James Taylor

DATA 630

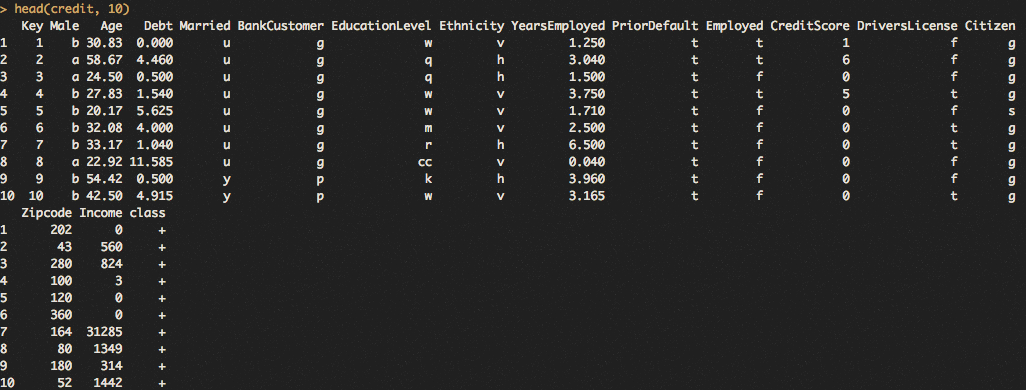
Exercise 2

**1) Introduction**

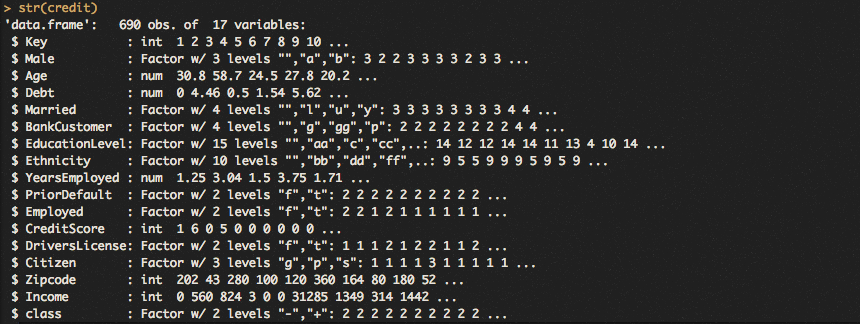
The data provided is from credit card applications. Its purpose is to determine how risky an individual is to loan money. Some of the variables have been altered to protect the person’s identity. The table below indicates the features and their values.

|  |  |  |  |
| --- | --- | --- | --- |
| **Male** : a,b | **Age** : continuous | **Debt** : continuous | **Married** : u, y, l, t |
| **Bank Customer** : g, p, gg | **Education Level** : c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff | **Ethnicity** : v, h, bb, j, n, z, dd, ff, o | **Years** **Employed** : continuous |
| **Prior** **Default** : t, f | **Employed** : t, f | **Credit** **Score** : continuous | **Driver’s** **License** : t, f |
| **Citizen** : g, p, s | **ZIP** **code** : continuous | **Income** : continuous | **Class** : +, - |

The meaning of the values for some of these features is impossible to know due to the need for confidentiality. It appears ‘t’ stands for True and ‘f’ is False. The Male feature is also Boolean but it’s unknowable which is true or false. The Class feature (last feature) is likely whether the individual was approved for the card with a ‘+’ or denied with a ‘-’. Married is interestingly not Boolean, so it likely captures divorced or never-married data.

