**2-2 Assignment: Merging Data with Python**

Kayla Sacks

Southern New Hampshire University

DAT-430 Leverage Data for Org Results

Professor Aubrey Condor

September 15, 2024

There are many tools we can use when it comes to data analytics. One tool that is often used is Python due to its flexibility, scalability, compatibility with other languages, easy-to-read syntax, and the vast amount of libraries that can be used along with a strong community that can support you (Karl, 2024). This guide was created to show how helpful Python is for one part of the data analytics process, which is merging datasets. Merging datasets is an important step that often must be done as part of data preprocessing.

The first step in merging the data will be to import the *pandas* library and have Python read the data files so that we can work with them. This step can be seen in the screenshot below.



In the next step, I use the **.info()** command to get a summary of the dataframes and **.head()** command to show the first 5 rows of each dataset to make sure the data was loaded properly. The first screenshot shows the output for the Feb2014Data and the second shows the March2014Data.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Looking at the information for both data sets, we can see that the columns and data types are the same. The .**info()** function also shows us the number of rows in each dataset. It is important to make sure the data type for the same variable in each data set matches as it can cause incorrect results once merged or even cause the merge to fail. We can also see that the February dataset has 550 rows, and the March dataset has 850 rows. It is important to know the total number of rows in each data set so we know the total number of rows the merged data set should have.

*Always check the information about the datasets so you know what you are working with. It will be important to know this information when you are getting ready to merge, and after merging, the data sets.*

Next, before combining the datasets we want to check if there are any null values in either. This step can be done using the .**isnull()** function. Using just **.isnull()** will list each cell in the dataset which can make it hard to spot any that may come up as ‘True’, so to make it easier you can use the **.values.any()** with the **.isnull()** which will let you know if there are any null or NaN values(Atlassian, 2024). I used only **.isnull()** for the February date set first so you can see the results compared to if you use **.isnull.values.any()**.

A screenshot of a computer

Description automatically generated

As you can see, using the .values.any() makes it much easier to learn if there are any null values.

A computer screen shot of a computer

Description automatically generated

I did the March null check both with and without using the **print()** function to show that it is not needed but it can add clarity to the output.

Since both outputs returned ‘False’ that means there are no null or NaN values. If the output came out as ‘True’ that would mean that there was at least one null value, and you would want to investigate more. You could use **.isnull().sum()** to learn how many missing values there are for each variable. If there is a high number of missing values, you will want to take a closer look at the data set to see if it should still be used.

*It is important to always check each data set for any missing data to ensure the quality of the data. If to many data points are missing, the data set could cause inaccuracy in the analysis.*

Before merging the data, we need to decide how it should be merged. One important thing to note before merging the two datasets is the index column ‘id’. We need to understand if the index is important or if we can ignore it. Looking at the data, we don’t want the two ‘Timestamp’ columns for the same ‘id’, so we don’t want to use the **merge()** function since it joins on the index, but we want to keep the ‘id’ column while also not overlapping the other columns so instead we will use the **concat()** function.

With the **concat()** function we would need to make sure we ignore the index; the screenshot below shows how we use this function to merge the two datasets. We can see that a new column was created to use as an index while also keeping the ‘id’ column. We want this new column as we can’t use the ‘id’ column as the index after merging since it is no longer ‘unique’. We can see that the new data set includes all 1400 rows and 4 original columns (this does not include the new index column that was created). I also checked to see if there were any null values or duplicated data to ensure all data was merged correctly.

A screenshot of a computer

Description automatically generated

*Always review the merged dataset for missing or duplicated values as well as the number of rows to confirm that all data was merged correctly to ensure the quality and integrity of the data.*

The final step for this process is to save the new dataset as a .csv file using the **.to\_csv** function.



Since I am using Jupyter Notebook, I did not specify a specific location and when saving the new dataset to a csv file. This saves the file in the notebook to be downloaded to wherever you need it.

*Be sure to review the csv file to confirm that it was saved correctly with no errors.*

I hope this guild was helpful in teaching you how Python can help you in your data analytics journey in how to merge datasets while protecting the quality and integrity of the data.

**References**

Atlassian. (2024). *NaN Detection in Pandas | Atlassian*. Atlassian. <https://www.atlassian.com/data/notebook/how-to-check-if-any-value-is-nan-in-a-pandas-dataframe#:~:text=Count%20missing%20values%20in%20DataFrame&text=the%20chain%20of%20.->

Karl, T. (2024, May 31). *Benefits of Python for Data Analytics Explained*. New Horizons. <https://www.newhorizons.com/resources/blog/benefits-of-python-for-data-analytics>