Philip Chenaux-Repond

FINAL PROJECT WRITEUP – *AUTOMATED DATA CLEANING*

Overview:

For this project, I implemented an interactive tool that as input takes a dataset and eventually yields a cleaned dataset. I would like to emphasize the *interactive* aspect of this tool; my automated data cleaning algorithm (called autoClean) leverages human expertise to help make certain decisions when cleaning the dataset. These specific decisions will be outlined in the subsequent section of the report. I also would like to note that the dataset that is outputted is by no means a "perfectly cleaned" dataset, rather it is one way to clean a dataset. Moreover, given that the cleaning algorithm relies heavily on humans, the resulting outputted dataset is directly attributed to the user's decisions.

I **highly** recommend that you explore the tool (accessible as a Jupyter Notebook) yourself with your datasets to explore how the tool works and understand the capabilities/limitations of the software. autoClean is designed to be interacted with, featuring helpful explanations and user input fields, so please try it out for yourself with a variety of datasets. Alongside this writeup and the Jupyter notebook codebase, I have also included a demo dataset that you can use to run the tool on. Finally, note that autoClean does **not** work on Safari, please try to use Firefox if possible.

Design & Implementation:

As evident from within the Jupyter Notebook, you will see that the tool is separated into a 6 (actually 7) stage process. First, there is the necessary step of installing and importing necessary dependencies and libraries for the tool to be run correctly. Then the user will provide the dataset that will be cleaned. Note that alongside the dataset, the user can specify a custom delimiter when parsing the dataset. Both of these two initial stages form the **setup** phase of the tool.

Next, the user can begin exploring and interacting with the dataset in the **exploration** phase. I included this stage since I saw that it was necessary for users to first understand the dataset that they are dealing with from a variety of standpoints. There is little use in cleaning data that one does not understand. Therefore, I provided a suite of different outputs and interactive cells that allow users to explore the dataset beyond simply just looking at the data itself. Specifically, I rendered general dataset characteristics, the number of missing entries, datatypes, most common data entries, and histograms that display the distribution of each column. Additionally, I added an interactive widget that describes each of the columns statistically.

Following the exploration phase, we have the first data cleaning stage: **cleaning non-numerical data**. This involves only looking at columns that have a datatype labeled as an object (as opposed to int or float). The nature of data cleaning at this stage depends on the type of numerical data. First, the algorithm tries to characterize the data into one of three groups: dates/timestamps, categorical data (i.e. data that falls into a series of categories, e.g. race, gender, etc.), or largely individualized data (e.g. names, SSN, etc.). Once the data is

categorized, different methods of data cleaning are carried out. First, dates and timestamps are verified to be actual dates.

Next, categorical data is first cleaned to be more uniform across the column. Specifically, leading and trailing spaces are deleted, and all strings are turned into title case (i.e. where the first character of each string is capitalized). Note that title case was already quite a nuanced implementation since a very naïve implementation would cause a string like "philip's final project" to become "Philip'S Final Project" when really we want "Philip's Final Project." Both of these basic cleaning techniques are to ensure that all the categories follow the same pattern and can therefore be categorized properly. The next step was that performed with the categorical data was to try to identify situations where two data entries that may slightly differ were actually supposed to be the same entry. For example, consider the case where a user mistakenly writes "balck" instead of "black" for race. The algorithm uses an NGram similarity function to notice the similarity between these two words. But instead of blindly making these two entries identical, autoClean actually asks the user whether these two values can be considered the same and specifically to which of the two inputs should the entry be corrected. This step is an essential way to remove data errors in the original dataset and is a key step that leverages human expertise. Additionally, I have included checks that ensure labels of certain countries (specifically the US and UK) remain uniform throughout the column. For example, it would standardize the entries of "United States of America", "USA", "usa", etc. all to simply "US." The same is true for the UK.

