COURSE ASSIGNMENT

SOFTWARE ENGINEERING DEPARTMENT

INTRODUCTION TO DATASCIENCE

ASSIGNMENT # 01

SUBMITTED BY

MUSHARRAF RAZA KHAN | 52024

BS-SE 8th

INSTRUCTOR

MA'AM SANIYA ASHRAF

Lecturer, Software Engineering Department,

Faculty of Information & Communication Technology

BUITEMS Quetta,

"Session Spring-2023"

“IDS-ASSIGNMENT BY-MUSHARRAF RAZA KHAN | 52024”

Question No 01:

Write a function called `proportion_of_education` which returns the proportion of children in the dataset who had a mother with the education levels equal to less than high school (<12), high school (12), more than high school but not a college graduate (>12) and college degree.

This function should return a dictionary in the form of (use the correct numbers, do not round numbers):

```
{
"less than high school":0.2,
"high school":0.4,
"more than high school but not college":0.2,
"college":0.2
}
```

CODE:

```
import pandas as pd
def proportion_of_education():
    # Load the dataset
    df = pd.read_csv("NISPUF17.csv", index_col=0)

    # Count the number of children for each education level
    education_counts = df["EDUC1"].value_counts()

    # Calculate the proportion of children for each education level
    total_children = education_counts.sum()
    proportions = {}
    for education_level in education_counts.index:
        count = education_counts[education_level]
        proportion = count / total_children
        if education_level == 1:
            proportions["less than high school"] = proportion
        elif education_level == 2:
            proportions["high school"] = proportion
        elif education_level == 3:
            proportions["more than high school but not college"] = proportion
        elif education_level == 4:
            proportions["college"] = proportion

    return proportions
```

This function first loads the dataset using the **pandas library**. It then counts the number of children for each education level and calculates the proportion of children for each level by dividing the count by the total number of children. Finally, it stores the proportions in a dictionary and returns it.

Note: In the dataset, education levels are represented by integer codes, where 1 represents "less than high school", 2 represents "high school", 3 represents "more than high school but not college", and 4 represents "college". The function uses these codes to create the dictionary keys.

EXPLAINATION:

The purpose of the function is to return a dictionary that shows the proportion of children in the dataset who had a mother with each education level. The education levels are defined as "less than high school" **(level 1)**, "high school" **(level 2)**, "more than high school but not college" **(level 3)**, and "college" **(level 4)**.

The function first imports the pandas library, which is a powerful library for data manipulation and analysis in Python. The pd.read_csv function is used to load the dataset from a CSV file named "NISPUF17.csv". The index_col=0 argument specifies that the first column of the dataset should be used as the index.

```
import pandas as pd
def proportion_of_education():

 # Load the dataset
    df = pd.read_csv("NISPUF17.csv", index_col=0)
```

Next, the function creates a Series object called education_counts that counts the number of children for each education level. The value_counts method of the Series object is used to compute the counts.

```
# Count the number of children for each education level
education_counts = df["EDUC1"].value_counts()
```

The total_children variable is then set to the total number of children in the dataset. This is computed by taking the sum of the counts in the education_counts series.

```
# Calculate the proportion of children for each education level
total_children = education_counts.sum()
```

The proportions dictionary is then created, and a loop is used to calculate the proportion of children for each education level. The loop iterates over the index values of the education_counts series, which correspond to the education level codes (1, 2, 3, and 4). For each level, the count is divided by the total number of children to get the proportion. The proportion is then stored in the proportions dictionary with the corresponding education level as the key.

```
proportions = {}
for education_level in education_counts.index:
    count = education_counts[education_level]
    proportion = count / total_children
    if education_level == 1:
        proportions["less than high school"] = proportion
    elif education_level == 2:
        proportions["high school"] = proportion
    elif education_level == 3:
        proportions["more than high school but not college"] = proportion
    elif education_level == 4:
        proportions["college"] = proportion
```

Finally, the proportions dictionary is returned as the output of the function.

```
return proportions
```

In Summary, the proportion_of_education function loads the dataset, counts the number of children for each education level, calculates the proportion of children for each level, and returns a dictionary that shows the proportions for each education level.

Question No 02:

Lets explore the relationship between being fed breastmilk as a child and getting a seasonal influenza vaccine from a healthcare provider. Return a tuple of the average number of influenza vaccines for those children we know received breastmilk as a child and those who know did not.

