Machine Learning Pipeline Methodology

I have developed a machine learning pipeline that is contained in a single file (ml_pipeline.py) and is modular and extensible. It can be directly imported into a python file to accomplish data loading, exploration, and pre-processing, as well as feature generation, model training, and model evaluation. The pipeline ensures that the end user is importing the appropriate libraries and contains functions that can be used individually to explore a single set of variables or iteratively to explore different combinations of variables. This module can be leveraged in one of two ways. All parameters can be set as global variables by following the instructions at the top of the file before running the code on a new dataset. The default values for the different pipeline phases are set to these global variables. Alternatively, parameters can be passed directly into functions, overriding these default values. For each parameter (global variable), clear instructions are provided indicating the type of value that is expected.

Read/Load Data

The 'read_data' function loads a new CSV file and converts it to a pandas dataframe. This function takes two parameters: a required CSV filename and an optional unique identifier to set as the index. These values can be specified as global variables (RAW_DATA and UNIQUE_ID) or passed directly into the function. In this case, I have set them as global variables.

Explore Data

a. Data Summaries & Correlations

The 'describe_data' function provides summary statistics for the entire dataset. It prints whether the index is unique, exports a table that describes the datatype of each variable, exports a table that describes the range of values for each column, exports a table that describes the number of null observations and the percent of total observations that are null for each variable, exports a table that describes the correlation between each pair of variables, and exports a heatmap that displays these correlations between variables. Filenames for these figures can be specified as global variables (DATATYPES, RANGE, NULLS, CORRELATION_TABLE, and CORRELATION_IMAGE) or passed directly into the function. In this case, I have set them as global variables. In this case, the print statement indicates that the index (PersonID) is unique.

Figure 1. Variable Datatypes

SeriousDlqin2yrs	int64
RevolvingUtilizationOfUnsecuredLines	float64
age	int64
zipcode	int64
NumberOfTime30-59DaysPastDueNotWorse	int64
DebtRatio	float64
MonthlyIncome	float64
NumberOfOpenCreditLinesAndLoans	int64

NumberOfTimes90DaysLate	int64
NumberRealEstateLoansOrLines	int64
NumberOfTime60-89DaysPastDueNotWorse	int64
NumberOfDependents	float64

The first summary table provides the names of each variable in the dataset and confirms that all of the variables are uploaded as numeric (floats or integers). No variables need to be converted to numeric from a string, for instance, in this case.

Figure 2. Variable Distributions

	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	zipcode	NumberOfTime30-59DaysPastDueNotWorse	DebtRatio
count	41016	41016	41016	41016	41016	41016
mean	0.16140043	6.37587004	51.6834894	60623.8242	0.58923347	331.458137
std	0.36790438	221.61895	14.7468798	11.9843572	5.20562765	1296.10969
min	0	0	21	60601	0	0
25%	0	0.0343101	41	60618	0	0.17637526
50%	0	0.18973028	51	60625	0	0.36973568
75%	0	0.66715967	62	60629	0	0.86647063
max	1	22000	109	60644	98	106885

	MonthlyIncome	NumberOfOpenCreditLinesAndLoans	NumberOfTimes90DaysLate	NumberRealEstateLoansOrLines	NumberOfTime60-89DaysPastDueNotWorse	NumberOfDependents
count	33042	41016	41016	41016	41016	39979
mean	6578.99573	8.40347669	0.41959235	1.00880144	0.3715867	0.77323095
std	13446.8259	5.20732393	5.19038209	1.15382559	5.16964114	1.12126905
min	0	0	0	0	0	0
25%	3333	5	0	0	0	0
50%	5250	8	0	1	0	0
75%	8055.75	11	0	2	0	1
max	1794060	56	98	32	98	13

The second summary table shows the number of non-null rows for each variable (count), as well as each column's mean, standard deviation, minimum value, maximum value, and quartile values. For the Credit Dataset, this table highlights that there are several missing (null) values for Monthly Income as well as Number of Dependents. This will need to be considered during processing. Additionally, the table demonstrates that while most values for Revolving Utilization of Unsecured Lines are less than 1, there are outliers (eg. maximum value of 22,000). Similarly,

values for Debt Ratio should be less than 1, but we can see that there is a maximum value of 106,885, indicating incorrect input that will need to be accounted for during data cleaning. It is likely that annual income is entered for certain individuals instead of monthly income, since the maximum value listed is $1,794,060.

