Bonus:
Missing Values Function

BANA 622
May 16, 2024

Emely Callejas, Ashley Cortez, Robert Pimentel, Angelica Verduzco

Table of Contents

Introduction

While navigating through cost reports csv files assigned for our project in BANA 620, we found that there were values missing with certain variables. To assess what variables had missing values and the percentage of missing values we created a user defined function using iterative methods that analyzes the variable and gives us a percentage of missing values within that certain variable. We also created a recursive version to see if there was a preference in which function computes faster and uses less memory. The purpose of this function is to display what the percentage of values missing are and also to assist us students on making decisions about what variables to include with whatever model chosen to train and test data on. This will also allow students to help facilitate decision making on whether to drop variables with missing values, or implement filling techniques such as mean, mode, median imputation and so on..

Methodology

When creating our analyze missing values function we created the function and implemented it in an iterative and recursive format. (Code snippets can be found in the appendix of this report.)

The iterative function itself was designed by first defining the function name and passing the parameter combined_data into it. We then implemented a user input where the user can type in a variable name, there is also a validation check to see whether or not the user inputs a variable that exists within the data frame and sets a counter to zero. Next, we create a for loop that searches each row for the variable inputted to count the values that are missing and increment the counter by 1 every time a missing value is found. Finally, we create calculated fields that calculate the number of missing values in the specified variable and another that calculates the percentage of missing values and rounds it to two decimal places.

The other method we used was to create the function recursively. This function was created by defining the function once again. After this, instead of just using the parameter combined_data, we also add two more, one for a current level and another for recursion depth. Next, we add a conditional if statement to check if the current level is greater than or equal to the max recursion depth then it will print out “Maximum Recursion Depth Reached”. Following this, we added another check to ensure that the user had an option to quit the loop. Most of the code in the middle is the same as the iterative method, the next part that is different is towards the end. When calling the recursive function we add plus one to the current level so that it can proceed to the next level of recursion.

Results

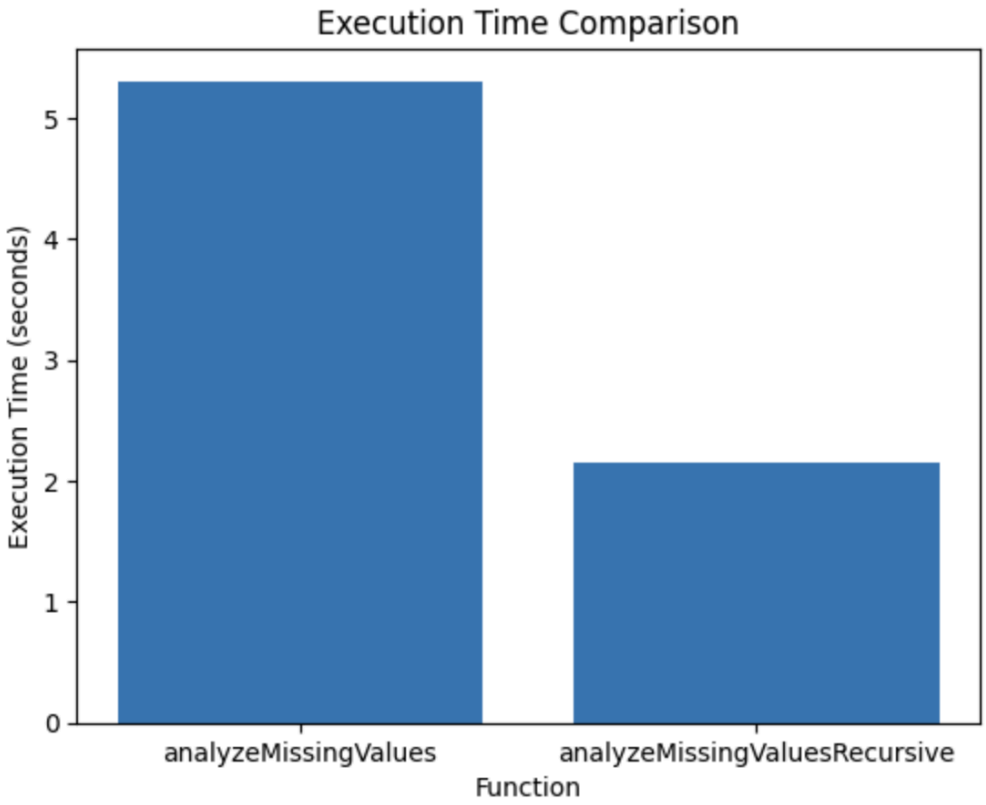
For testing and validation purposes, we ensured that each version of the function received the same variable input and validated that each version calculated the same percentage of missing values for a variable. An essential portion of the code was that the output displayed the name of the variable and the percentage of missing values.