This function should return a tuple in the form (use the correct numbers:

```
(2.5, 0.1)
```

CODE

```
import pandas as pd


def breastfeeding_vaccine():
    # Load the dataset
    df = pd.read_csv("NISPUF17.csv", index_col=0)


    # Filter the dataset to include only children with valid values for breastfed and influenza vaccination
    filtered_df = df[(df["CBF_01"] == 1) | (df["CBF_01"] == 2)]
    filtered_df = filtered_df[(filtered_df["P_NUMFLU"] >= 0)]


    # Calculate the average number of influenza vaccines for children who were breastfed and those who were not
    breastfed_avg = filtered_df[filtered_df["CBF_01"] == 1]["P_NUMFLU"].mean()
    not_breastfed_avg = filtered_df[filtered_df["CBF_01"] == 2]["P_NUMFLU"].mean()


    return (breastfed_avg, not_breastfed_avg)
```

The function first loads the **NISPUF17.csv** dataset and filters it to include only the columns that we need for our analysis. The CBF_01 column indicates whether a child was fed breastmilk as a child, with a value of 1 indicating "yes" and a value of 2 indicating "no". The P_NUMFLU column indicates the number of seasonal influenza vaccines the child received from a healthcare provider.

Next, the function filters the dataset to include only those children with valid values for breastfed and influenza vaccination. We exclude children with missing values for either variable.

Finally, the function calculates the average number of influenza vaccines for children who were breastfed and those who were not. The mean method of the pandas.Series object is used to compute the means.

The function returns a tuple of the two means. The first element of the tuple is the average number of influenza vaccines for children who were breastfed, and the second element is the average number of influenza vaccines for children who were not breastfed.

EXPLAINATION:

The code is an implementation of a function called **breastfeeding_vaccine()** which explores the relationship between being fed breastmilk as a child and getting a seasonal influenza vaccine from a healthcare provider.

The function uses the NISPUF17.csv dataset which contains data about a national sample of children and their families in the United States. The dataset has many variables that describe the characteristics of the children, their parents, and their families.

The **pandas** library is imported to handle the dataset. The **read_csv()** function of the pandas library is used to read the **NISPUF17.csv** dataset and load it into a pandas.DataFrame object called **df.** The index_col parameter is set to 0 to use the first column of the dataset as the index column of the DataFrame.

```
# Filter the dataset to include only children with valid values for breastfed and influenza vaccination
    filtered_df = df[(df["CBF_01"] == 1) | (df["CBF_01"] == 2)]
    filtered_df = filtered_df[(filtered_df["P_NUMFLU"] >= 0)]
```

The function filters the DataFrame object to include only the rows with valid values for the CBF_01 and P_NUMFLU columns. The CBF_01 column indicates whether a child was fed breastmilk as a child, with a value of 1 indicating "yes" and a value of 2 indicating "no". The P_NUMFLU column indicates the number of seasonal influenza vaccines the child received from a healthcare provider.

The first line of code above filters the DataFrame to include only rows where CBF_01 is either 1 or 2. The second line of code filters the resulting DataFrame to include only rows where P_NUMFLU is greater than or equal to 0.

```
   # Calculate the average number of influenza vaccines for children who were breastfed and those who were not
```

breastfed_avg = filtered_df[filtered_df["CBF_01"] == 1]["P_NUMFLU"].mean()

not_breastfed_avg = filtered_df[filtered_df["CBF_01"] == 2]["P_NUMFLU"].mean()

The function then calculates the average number of influenza vaccines for children who were breastfed and those who were not. The **mean()** method of the pandas.Series object is used to compute the means.

The first line of code above calculates the mean of the P_NUMFLU column for the rows where CBF_01 is equal to 1 (i.e. children who were breastfed). The second line of code calculates the mean of the P_NUMFLU column for the rows where CBF_01 is equal to 2 (i.e. children who were not breastfed).

return (breastfed_avg, not_breastfed_avg)

Finally, the function returns a tuple of the two means. The first element of the tuple is the average number of influenza vaccines for children who were breastfed, and the second element is the average number of influenza vaccines for children who were not breastfed.

Question No 03:

It would be interesting to see if there is any evidence of a link between vaccine effectiveness and sex of the child. Calculate the ratio of the number of children who contracted chickenpox but were vaccinated against it (at least one varicella dose) versus those who were vaccinated but did not contract chicken pox. Return results by sex.

This function should return a dictionary in the form of (use the correct numbers):

```
{
"male":0.2,
"female":0.4
}
```

Note: To aid in verification, the `**chickenpox_by_sex()**['female']` value the autograder is looking for starts with the digits `0.0077`.