Figure 3. Null Values

	count_null	pct_null
SeriousDlqin2yrs	0	0
RevolvingUtilizationOfUnsecuredLines	0	0
age	0	0
zipcode	0	0
NumberOfTime30-59DaysPastDueNotWorse	0	0
DebtRatio	0	0
MonthlyIncome	7974	0.0162
NumberOfOpenCreditLinesAndLoans	0	0
NumberOfTimes90DaysLate	0	0
NumberRealEstateLoansOrLines	0	0
NumberOfTime60-89DaysPastDueNotWorse	0	0
NumberOfDependents	1037	0.0021

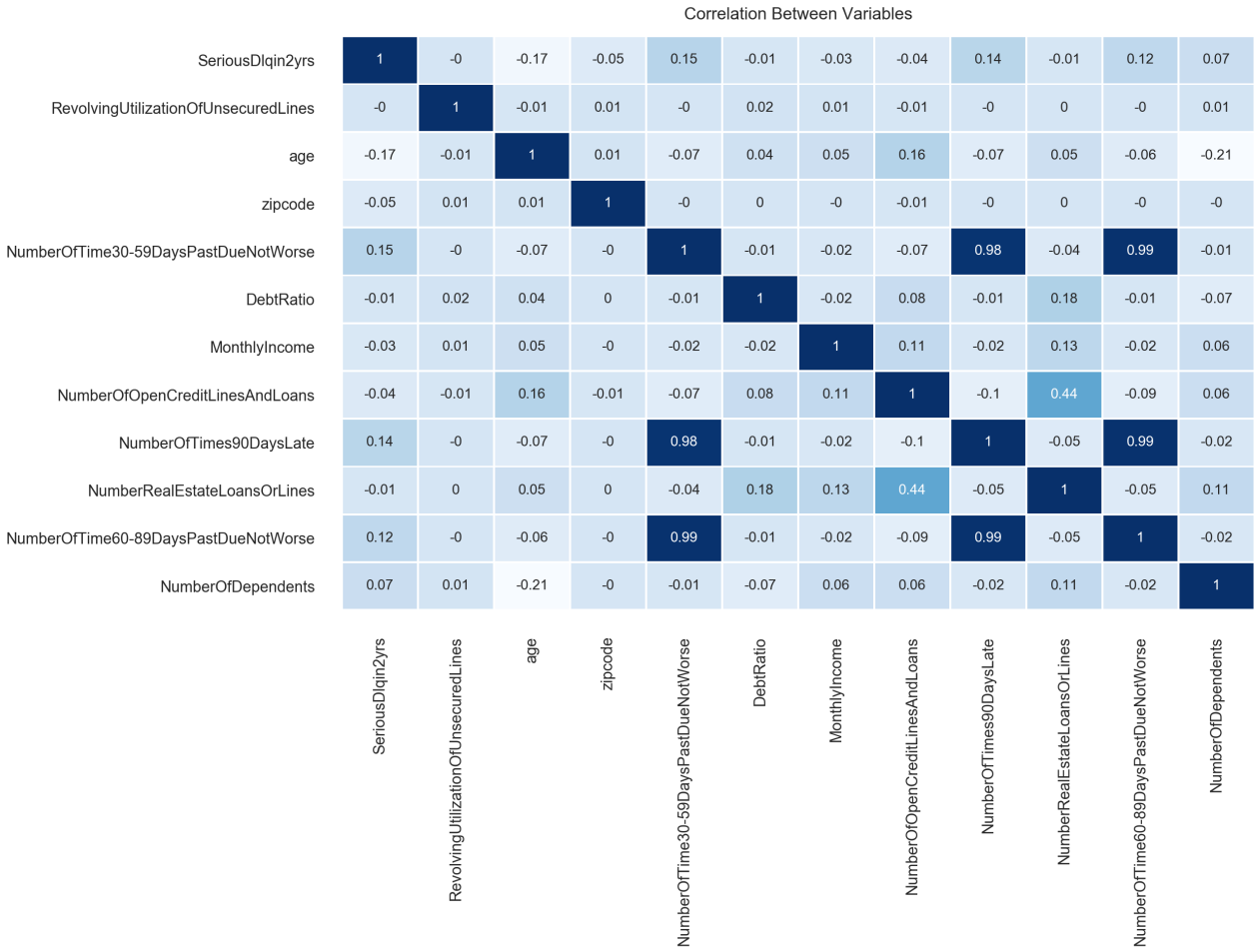
The third summary table shows the number of null values for each attribute as well as the percent of total rows that are null for each attribute. For the Credit Dataset, this table quickly highlights that approximately 1.6% of observations do not contain data on Monthly Income, and approximately 0.2% of observations do not contain data on Number of Dependents.

Figure 4. Variable Correlation Table

	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	zipcode	NumberOfTime30-59DaysPastDueNotWorse	DebtRatio
SeriousDlqin2yrs	1	-0.0046	-0.1737	-0.0451	0.1493	-0.0135
RevolvingUtilizationOfUnsecuredLines	-0.0046	1	-0.008	0.006	-0.002	0.0223
age	-0.1737	-0.008	1	0.0054	-0.0687	0.0388
zipcode	-0.0451	0.006	0.0054	1	-0.0024	0.0021
NumberOfTime30-59DaysPastDueNotWorse	0.1493	-0.002	-0.0687	-0.0024	1	-0.0116
DebtRatio	-0.0135	0.0223	0.0388	0.0021	-0.0116	1
MonthlyIncome	-0.0328	0.0058	0.0481	-0.005	-0.0152	-0.023
NumberOfOpenCreditLinesAndLoans	-0.0399	-0.0146	0.1599	-0.0092	-0.0707	0.0828
NumberOfTimes90DaysLate	0.1396	-0.0017	-0.069	-0.0015	0.9845	-0.0148
NumberRealEstateLoansOrLines	-0.0106	0.0048	0.0492	0.0031	-0.0379	0.1779
NumberOfTime60-89DaysPastDueNotWorse	0.1219	-0.0014	-0.0636	-0.0012	0.9885	-0.0133
NumberOfDependents	0.0657	0.0053	-0.211	-0.0017	-0.0078	-0.0706