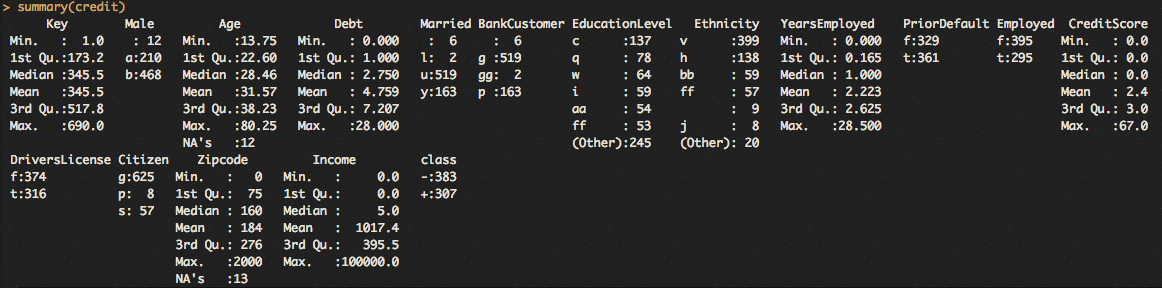


The above figure shows the first 10 rows of the dataset. Each row is an individual applying for credit. It is helpful to see a sample of the magnitudes of the continuous variables. From above we see that Credit Score is from zero to six, though these may not be the maximum and minimum for the whole dataset, but credit scores are between 300 and 850. This means they have been binned, or moved from a number to a categorical value like poor or excellent. The ZIP code feature doesn’t appear to be the same five-digit American style. Income doesn’t look to be annualized based on the fact that they are so low, for example there is a ‘3’ value which $3 earned annually seems unlikely. Debt also appears to be transformed, likely 11.5 denotes $11,500 of debt.



The above figure shows the structures of the features in the dataset. Many of features are factors, which store common values as levels. Using the Male feature as an example, there are 3 levels being 1,2 and 3 which correspond to the three unique values of the feature, a, b and empty. This is helpful for categorical values. The structure of each feature dictates the statistical approach taken to understand it. The rest of the features were integers or real numbers.

**2) Descriptive Statistics**

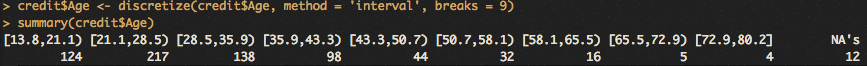
****

The above figure shows the descriptive statistics for each feature. It should be noted factor and numerical features have very different outputs. Numerical summary statistics display the quartiles, mean, minimum and maximum, showing the distribution of the feature. Take the age variable, where the output shows the youngest applicant was 13 years old and the oldest 80. The mean is 31 and the median is 28, showing the distribution may be skewed older. The factor variables only display the frequency of the levels. Looking at the male variable, we see there are 210 a’s and 468 b’s along with 12 empty values. It doesn’t break down the variable more than that.

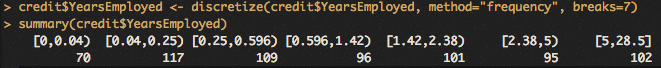
The difference between the str() and summary() command is their purpose. The structure command is only to understand what kind of data a variable is, like string or numerical. The summary command shows the distribution of the variable, like mean and quartiles, max and min, frequency of levels, etc.

**3) Variable Filters**

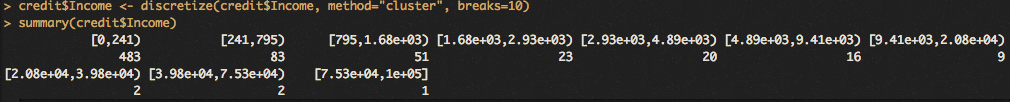
Discretization is the process of transforming a numerical variable into a categorical one, often referred to as binning. A simple example of this is grades, where 90 or more is an A, 80-89 is a B, and so on. I performed three different discretization techniques below.



Above is a discretization of the age variable by the equal interval length method, meaning he bin widths are the same size. The command executed above uses this method, where the input parameters specified to “interval” method and nine breaks. This means there will be nine bins crated that will all have the same width. The summary command output proves this to be the case with each bin being 7.4 units wide. It automatically removed the 12 missing values.



The above figure is a discretization of the Years Employed variable by the equal frequency method. In this method, approximately the same number of values will fall into the specified number of bins. In the above command, the frequency parameter was used and seven breaks were specified. The summary output shows about the same amount of values in each bin, however the bins are not the same width.



The figure above shows the discretization of the income variable by the K-means clustering method. This algorithm attempts to assign bins around means that best separate the values. The method parameter takes the cluster input and the number of breaks were specified to be 10. This is an algorithm that loops until no values change between groups when means are recalculated for each group and values are placed in them by their proximity. It group many of the low-income applications together and spread out the higher income groups, giving it skew towards high income distributions.

There are multiple reasons one should use a specific technique that is listed above. The equal interval method is helpful when the emphasis is on grouping things based on similar values. For example, if a race organizer want the top 10% of runners to compete, they would want 10 bins and use the fastest group. There are trade-offs with this method, however, as the groups close to the average would have lots of runners in them. A reason to use the equal frequency method is a desire to have groups of similar size. One could imagine a marathon with staggered starts, and the organizer want to break up groups by their qualifying time but also have the same number of runners in each staggered start. This way all the groups have similar sizes and similar ability. The k-means clustering method is helpful when preserving the original distribution is important. In the race example, if a runner is truly in a league of their own and grouping them with others is going to impede their world-class time, then a k-means would be helpful.