CODE:

```
import pandas as pd
def chickenpox_by_sex():
    # Load the dataset
    df = pd.read_csv('NISPUF17.csv', index_col=0)


    # Filter the dataset to only include children who were vaccinated against chickenpox
    df = df[df['P_NUMVRC'] >= 1]


    # Create a dictionary to store the results by sex
    results = {}


    # Calculate the ratio for males
    male_vaccinated_cpox = df[(df['P_NUMVRC'] >= 1) & (df['SEX'] == 1) &
(df['HAD_CPOX'] == 1)]
    male_vaccinated_no_cpox = df[(df['P_NUMVRC'] >= 1) & (df['SEX'] == 1) &
(df['HAD_CPOX'] == 2)]
    male_ratio = len(male_vaccinated_cpox) / len(male_vaccinated_no_cpox)
    results['male'] = round(male_ratio, 4)


    # Calculate the ratio for females
    female_vaccinated_cpox = df[(df['P_NUMVRC'] >= 1) & (df['SEX'] == 2) &
(df['HAD_CPOX'] == 1)]
    female_vaccinated_no_cpox = df[(df['P_NUMVRC'] >= 1) & (df['SEX'] == 2) &
(df['HAD_CPOX'] == 2)]
    female_ratio = len(female_vaccinated_cpox) / len(female_vaccinated_no_cpox)
    results['female'] = round(female_ratio, 4)


    return results
```

Here's what this function does:

1. Loads the dataset using Pandas
2. Filters the dataset to only include children who were vaccinated against chickenpox
3. Creates an empty dictionary to store the results by sex
4. Calculates the ratio of the number of vaccinated children who contracted chickenpox versus the number who did not for males and stores it in the dictionary
5. Calculates the ratio of the number of vaccinated children who contracted chickenpox versus the number who did not for females and stores it in the dictionary
6. Returns the results dictionary

EXPLAINATION:

Note that we round the ratios to four decimal places using the **round()** function. This is done to match the expected output format.

The code defines a function **called chickenpox_by_sex**() that calculates the ratio of the number of children who were vaccinated against chickenpox but contracted it versus those who were vaccinated but did not contract it, broken down by sex.

The first line of the function imports the Pandas library, which is used to read and manipulate datasets in Python.

The second line reads in the dataset from a CSV file using **the read_csv()** method of Pandas. The index_col=0 argument specifies that the first column of the dataset should be used as the index of the resulting DataFrame.

The third line filters the dataset to only include children who were vaccinated against chickenpox. The df['P_NUMVRC'] >= 1 condition checks whether the P_NUMVRC column of the dataset (which specifies the number of varicella (chickenpox) vaccine doses a child has received) is greater than or equal to 1. This filters out children who have not been vaccinated against chickenpox.

The fourth line creates an empty dictionary called results, which will be used to store the ratios calculated for each sex.

The next few lines of code calculate the ratio of the number of vaccinated children who contracted chickenpox versus the number who did not for males and females, respectively.

For Example, the following code calculates the ratio for males:

```
male_vaccinated_cpox = df[(df['P_NUMVRC'] >= 1) & (df['SEX'] == 1) &
(df['HAD_CPOX'] == 1)]

male_vaccinated_no_cpox = df[(df['P_NUMVRC'] >= 1) & (df['SEX'] == 1) &
(df['HAD_CPOX'] == 2)]

male_ratio = len(male_vaccinated_cpox) / len(male_vaccinated_no_cpox)

results['male'] = round(male_ratio, 4)
```

The first line of this code creates a new DataFrame called male_vaccinated_cpox that includes only the rows of the original dataset where:

- P_NUMVRC is greater than or equal to 1 (i.e., the child was vaccinated against chickenpox)
- SEX is equal to 1 (i.e., the child is male)
- HAD_CPOX is equal to 1 (i.e., the child contracted chickenpox)

The second line of code creates a new DataFrame called male_vaccinated_no_cpox that includes only the rows of the original dataset where:

- P_NUMVRC is greater than or equal to 1 (i.e., the child was vaccinated against chickenpox)
- SEX is equal to 1 (i.e., the child is male)
- HAD_CPOX is equal to 2 (i.e., the child did not contract chickenpox)

The third line calculates the ratio of the number of rows in **male_vaccinated_cpox** to the number of rows in **male_vaccinated_no_cpox.**

Finally, the fourth line stores the calculated ratio in the results dictionary, using the key "male". **The round()** function is used to round the ratio to four decimal places, as specified in the expected output format.

The same process is repeated for females, with the results stored using the key "female".

The final line of the function returns the results dictionary.

Question No 04:

A correlation is a statistical relationship between two variables. If we wanted to know if vaccines work, we might look at the correlation between the use of the vaccine and whether it results in prevention of the infection or disease [1]. In this question, you are to see if there is a correlation between having had the chicken pox and the number of chickenpox vaccine doses given (varicella).

Some notes on interpreting the answer. The `had_chickenpox_column` is either `1` (for yes) or `2` (for no), and the `num_chickenpox_vaccine_column` is the number of doses a child has been given of the varicella vaccine. A positive correlation (e.g., `corr > 0`) means that an increase in `had_chickenpox_column` (which means more no's) would also increase the values of `num_chickenpox_vaccine_column` (which means more doses of vaccine). If there is a negative correlation (e.g., `corr < 0`), it indicates that having had chickenpox is related to an increase in the number of vaccine doses.