	MonthlyIncome	NumberOfOpenCreditLinesAndLoans	NumberOfTimes90DaysLate	NumberRealEstateLoansOrLines	NumberOfTime60-89DaysPastDueNotWorse	NumberOfDependents
SeriousDlqin2yrs	-0.0328	-0.0399	0.1396	-0.0106	0.1219	0.0657
RevolvingUtilizationOfUnsecuredLines	0.0058	-0.0146	-0.0017	0.0048	-0.0014	0.0053
age	0.0481	0.1599	-0.069	0.0492	-0.0636	-0.211
zipcode	-0.005	-0.0092	-0.0015	0.0031	-0.0012	-0.0017
NumberOfTime30-59DaysPastDueNotWorse	-0.0152	-0.0707	0.9845	-0.0379	0.9885	-0.0078
DebtRatio	-0.023	0.0828	-0.0148	0.1779	-0.0133	-0.0706
MonthlyIncome	1	0.1071	-0.018	0.1273	-0.0153	0.0605
NumberOfOpenCreditLinesAndLoans	0.1071	1	-0.0982	0.4428	-0.0872	0.0602
NumberOfTimes90DaysLate	-0.018	-0.0982	1	-0.0547	0.9921	-0.0157
NumberRealEstateLoansOrLines	0.1273	0.4428	-0.0547	1	-0.048	0.1149
NumberOfTime60-89DaysPastDueNotWorse	-0.0153	-0.0872	0.9921	-0.048	1	-0.0165
NumberOfDependents	0.0605	0.0602	-0.0157	0.1149	-0.0165	1

Figure 5. Variable Correlation Heatmap



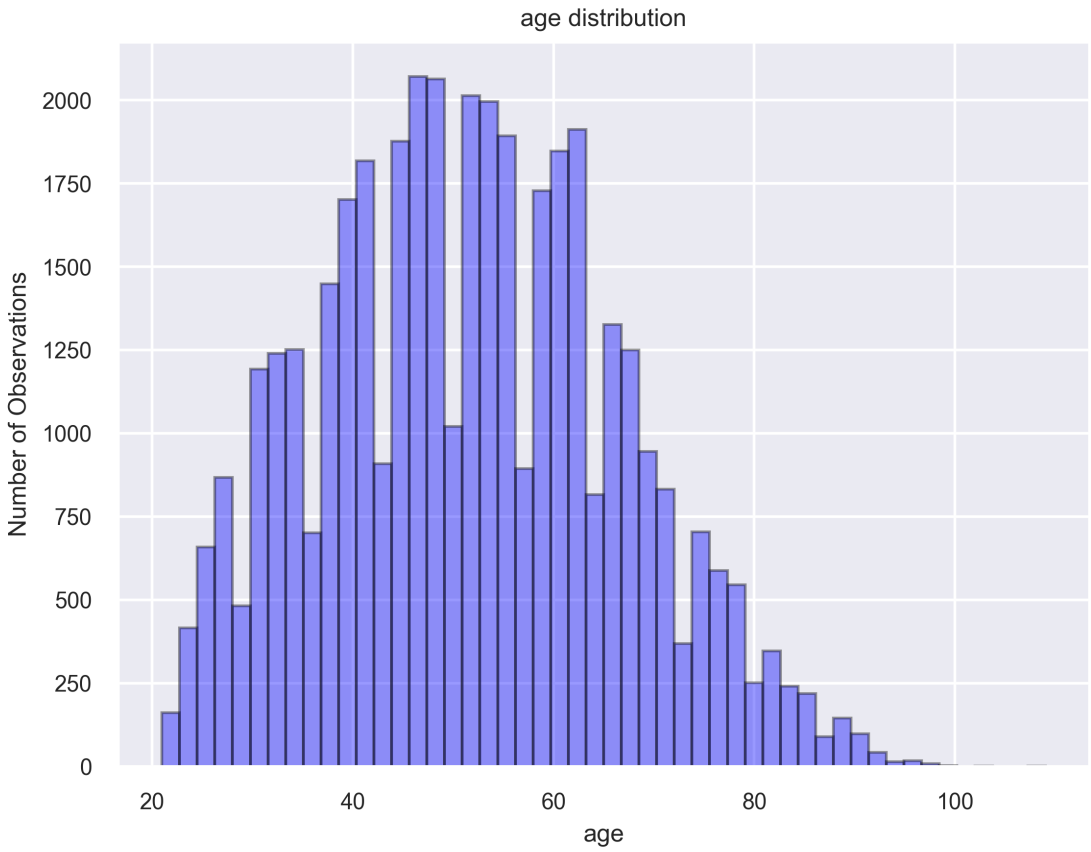
The fourth summary table highlights correlation between every pair of variables, and the correlation heatmap allows the end user to visualize this same data. For the Credit Dataset, it is interesting to note the high correlation between Number of Times 30-59 Days Past Due but No Worse in the Last 2 Years, Number of Times 60-89 Days Past Due but No Worse in the Last 2 Years, and Number of Times 90 or More Days Late. It is also interesting to note the relatively high correlation of 0.44 between Number of Open Credit Lines/Loans and Number of Real Estate Loans/Lines. In a regression analysis, for instance, an Analyst might be cautious about including highly correlated variables, as one may be masking variation in the outcome variable attributable to the other (things to consider during feature selection). It will be interesting to explore the 0.15 correlation between the outcome variable (Person has Experienced 90 Days Past Due Delinquency or Worse) and Number of Times 30-59 Days Past Due but No Worse.