Finally, if the type of non-numerical data is individualized, then the algorithm will address duplicates in the column. The duplicate cleaning procedure again requires user interaction in which the user is prompted with 3 ways to address duplicates depending on the column: keep all duplicates, drop duplicates that are identical across *all* columns, and drop duplicate values specific to this column. The final method therefore will result in the most number of duplicates being deleted and should be used cautiously. Note that there is also a 'smart duplication deletion' method that will be offered to the user after the next phase of cleaning.

The next stage involves **numerical data**. Unlike the non-numerical case, this does not try to categorize the data. Instead, it treats all numerical data in the same manner. First, it prompts users to consider data outliers. The program normalizes the data and asks users how many standard deviations of the mean they are willing to include in the dataset. Note that there is still an option to include all numerical data values of that column. Next, the numerical data runs through the same duplicate removal algorithm as described for the non-numerical data. Once all column-specific duplicates have been addressed, autoClean will now offer the opportunity to perform a more comprehensive duplicate cleaning scheme that looks at the entire data table – this is titled "Smart Duplicate Deletion." Specifically, this method considers all numerical columns of the dataset and computes a similarity matrix between all rows of the dataset. Then autoClean will output a set of two columns that it deems to be identical/almost identical to one another. The user will be able to view each of these pairs of duplicates in an interactive manner. Again, relying on the user's expertise and understanding of the dataset in this stage is an essential part of determining how to deal with duplicates in a safe way since oftentimes the program does not know what the dataset will be ultimately used for. Finally, in the case that the data columns are ZIP codes or

human ages, the algorithm makes sure that these are valid. If they are not, autoClean will set them to null which will then be addressed in the next stage.

The final data-cleaning stage addresses **missing data** entries. Before addressing the missing entries, the algorithm presents the user with a series of three charts that graphically visualize the missing dataset. This can help with identifying whether the missing data is MCAR or MAR. I recommend testing out the algorithm to see what each of these visualizations looks like as they can be quite powerful when addressing missing data entries. After the visualizations, the user is then prompted with a series of ways to address the missing data for each column. Depending on the data type there are 4-5 possibilities. The first three options relate to data deletion, namely: leave the missing data as it is, drop the rows with the missing entry, or drop the entire column. Then there is an option to perform simple imputation where the missing data is substituted by some constant value. Depending on the datatype of the column, this can either be some constant value (specified by the user), the most common value of that column, or if it is numerical data either the mean or median. The final option for dealing with the missing data only works for numerical data by using advanced imputation techniques. Specifically, I employed the use of Scikit Learn's Imputation library allowing users to use either the K-Nearest Neighbor Imputation method or the Multivariate Imputation by Chained Equations (MICE) approach. Both advanced options use machine learning techniques: KNN uses the n nearest neighbors which have values for that missing entry. Meanwhile, MICE takes multiple regressions of random samples of the dataset, then takes an average of the regressions, and imputes the missing entry based on the average regression.

Once the missing data has been addressed, then the final stage can take place, in which the dataset is once again reviewed and can be exported for further use. In this stage, the user is provided with three sets of interactive visualizations of the data. The first simply statistically describes each column (which can be compared with the initial descriptions provided at the beginning of the data cleaning phase. The following two visualizations attempt to graph the new dataset to try to extract meaningful information from the clean dataset. One is a histogram plot of each column to once again look at the distribution of entries, the second is a scatter plot allowing for the user's to observe any correlations between features of the dataset. Finally, the user can specify a filename and extension to export the cleaned dataset for further use.

Ethical Considerations:

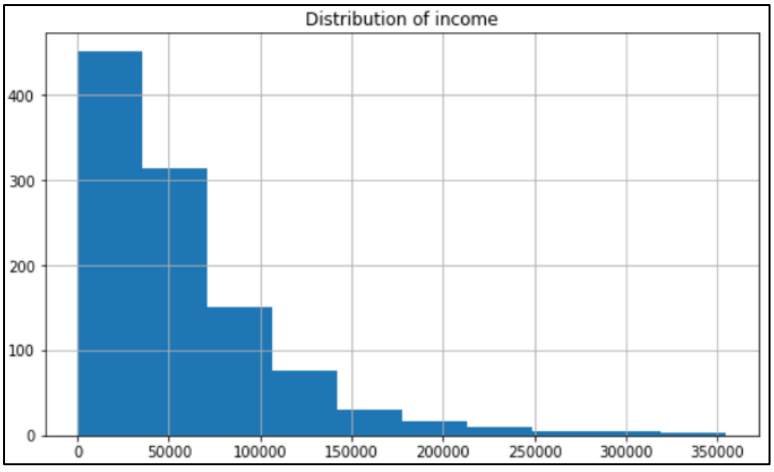
Throughout the implementation process, I had to think through what constitutes 'cleaned' data and whether cleaning the data in such a way would negatively impact the intrinsic value of the dataset. This is primarily why I decided to pair human interaction and expertise with this tool to create a more collaborative software that tries to mitigate the effect of a computer exclusively making poorly formed decisions without the validation/approval from humans that may understand the dataset better. One can argue that autoClean is simply shifting the blame away from the software to the user interacting with the software, but that is not the desired implication. Instead, I want this tool to be a way for users to experiment with cleaning data and see how different processes are applied to manifest different results. I did not want autoClean to be a deterministic program that always outputs the same 'cleaned' dataset every time. Thus, I opted for this interactive approach that is also

very explicit about the steps that it is taking. There is no black-box implementation where the user is unaware of the cleaning algorithms performed on its dataset. In fact, autoClean is quite the opposite. When manipulating datasets, transparency is both helpful and necessary.

Overall, this project made me realize a lot of the ethical design decisions that go into cleaning a dataset, and the significant repercussions that can ensue just by making a rather trivial change to the dataset. For example, dropping rows with missing data values is oftentimes a standard approach that users may not think twice about. But doing so may inadvertently shift the distribution or mean of other columns substantially. There is a fine balance that users ideally want to achieve through this cleaning process where the dataset can perform optimally without altering the underlying trends that the dataset raises. Even though autoClean may not be able to provide an optimally cleaned dataset that perfectly strikes this balance, it offers the necessary tools for the user to try to get as close to it as possible.

Sample Dataset Performance:

The dataset that is also submitted with the code is the data about whether individuals were approved to receive a loan used in an earlier assignment. I manipulated these datasets slightly so that they would work better with the algorithm (i.e. to show off more of its capabilities). But again, I *strongly* encourage you to test out running autoClean yourself on these provided datasets or any of your own to see how it performs. In the exploratory phase of autoClean, we can see that the dataset has some missing data in 4 of the 12 columns. The tool shows a handful of other visualizations, but one of which displays the most common data entries for each column, in addition to the distribution of each of the variables (below is the histogram of income). This will give us an indication of the range of values if we were to drop outliers later.




```
Characteristics of Input Dataset
Number of fields/characteristics: 12
Number of data entries: 1307
------------------------------------------------
Number of missing entries per column:
race          30
gender        18
date           0
zip           31
income       250
type           0
term           0
interest       0
principal      0
approved       0
adj_bls_2      0
id             0
dtype: int64
```

```
------------------------------------------------
Most Common Data Entries By Column

Column: race
black              420
white              400
hispanic/latino    218
other              161
asian               66
Name: race, dtype: int64

Column: gender
female        663
male          621
non-binary      5
Name: gender, dtype: int64

Column: date
2016-01-01    24
2018-03-01    24
2018-11-01    24
2017-07-01    23
2015-06-01    23
Name: date, dtype: int64

Column: zip
60623.0    265
60625.0    236
60614.0    203
60637.0    166
60615.0    160
Name: zip, dtype: int64

Column: type
home         497
auto         481
personal     329
Name: type, dtype: int64

Column: term
72     247
360    202
84     153
60     142
300    101
Name: term, dtype: int64

Column: approved
False    743
True     564
Name: approved, dtype: int64
```