The command to remove a variable are above. The dim() command verifies the new data frame has one less variable (16 instead of 17). Removing variables is useful when they don’t help with our analysis. The above example removed a unique identifier for each application. Because we aren’t interested in the individual, and more so in the statistics.

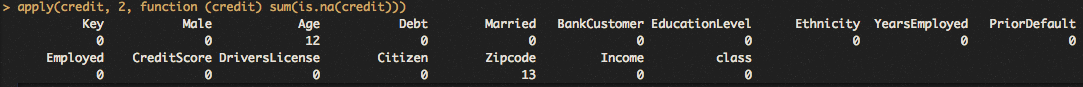
**4) Row Filters**

The dataset has missing values. The below command returns that 24 rows have missing values in them.

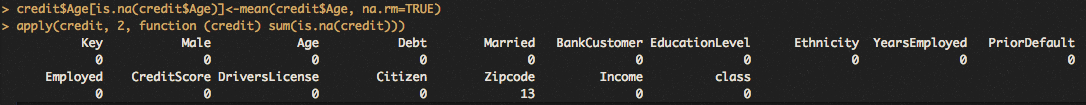
../../Desktop/7.5.png

This command counts the row, so if there are two missing values in the same row it will only count as one missing.

Below is a command and output to check which variables have the missing values.

****

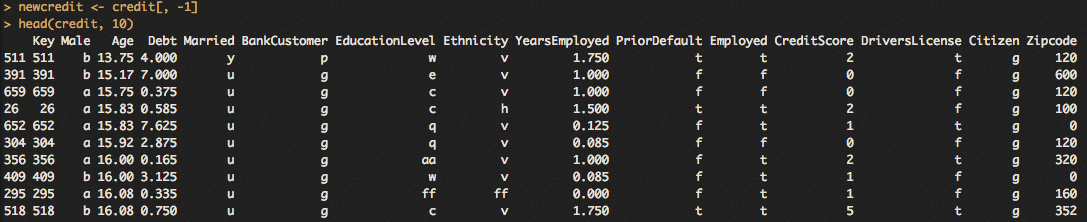
The output above shows that the age and ZIP code variables have missing values. Obviously 12 and 13 add to 25, so there are two NA’s in one row somewhere.



The command replaced all empty values with the mean in the Age variable. The output shows no empty values exist in the Age variable now. The command that did this takes the mean of the age variable (with NA’s removed) and assigns it to the age variable values where is.na = True. This replaces all NA’s in the variable with the mean.

It is necessary to handle the missing value because it would cause errors in calculations later.

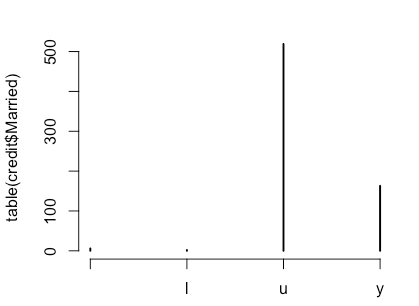
There are other ways to deal with missing data. One is to fill it with the most probable value, which would be done through regression. Another way is to replace the empty value with the mean of a class which this data point belongs two. For example, if this missing value is a soccer player and is missing income, take the average of all soccer players (as opposed to football and baseball) incomes and place it in the empty value.



The command and output above shows that the data frame was filtered by the age variable, youngest to oldest. The sorting would help with finding erroneous errors and finding specific observations. The first observation above has someone who is 13 years old. Not only oes this seem very young to apply for a credit card, but the Married variable has a ‘y’ variable which I assume means ‘yes’. This is likely an error.

**5) Data Visualization**

**../../Desktop/13.png**

****

Above is the command and plot outputted. It shows the counts of the levels in the married category. It shows ‘u’ is the most common followed by ‘y’. Also ‘l’ makes up a small percentage and there is an empty value that is also counted, or “”.

**6) Summary**

Pre-processing is important because low quality data will lead to low quality analysis. It is not much effort to fix the missing and erroneous values either, so the time spent is worth the return. Working with clean, accurate data yields more precise results due to the authenticity of the information.

Column filters only effect columns. They would be adding, removing or changing a data type for a column. The number of rows remains the same through all column filters. Row filters only effect rows. They add, remove or edit rows, all the while the number of columns remain unchanged.

The most difficult portion of this assignment was getting the mean of a column to replace the empty values in that same column. Firstly, one has to find the mean of the variable with the NA’s removed, which isn’t entirely hard. The interesting step is to assign this value to rows where a condition is met, in this case an empty value being there. It is similar to any other IF THEN assignment done in other programs, it was just new syntax and dealing with data frames.