b. Variable Distributions

The ‘distribution’ function creates a histogram for any attribute to explore the distribution of the variable and highlight outliers that may be skewing the distribution. If outliers are removed during processing, this function can be called again following cleaning to ensure that distributions are no longer skewed. Each time the function is called on a single variable, a PNG

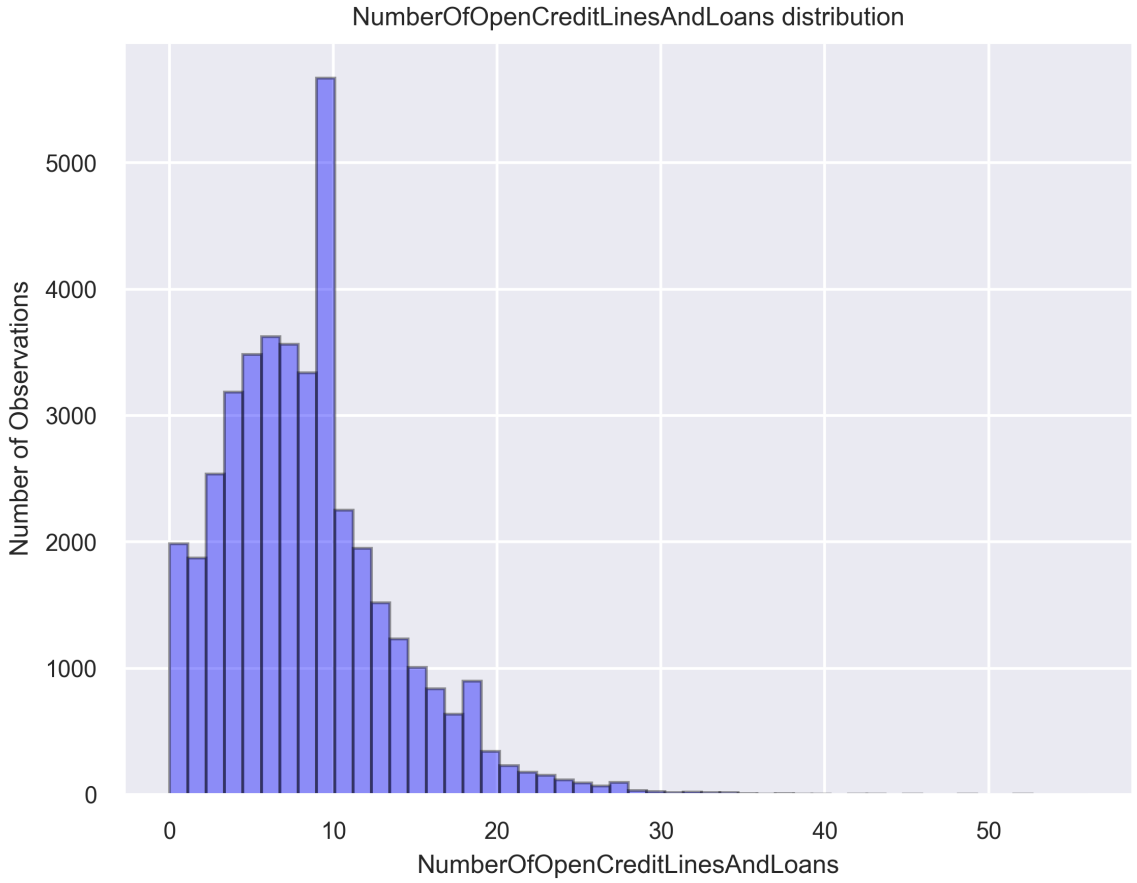
is exported for that variable. The feature of interest can be specified as a global variable (DISTRIBUTION) and a filename can be specified (HISTOGRAM), or these parameters can be passed directly into the function. In this case, I iterate through each variable in the dataset. Three histograms of interest are highlighted below.