As seen below both the iterative method and the recursive method were successful in displaying the same things:

```
Please enter the name of the variable you would like to analyze: Total_other_Assets
The percentage of missing values in 'Total_other_Assets' is 49.15%.
```

```
Please enter the name of the variable you would like to analyze (or 'q' to quit): Total_other_Assets
The percentage of missing values in 'Total_other_Assets' is 49.15%.
Please enter the name of the variable you would like to analyze (or 'q' to quit): q
```

For our performance analysis in this assignment, we conducted a time execution test. This test case included recording the execution time of the same input variable by the seconds it was executed. Below is a visual representation of the results comparing the performance of the two versions:



As we can see, the regular function typically provided the final output in a longer time than the recursive function. As seen in the chart above, the difference in time

for the two functions is somewhat small. Similarly to our task 2 assignment, this function also primarily relies on input from the user, where timing of function execution was dependent on the user's typing speed. For testing purposes we made sure to keep one name for the variable input to keep variance at a minimum.

As far as memory usage there was little to no difference between the two functions. As you can see below there is .02 difference between peak memory usage and the incremental memory usage stays the same for both functions at .02 MiB.

```
%memit analyzeMissingValues(combined_data)
```

```
Please enter the name of the variable you would like to analyze: Total_Income
The percentage of missing values in 'Total_Income' is 31.35%.
peak memory: 373.11 MiB, increment: 0.02 MiB
```

```
[20] %memit analyzeMissingValuesRecursive(combined_data)
```

```
Please enter the name of the variable you would like to analyze (or 'q' to quit): Total_Income
The percentage of missing values in 'Total_Income' is 31.35%.
Please enter the name of the variable you would like to analyze (or 'q' to quit): q
peak memory: 373.13 MiB, increment: 0.02 MiB
```

Discussion & Conclusion

This type of function allowed us to quickly assess whether certain variables in the dataset had missing values. If there were variables with a significant amount of missing values, it often made more sense to drop the variable since there is not much value it would add to a predictive model.

As far as functionality, there wasn't much of a difference between the accuracy or success of using a regular iterative function or a recursive function for this task. Through this exercise, we found that both regular and recursive functions were able to complete the task of inputting the variable name and finding the percentage of missing values for the given variable.

For our assignment, we would ultimately implement the regular iterative function over the recursive function primarily because of the maximum depth error we encountered while testing. We believe that this error happens when using the recursive function due to the fact that the dataset is quite large/contains many values, causing the program to overload with the amount of recursive calls.

Appendix

Regular Function:

```
[14] def analyzeMissingValues(combined_data):
        # Ask the user for the variable name
        variable = input("Please enter the name of the variable you would like to analyze: ")

        # Check if the variable exists in the DataFrame
        if variable in combined_data.columns:
            # Initialize a counter for missing values
            missingCount = 0

            # Iterate over each row in the specified column to count missing values
            for value in combined_data[variable]:
                if pd.isnull(value):
                    missingCount += 1

            # Calculate the percentage of missing values
            totalValues = len(combined_data[variable])
            missingPercentage = (missingCount / totalValues) * 100
            print(f"The percentage of missing values in '{variable}' is {missingPercentage:.2f}%.")
        else:
            print(f"The variable '{variable}' does not exist in the dataset.")

    # Call the function to analyze missing values
    analyzeMissingValues(combined_data)
```

```
Please enter the name of the variable you would like to analyze: Total_Income
The percentage of missing values in 'Total_Income' is 31.35%.
```

Recursive Function:

```
def analyzeMissingValuesRecursive(combined_data, currentLevel=0, maxRecursionDepth=3):
  # Base case: reached maximum recursion depth
  if currentLevel >= maxRecursionDepth:
    print("Maximum recursion depth reached.")
    return

  # Ask the user for the variable name
  variable = input("Please enter the name of the variable you would like to analyze (or 'q' to quit): ")

  # Check if the user wants to quit
  if variable == 'q':
    return

  # Check if the variable exists in the DataFrame
  if variable in combined_data.columns:
    # Initialize a counter for missing values
    missingCount = 0

    # Iterate over each row in the specified column to count missing values
    for value in combined_data[variable]:
      if pd.isnull(value):
        missingCount += 1

    # Calculate the percentage of missing values
    totalValues = len(combined_data[variable])
    missingPercentage = (missingCount / totalValues) * 100
    print(f"The percentage of missing values in '{variable}' is {missingPercentage:.2f}%.")

    # Recursively call the function with the next variable
    analyzeMissingValuesRecursive(combined_data, currentLevel + 1, maxRecursionDepth)
  else:
    print(f"The variable '{variable}' does not exist in the dataset.")

# Call the recursive function
analyzeMissingValuesRecursive(combined_data)
```

```
Please enter the name of the variable you would like to analyze (or 'q' to quit): Total_Income
The percentage of missing values in 'Total_Income' is 31.35%.
Please enter the name of the variable you would like to analyze (or 'q' to quit): q
```