Below is an example of a data cleaning process happening during the non-numerical phase where initially we see that there are several ‘unique’ entries in the ‘race’ column that we would classify as the same. Specifically, we have several instances of misspelled words, as well as what appear to be inconsistent instances of leading and trailing spaces. autoClean first addresses the formatting issues and then addresses misspellings of certain words like “Asiaan” and “Blck.” Note that by the end we have a set of six categories which is the desired outcome.

```
 Initial set of unqiue entries in 'race' column: hispanic/latino, other, black, white, asian, Hispanic/latino, Blac
k,  black, black , white , White, blck, None, hispanic/lation, asiaan

 Hispanic/Latino Hispanic/Lation
Would you consider the above two entries the same? (yes/no) yes
To which entry would you like the text to be merged to? (input field name)Hispanic/Latino

 Black Blck
Would you consider the above two entries the same? (yes/no) yes
To which entry would you like the text to be merged to? (input field name)Black

 Asian Asiaan
Would you consider the above two entries the same? (yes/no) yes
To which entry would you like the text to be merged to? (input field name)Asian


Final set of unqiue entries in 'race' column: Hispanic/Latino, Other, Black, White, Asian, None
```

Next, when addressing duplicates in the data, I eventually used the ‘smart duplication deletion’ method which yielded the set of following supposed identical results below. You can see that some human intuition is required in this stage to determine that these are likely duplicates, but the algorithm does a good job in identifying these similarities and is therefore, a very powerful tool when identifying similar rows.

	race	gender	date	zip	income	type	term	interest	principal	approved	adj_bls_2	id
254	hispanic/latino	male	NaN	60626.0	48705.0	home	180	5.463631	360591	False	0.12	254
253	hispanic/latino	male	2015-05-01	60626.0	48705.0	home	180	5.463631	360591	False	0.12	253

	race	gender	date	zip	income	type	term	interest	principal	approved	adj_bls_2	id
271	black	NaN	2012-08-01	60625.0	14199.0	NaN	48	8.784563	14844	True	0.13	271
270	black	NaN	2012-08-01	60625.0	14199.0	personal	48	8.784563	14844	True	0.13	270

	race	gender	date	zip	income	type	term	interest	principal	approved	adj_bls_2	id
1299	NaN	female	2018-06-01	60615.0	NaN	personal	24	9.658455	71093	True	1.82	1299
315	black	female	2018-06-01	60615.0	NaN	personal	24	9.658455	71093	True	1.82	315

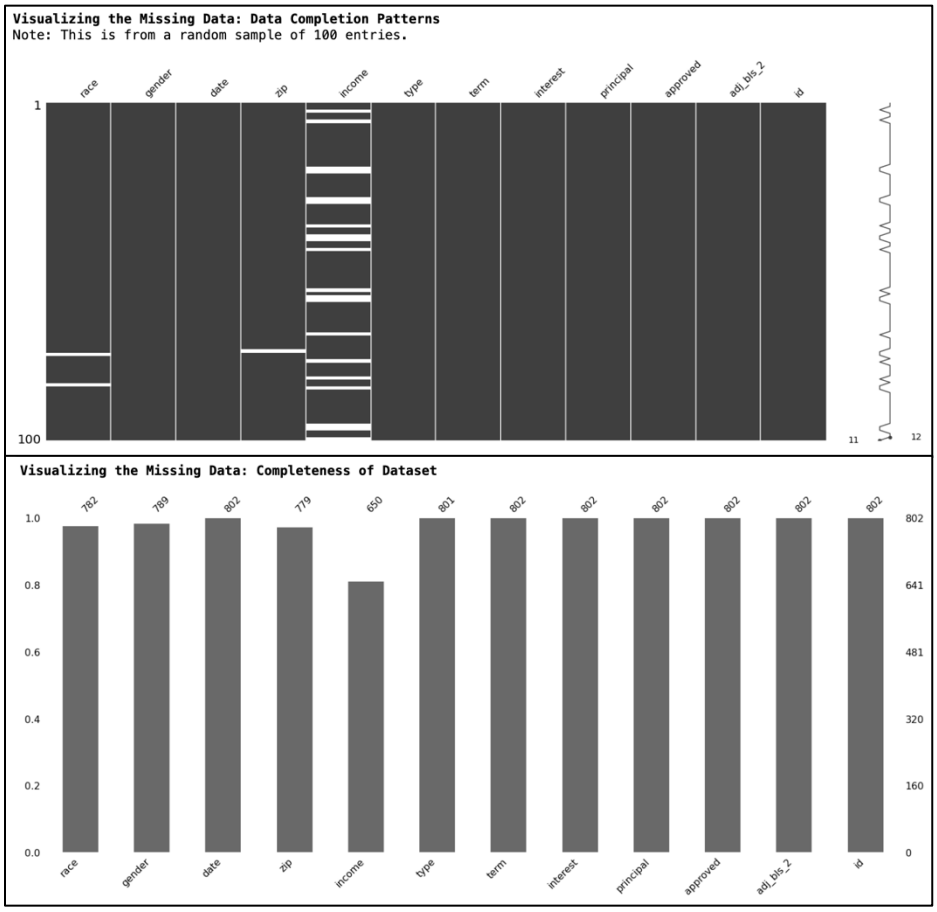
	race	gender	date	zip	income	type	term	interest	principal	approved	adj_bls_2	id
1301	white	female	2011-07-01	60625.0	25230.0	home	360	0.97487	182214	False	0.07	1301
1063	white	female	2011-07-01	NaN	25230.0	home	360	0.97487	182214	False	0.07	1063