Figure 6. Age Distribution



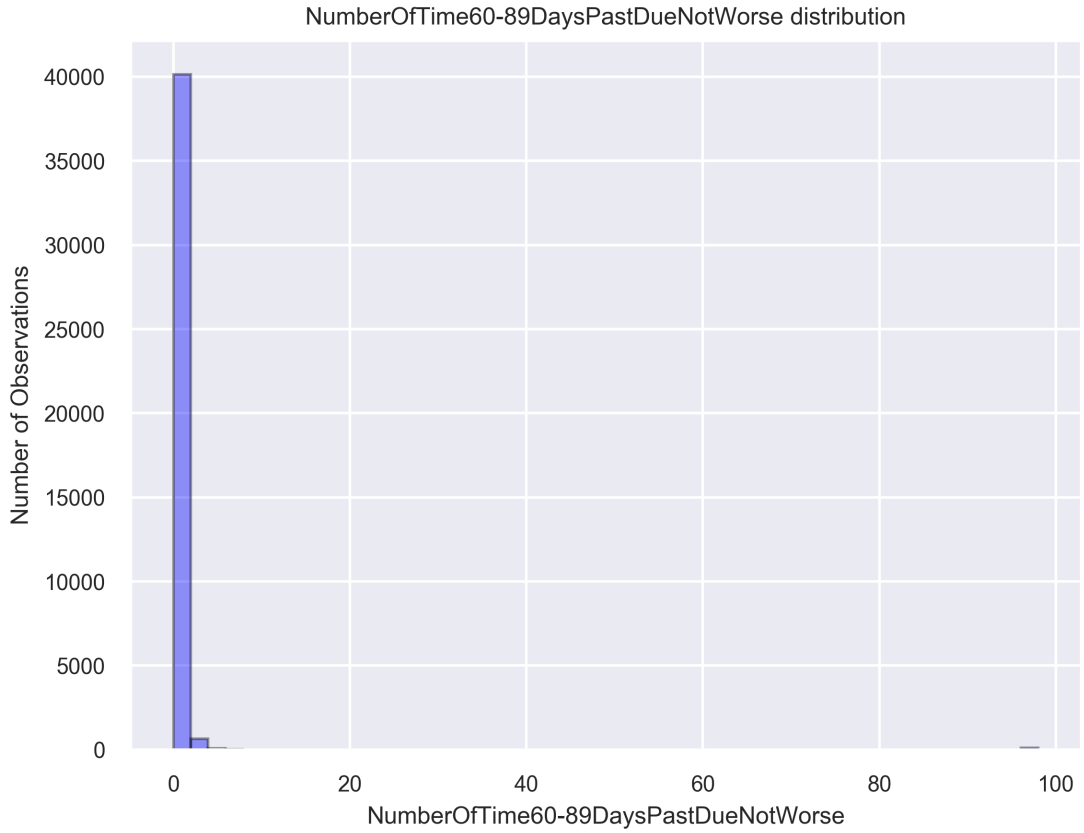
For the Credit Dataset, the age distribution histogram demonstrates the general distribution of ages in the data. It is quickly evident that only adults are included in the sample (above age 20), and that there is a concentration of individuals between 40 and 60 years old. It would be interesting to further explore why there appears to be a dip in the number of observations after every third bin, which may indicate a data collection error or rounding.

Figure 7. Number of Open Credit Lines and Loans Distribution



For the Credit Dataset, this distribution highlights the general spread of number of open credit lines and loans. There appears to be a steep increase from 0 to approximately 10 credit lines, and a gradual decline after that. It would be interesting to further explore why there are several thousand more observations in a single bin (approximately 10), which may suggest rounding during data collection and must be considered during analysis.

Figure 8. Number of Times 60-89 Days Past Due Distribution



For the Credit Dataset, this distribution quickly highlights a potential outlier at approximately 95 occurrences, which is likely skewing this distribution and hiding important details concentrated around 1-10 times. By removing this observation during data cleaning, replacing it, or simply masking it from the histogram, an Analyst could capture the true distribution of this variable and may be able to account for a skewed mean.

