SPECIAL COURSE IN SOFTWARE ENGINEERING, AUTUMN 2024

EXERCISE 2 – DATA INSPECTION

In this exercise, you will **work on your own dataset** that you've chosen for your mini project. The objective of this exercise is to **load the dataset, clean, do a preliminary analysis, and then visualize using suitable plots**.

THIS EXERCISE WILL BE GRADED (MAX: 15 points)

SUBMISSION DEADLINE: OCTOBER 7, 2024, MIDNIGHT

Use the following link to accept your **GRADED EXERCISE 2 assignment**, which you will clone on your machine and then start working on. This is just like what you did in your *Graded Exercise 1*.

https://classroom.github.com/a/K882yCUW

Task 1. DataInspection class

Your program MUST be written inside the **DataInspection** class. The program's structure is already provided. Just fill in the logic.

Task 2. __init__(self) function:

This function acts as the **constructor**, just like in **Java**. You will need it to ensure the dataset you load and use as a **DataFrame** is accessible to every function for as long as it is running. '**self'** keyword is just the object of your class.

You don't need to write anything here. The provided code is sufficient for this exercise.

Task 3. Complete the load_csv() function

This function is responsible for loading the dataset. However, you will not **HARDCODE** the logic to load the dataset here. This function will actually be called from inside your **main()** function (instruction later in the document). In the **main()** function, you will ask the user to provide the path to the dataset, which will then be passed to the **load_csv()** function as the **filepath** argument. More instructions later in this document.

Task 4. Complete the classify_columns() function

This function will simply take every column from your **DataFrame** and call another function called **classify_and_calculate()**, where that column will be sent as an **argument**.

Task 5. Complete the classify_and_calculate () function

This function first checks that the column does not have too many missing values, using the **handle_missing_values()** function. If the number of missing values is more than 50%,

the column should be completely dropped from the **DataFrame**. More instructions on **handle_missing_values()** function later in the document.

Assuming the column is not dropped, the next thing you will do is call the **check_data_types()** function, where the data inside the column is checked for their datatype. More instruction on this function later in the document.

Next, calculate total number of unique values in the column. Use **nunique()** function. It will return the total number of unique values inside a column.

Check if the column values are numeric. Use **pd.api.types.is_numeric_dtype()** function to do that. Similarly, you can check if a column is made up of non-numeric data by using **pd.api.types.is_object_dtype()** function.

For a **numeric column**, you must also find out if that column has **ordinal** or **interval/ratio** type of data. You can use the **count of unique values** you calculated before to distinguish between these two types. For a **non-numeric column**, you can again use the **unique value count** to figure out if the column has **ordinal** or **nominal** type of data.

Depending on the type of data, calculate **mean, median,** or **mode** for the column.

Also, depending on what central tendency value (mean, median, mode) you calculate, you will visualize the column using suitable plot charts as follows:

- If you calculated mean, then you will call another function called **plot_histogram()** function to plot histogram for the column.
- If you calculated median on a numerical ordinal column, then you will call another function called **plot_boxplot()** to plot box plot for the column
- If you calculated median on a non-numerical ordinal column, then you will call another function called **plot_bar_chart()** to plot bar chart for the column.

This function **must return whatever central tendency value it calculates**.

Task 6. Complete the plot_histogram() function

This function simply takes a single column as an argument and plots a histogram for that column. The function will not return anything, but just show the plot.

Task 7. Complete the plot_box_plot() function

This function takes two columns as its arguments and plots a box plot for these columns. The function will not return anything, but just show the plot.

Task 8. Complete the plot_bar_chart() function

This function simply takes a single column as an argument and plots a bar chart for that column. The function will not return anything, but just show the plot.

Task 9. Complete the plot_scatter() function

This function takes two columns as its arguments and plots a scatter plot for these columns. The function will not return anything, but just show the plot.

Task 10. Complete the handle_missing_values () function

As mentioned in **Task 5**, this function checks if a column has any missing values. Depending on the count of missing values, appropriate stems must be taken.

This function takes a column as an argument, and calculates the percentage of missing values in that column:

- If the percentage is more than 50%, drop the column
- If the percentage is less than 50% and the column has numeric values, then fill the missing values with the column's median.
- If the percentage is less than 50% and the column has nominal values, then fill the missing values with the column's mode.

This function will simply return **True** if no column was dropped. Otherwise, return **False.**

Task 11. Complete the check_data_types() function

As mentioned in **Task 5**, this function checks the data type of the column's value. This function checks if a numeric value was stored as a string. If it was, then the values are changed to number using **to_numeric()** function.

This function will ***not return*** anything.

Task 12. Complete the numeric_columns() function

This function will collect all the numerical columns from the DataFrame into a single list. The function will return this list.

Task 13. Complete the ask_for_scatterplot() function

This function is called from inside the **main()** function.

In this function, the user is first shown all the numerical columns, and then asked to choose two columns for which the user wants to visualize a scatterplot.

This function does not take any argument and does not return anything.

Task 14. Complete the ask_for_boxplot() function

This function is called from inside the **main()** function.

In this function, the user is first shown all the numerical columns and ordinal columns, and then asked to choose one column from each list for which the user wants to visualize a box plot.

This function does not take any argument and does not return anything.

This function will ***not return*** anything.

Task 15. Complete the ask_for_correlation() function

This function is also called from inside the **main()** function.

The user is first shown all the numerical columns, and then asked to choose two for which correlation must be calculated.

This function takes the list of numerical columns as an argument and returns the correlation value.

Task 16. Complete the ask_for_std() function

This function is also called from inside the **main()** function.

The user is first shown all the numerical columns, and then asked to choose one column for which standard deviation must be calculated.

This function takes the list of numerical columns as an argument and returns the standard deviation value.

Task 17. Complete the ask_for_kurtosis() function

This function is also called from inside the **main()** function.

The user is first shown all the numerical columns, and then asked to choose one column for which kurtosis must be calculated.

This function takes the list of numerical columns as an argument and returns the kurtosis value.

Task 18. Complete the ask_for_skewness() function

This function is also called from inside the **main()** function.

The user is first shown all the numerical columns, and then asked to choose one column for which skewness must be calculated.

This function takes the list of numerical columns as an argument and returns the skewness value.

Task 19. Complete the main() function

This function is where all the other functions are controlled from.

Firstly, create an **object of the DataInspection** class. Call that object **analysis**.

Then ask the user to provide the file path to the dataset. After receiving the path, the function will **load_csv()** function.

Call the **classify_columns()** function.

Call the **ask_for_scatterplot()** and then **ask_for_boxplot()** functions.

Identify the list of numerical columns by calling the **numeric_columns()** function.

Then call the function to calculate correlation, standard deviation, kurtosis, and skewness passing the numerical columns as an argument.

Task 20. __init__ == '__main__' function

This function is needed to ensure that if your program is executed independently, then everything will be executed. However, if the program is imported as a module in another program, then only the necessary functions are executed and not the entire program.

You don't need to write anything here. The code is sufficient for this exercise.

Finish your code by following the above instructions carefully. Ensure everything is correct by running the test **LOCALLY ON YOUR MACHINE first**. Once all the tests pass on your machine, **ONLY THEN** should you commit and push. Doing so means officially submitting your exercise. Just like last time.