Also, `pval` is the probability that we observe a correlation between `had_chickenpox_column` and `num_chickenpox_vaccine_column` which is greater than or equal to a particular value occurred by chance. A small `pval` means that the observed correlation is highly unlikely to occur by chance. In this case, `pval` should be very small (will end in `e-18` indicating a very small number).

[1] This isn't really the full picture, since we are not looking at when the dose was given. It's possible that children had chickenpox and then their parents went to get them the vaccine. Does this dataset have the data we would need to investigate the timing of the dose.

CODE:

```
import pandas as pd


def corr_chickenpox():

    # Load the data
    df = pd.read_csv('vaccination_chickenpox.csv', index_col = 0, usecols=[0, 1, 2])


    # Clean the data
    df = df[df['HAD_CPOX'].isin([1,2])]
    df = df[df['P_NUMVRC'].notna()]


    # Convert the values in HAD_CPOX to boolean values
    df['HAD_CPOX'] = df['HAD_CPOX'].apply(lambda x: True if x == 1 else False)


    # Calculate the correlation and p-value
    corr, pval = df.corr(method='pearson').loc['HAD_CPOX', 'P_NUMVRC']


    return corr
```

OR

2nd Code

```
import scipy.stats as stats
import pandas as pd


def corr_chickenpox():
    df = pd.read_csv('NISPUF17.csv', index_col=0)
    df = df[['HAD_CPOX', 'P_NUMVRC']].dropna()
    df = df[df['HAD_CPOX'] <= 2]
    df['HAD_CPOX'] = df['HAD_CPOX'].replace(2, 0)
    corr, pval = stats.pearsonr(df['HAD_CPOX'], df['P_NUMVRC'])
    return corr
```

The above question asks us to investigate if there is a correlation between having had chickenpox and the number of varicella (chickenpox) vaccine doses given. To accomplish this, we need to calculate the correlation coefficient between the had_chickenpox_column and num_chickenpox_vaccine_column columns of the given dataset using the scipy.stats.**pearsonr()** function.

The **pearsonr()** function returns two values - the correlation coefficient and the corresponding p-value. The correlation coefficient ranges between -1 and 1, where -1 indicates a perfect negative correlation, 1 indicates a perfect positive correlation, and 0 indicates no correlation. A p-value less than 0.05 indicates that the correlation is statistically significant.

We need to filter out the rows where had_chickenpox_column is not equal to 1 or 2, as these are the only two values that indicate whether a child had chickenpox or not. We also need to filter out rows where num_chickenpox_vaccine_column is missing (NaN).

After filtering the dataset, we calculate the correlation coefficient using **pearsonr()** and store the result in the variable corr. We then check the sign of the correlation coefficient to see if it's positive or negative. If it's positive, it means that an increase in the number of vaccine doses is associated with an increase in the number of children who did not have chickenpox. If it's negative, it means that having had chickenpox is associated with an increase in the number of vaccine doses.

Finally, we calculate the p-value and check if it's less than 0.05 to determine if the correlation is statistically significant.

EXPLAINATION:

The chickenpox_vaccine_correlation function takes in the df Pandas DataFrame as input, which is the dataset containing information about children's health and vaccination status.

The function first creates a new DataFrame df2 that only includes rows where the had_chickenpox_column and num_chickenpox_vaccine_column columns have valid values (i.e., not NaN).

Then, it converts the values in the had_chickenpox_column column to binary values, where 1 represents "Yes" (the child had chickenpox) and 0 represents "No" (the child did not have chickenpox). This is done using the replace method of the Pandas DataFrame.

Next, it calculates the Pearson correlation coefficient between the had_chickenpox_column and num_chickenpox_vaccine_column columns using the pearsonr function from the scipy.stats module. The correlation coefficient measures the strength and direction of the linear relationship between the two variables, with values ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation).

Finally, the function returns a tuple containing the correlation coefficient and the p-value. The p-value represents the probability of observing a correlation as extreme as the one calculated, assuming that the null hypothesis (that there is no correlation) is true. A very small p-value (typically less than 0.05) suggests that the observed correlation is unlikely to be due to chance alone, and therefore provides evidence for a correlation between the two variables.

NOTE: Link To Github Repository

GitHub - Musharraf-Raza-Khan/Introduction-To-Data-Science-IDS-: Welcome To My (IDS) Repository & Happy Learning !

“IDS-ASSIGNMENT

BY-MUSHARRAF RAZA KHAN | 52024”

THE END !