	race	gender	date	zip	income	type	term	interest	principal	approved	adj_bls_2	id
1302	NaN	female	2011-07-01	60625.0	25230.0	home	360	0.97487	182214	False	0.07	1302
1063	white	female	2011-07-01	NaN	25230.0	home	360	0.97487	182214	False	0.07	1063

	race	gender	date	zip	income	type	term	interest	principal	approved	adj_bls_2	id
1302	NaN	female	2011-07-01	60625.0	25230.0	home	360	0.97487	182214	False	0.07	1302
1301	white	female	2011-07-01	60625.0	25230.0	home	360	0.97487	182214	False	0.07	1301

Philip Chenaux-Repond

In the final stage of data cleaning where missing data is addressed, we are first shown a series of helpful visualizations that depict the extent of the missing data. Note that these are not all the visualizations and I encourage you to explore the charts. But we can see that the income column is missing quite a few entries. One of the other visualizations maps the correlations between data which indicates that the missing income data is likely MCAR. Then when addressing the missing data, I simply imputed 'None' into the 'race' column and for 'gender' I imputed the most common data gender. Note that this is just for the purpose of demonstration, and by no means are the recommended steps to be taken at this stage. For the missing ZIP codes, I tried the advanced KNN imputation method and for income, I used the MICE imputation method. The results can be explored in-depth with the various charts in the final stage of autoClean, but it is interesting to see how various characteristics change after data cleaning.



Philip Chenaux-Repond

Sources Cited for this Project:

https://stackoverflow.com/questions/8924173/how-to-print-bold-text-in-python
https://stackoverflow.com/questions/49551336/pandas-trim-leading-trailing-white-space-in-a-dataframe
https://www.pythontutorial.net/python-string-methods/python-titlecase/
https://stackoverflow.com/questions/942543/operation-on-every-pair-of-element-in-a-list
https://pythonhosted.org/ngram/ngram.html
https://stackoverflow.com/questions/25341945/check-if-string-has-date-any-format
https://stackoverflow.com/questions/19363881/replace-none-value-in-list
https://towardsdatascience.com/interactive-controls-for-jupyter-notebooks-f5c94829aee6
https://stackoverflow.com/questions/64646490/calculate-similarity-between-rows-of-a-dataframe-count-values-in-common
https://www.kaggle.com/code/parulpandey/a-guide-to-handling-missing-values-in-python/notebook
https://scikit-learn.org/stable/modules/impute.html
https://www.kaggle.com/code/vvineeth/plotly-cufflinks-and-iplot/notebook