b. Visualize Variable Relationships

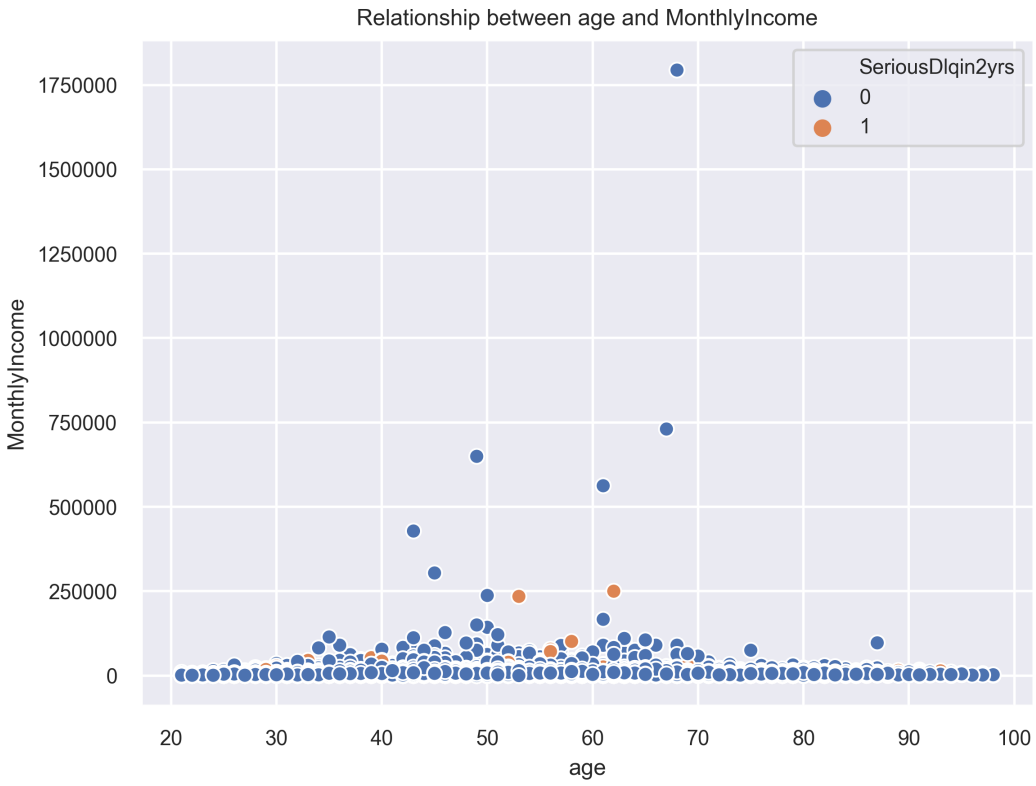
The next three functions can be used to visualize the relationship between any two variables in a dataset.

Continuous Variables: The function 'create_scatterplot' takes a dataframe, the names of two continuous variables, and a filename as input and returns a scatterplot mapping the relationship between these variables. The outcome variable appears on color in the case that it is binary (can be specified as a global variable). The two features of interest can also be specified as a global variable (CONTINUOUS_TWO) and the filename can be specified as a global variable (SCATTERPLOT). Alternatively, these parameters can be passed directly into the function. In this case, I pass several variable combinations of interest into the function as a parameter and highlight an example below.

Categorical Variables: The function 'create_heatmap' takes a dataframe, the names of two categorical variables, and a filename as input and returns a heatmap demonstrating the relationship between these variables. The two features of interest can be specified as a global variable (CATEGORICAL_TWO) and the filename can be specified as a global variable (HEATMAP). Alternatively, these parameters can be passed directly into the function. In this case, I pass several variable combinations of interest into the function as a parameter and highlight an example below.

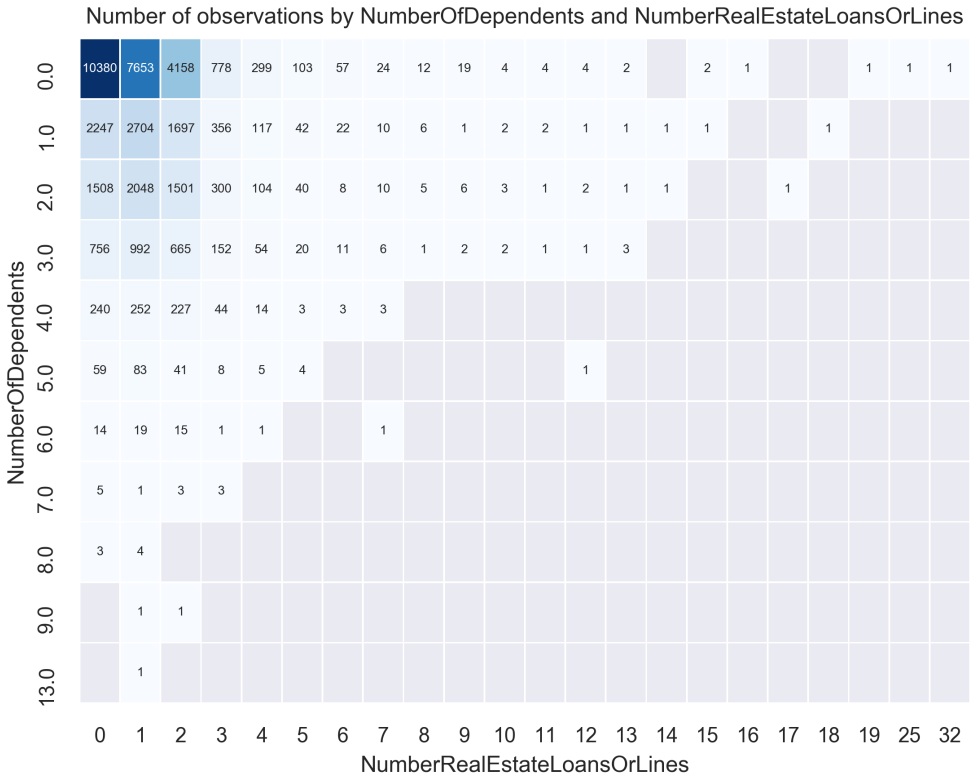
Continuous & Categorical Variables: The function 'create_barplot' takes a dataframe, the name of one categorical variable, the name of one continuous variable, and a filename as input and returns a barplot demonstrating the relationship between these variables. Specifically, it shows the average value of the continuous variable for each category. The two features of interest can be specified as global variables (CATEGORICAL_VAR and CONTINUOUS_VAR) and the filename can be specified as a global variable (BARPLOT). Alternatively, these parameters can be passed directly into the function. In this case, I pass several variable combinations of interest into the function as a parameter and highlight an example below.

