'F', 'X2'])
```

```
  for feature in df.columns:
    if feature != label:
      df_temp = df[[feature, label]]
      df_temp = df_temp.dropna()
      missing = (df.shape[0] - df_temp.shape[0]) / df.shape[0]

      if pd.api.types.is_numeric_dtype(df_temp[feature]) and
pd.api.types.is_numeric_dtype(df_temp[label]):
        m, b, r, p, err = stats.linregress(df_temp[feature],
df_temp[label])
        output_df.loc[feature] = [f'{missing:.2%}', round(p, roundto),
round(r, roundto), f'y = {round(m, roundto)}(x) + {round(b,
roundto)}', '-', '-']

      elif not pd.api.types.is_numeric_dtype(df_temp[feature]) and not
pd.api.types.is_numeric_dtype(df_temp[label]):
        contingency_table = pd.crosstab(df_temp[feature],
df_temp[label]) # Calculate the crosstab
        X2, p, dof, expected =
stats.chi2_contingency(contingency_table)  # Calculate the Chi-square
based on the crosstab
        output_df.loc[feature] = [f'{missing:.2%}', round(p, roundto),
'-', '-', '-', round(X2, roundto)]

      else:
        if pd.api.types.is_numeric_dtype(df_temp[feature]):
          num = feature
          cat = label
        else:
          num = label
          cat = feature

        groups = df_temp[cat].unique()
        group_lists = []
        for g in groups:
          g_list = df_temp[df_temp[cat] == g][num]
          group_lists.append(g_list)

        results = stats.f_oneway(*group_lists)
        F = results[0]
        p = results[1]
        output_df.loc[feature] = [f'{missing:.2%}', round(p, roundto),
'-', '-', round(F, roundto), '-']
  return output_df.sort_values(by=['p'])
```

Question 1:

Import the university student mental health dataset provided with this assessment. Pass the dataset into the univariate_stats() function copied from the chapter into the *Automating_Bivariate_Statistics.ipynb* included with this assessment. Print the results.

```
# Question 1:
```

Question 2:

Notice that many of the features are object or int data types even though they only have two unique values. Let's examine the actual data to determine whether we can easily cast these features into Boolean data types (e.g., if they are 0/1 values).

Print out the first five records of the dataset. This is technically not necessary to answer the question below, but we should always directly examing the data for any surprises.

```
# Question 2:
```

Question 3:

One of the primary purposes of exploring the data is to assess it's cleanliness and shape which determine the types of predictive and prescriptive analytics we can perform. The current version of unistats() does not tell us about the missing data that might cause problems later.

Modify the unistats function to also include the number of missing values and the percent of missing values for each column. Add the logic for this code in the most appropriate place in the flow of logic. Include these two new columns after count and before unique.

IMPORTANT: we don't want to overwrite the prior version of unistats because your answers in Question 1 and 2 depend on it. So you should copy the function into this question code block and modify this copied version.

```
# Question 3
```

Question 4:

Normally, we would also create univariate charts (histograms and count bar charts). But we will skip that step to keep this assignment as short as possible.

Next, recall that the question we are interested in answering with this dataset is, "What causes university students to seek mental health treatment?" There is a feature called SpecialistTreatment that represents whether each case/row/student has sought treatment; 0 = False, 1 = True. We will use this feature as our label.

Call the bivariate_stats function using this dataset and label and print out the results.

```
# Question 4:
```

Question 5:

Take a closer look at the results for *HasMentalHealthSupport* from the prior question and from Question 1. Notice that there is a perfection correlation (r = 1.0) between *HasMentalHealthSupport* and *SpecialistTreatment*. They also have the same mean (0.07) and

other univariate properties. This tells me that the student survey respondents likely interpreted having mental health support the same as seeking treatment. In other words, the data in these two features is identical. Therefore, we should not use HasMetnalHealthSupport as a predictor or indicator of SeekingTreatment because they are essentially the same thing.

Instead, let's focus on understanding the other two features that had p-values close to zero: Course and YearOfStudy. We already know that these features highly related to *SpecialistTreatment*. But to understand how these features determine *SpecialistTreatment*, we need to generate bar charts of their relationships with *SpecialistTreatment*. We already did this in the chapter. So you are welcome to copy any code you want from the chapter or generate your own new code.

Create a bivariate chart for at least *Course* and *YearOfStudy* with the *SpecialistTreatment*. However, if you want to use the automation code from the chapter, you can create bivariate charts for every feature with *SpecialistTreatment*.

NOTES:

- Before you plot all course's mean *SpecialistTreatment*, first group courses based on the 5% rule as we did in the chapter. You should end up with only four courses: Engineering, BIT, BCS, and Other
- It may be hard to see if you copied the code from the book because there are so many courses and the text runs together. Therefore, add code to rotate the x-axis labels 90 degrees so the text is easier to read.