Figure 9. Relationship Between Age and Monthly Income



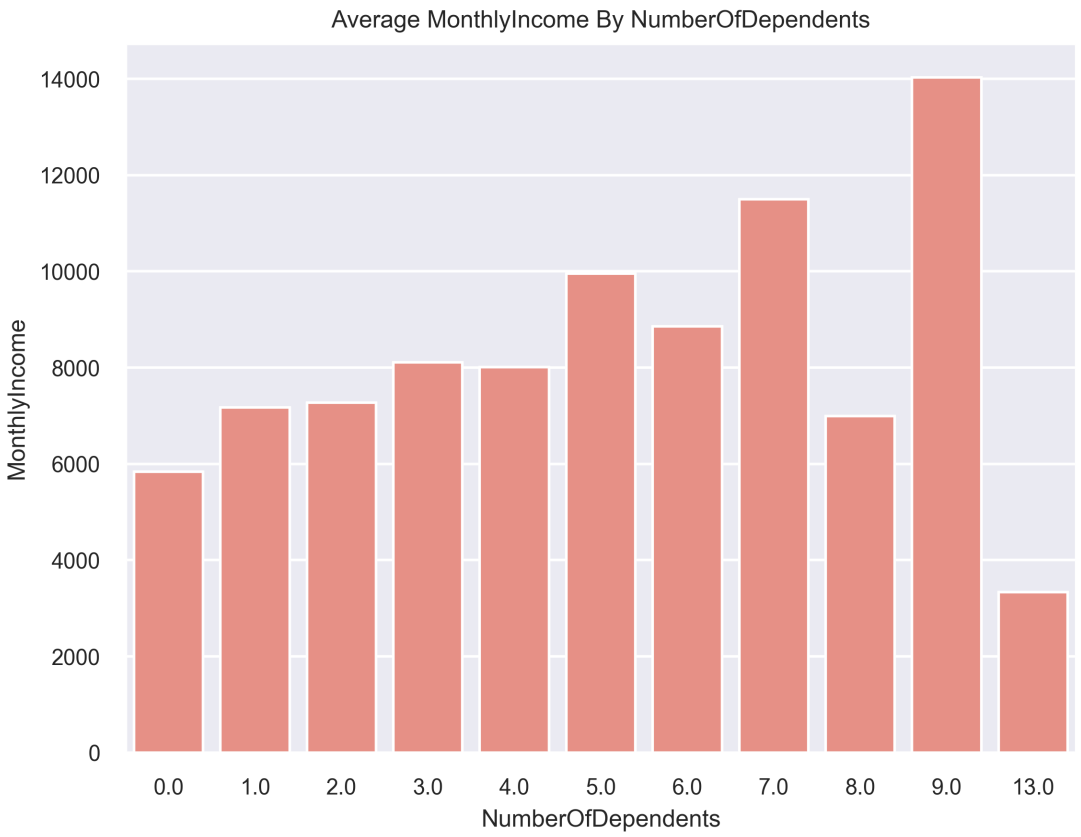
For the Credit Datatset, this visualization demonstrates that monthly income for at least 10 individuals is likely recorded as annual income. This visualization also demonstrates that monthly income appears to increase with age to a point, and then appears to decline again. By masking the outliers, an Analyst could gain a fuller picture of the underlying data.

Figure 10. Relationship between Number of Dependents & Number of Real Estate Loans/Lines



This visualization demonstrates that the Credit Dataset contains a high volume of observations for individuals who have taken out fewer than five real estate loans/lines and who have fewer than 4 dependents. There appears to be a wider spread of number of real estate loans/lines among individuals with no dependents, as compared to individuals with 6+ dependents.

Figure 11. Average Monthly Income by Number of Dependents



This visualization demonstrates the average monthly income among individuals by number of dependents. Specifically, average monthly income appears to increase as number of dependents increases, but drops dramatically after 9 dependents.

c. Identify Outliers

The 'find_outliers' function identifies and prints a dataframe containing potential outliers for a column defined as points that fall more than 1.5 times the interquartile range above the third quartile or below the first quartile for a given numeric column. The column of interest can be specified by the global variable OUTLIER, or can be passed directly into the function. In this case, I iterate through each column and print potential outliers by PersonID. It is important to examine the outliers in each column carefully, as some may actually be accurate and others may imply data entry mistakes (eg. it is possible for somebody in the dataset to be 103 years old, even though this is presented as an outlier). Similarly, while iterating over all variables presents "outliers" for zipcodes, it does not make sense to remove zipcodes that fall outside of a given numerical range.

Pre-Process Data

The 'pre_process' function fills in missing values with the mean value for that column. This function should only be applied to numeric columns. In this case, because all columns are numeric and only two columns contain missing data, the function is applied to the entire dataframe. The resulting dataframe should be saved over the original dataframe to ensure that the changes are captured. For the Credit Dataset, average values (by column) replace 7,974 rows of Monthly Income data and 1,037 rows of Number of Dependents data.

Generate Features/Predictors

The 'discretize_continuous_variable' function creates a new column in the dataframe containing a categorical variable to represent a specified continuous variable. The resulting dataframe should be passed to the next function to ensure that changes are preserved prior to training the model. The column of interest can be specified by the global variable FEATURE, or can be passed directly into the function. Bins can either by set manually or the function can create an even split based on a specified number. Again, these options can be specified in the global variables section (BINS). Finally, a list of labels can be specified and passed into the function (LABELS). In the case that labels are not passed, they will be set automatically.

Next, the 'create_dummies' function takes a categorical variable and creates dummy variables from it, which are concatenated to the end of the dataframe. The column of interest can be specified by the global variable DUMMIES, or can be passed directly into the function. Dummy headers will contain a prefix of up to 3 characters from the original variable name.

For this analysis and for the purpose of demonstrating the functionality of the pipeline without overfitting the data, I build a predictive model off of two features: age and zip code. The number of features can easily be increased based on further exploration such as a Principal Components Analysis. Therefore, I begin by slimming down the dataset to only include the two features of interest (age and zip code) as well as the outcome variable (SeriousDlqin2yrs). I then use the first function to discretize the age variable by a set of three inclusive, manually selected bins (set by the global variable BINS to [20, 39], [40, 69], [70, 110]). Note that all parameters are specified by the global variables. Next, I use the second function to create dummy variables for the zip code feature (again, specified as a global variable) as well as the new age_bin feature. Finally, for the purpose of this analysis I drop the original age, zip code, and age_bin variables, leaving the dummy features to use as predictors along with the outcome/target variable.

Build Decision Tree Classifier

The 'Classifier' class represents a decision tree classifier. Once a dataframe has been cleaned and limited to the features and target variable of interest, simply run "tree = Classifier(df)" to initialize and train a decision tree. This tree contains attributes for: a dataframe containing all features (x_data), a dataframe containing the target variable (y_data), a dataframe containing the training features (x_train), a dataframe containing the training target (y_train), a dataframe containing the testing features (x_test), a dataframe containing the testing target (y_test), a decision tree model trained on the training data (trained_model), and an array of predicted

outcomes for the x_test dataframe (y_hat). The tree also contains properties for: an accuracy score (accuracy) and the number of features used to predict (predictor_set_size). Specifically, based on the specified global outcome variable (TARGET), the 'create_x' method creates a dataframe of all features to be used as predictors and the 'create_y' method creates a dataframe containing the target column. The 'train' method takes a test size and random state as parameters (set by default to the global variables) and splits the full x and y dataframes into training and testing sets. It then trains scikit-learn's decision tree model to predict outcome values based on the training dataset (x_train). Finally, the 'predict' method runs a prediction on the trained model from the testing data (x_test) and saves these predictions as an attribute.

Using the slimmed down Credit Dataset, I train the decision tree model to predict whether a given observation or person will experience 90 days past due delinquency or worse (outcome variable) based on their features (in this case, zip code and age). This model is trained using a 0.2 size split (80% training, 20% testing) and a random seed of 1. The final output is a tree object.

Evaluate Classifier

Finally, the accuracy of the decision tree model trained on the Credit Dataset is assessed using the accuracy property of the Classifier class. The accuracy score represents the percentage of accurate predictions using the testing dataset, and can be obtained using 'tree.accuracy'. The model trained using the Credit data obtains an accuracy score of **0.84**. In future analyses, this score may be improved by methods including: adjusting the threshold at which the binary classification is considered to be a 1 or a 0 (currently set to the default of 0.5), performing additional data processing, and running a Principle Component Analysis to identify the most important features to include in the model to ensure accuracy and prevent overfitting.