Chapter 1: Data Exploration and Cleaning

Activity 1: Exploring Remaining Financial Features in the Dataset

Before beginning, set up your environment and load in the cleaned data set as follows:

import pandas as pd

import matplotlib.pyplot as plt #import plotting package

#render plotting automatically

%matplotlib inline

import matplotlib as mpl #additional plotting functionality

mpl.rcParams['figure.dpi'] = 400 #high resolution figures

mpl.rcParams['font.size'] = 4 #font size for figures

from scipy import stats

import numpy as np

df = pd.read\_csv('../Data/Chapter\_1\_cleaned\_data.csv')

1. Create lists of feature names for the remaining financial features.

These fall into two groups, so we will make lists of feature names as before, to facilitate analyzing them together. You can do this with the following code:

bill\_feats = ['BILL\_AMT1', 'BILL\_AMT2', 'BILL\_AMT3', 'BILL\_AMT4', 'BILL\_AMT5', 'BILL\_AMT6']

pay\_amt\_feats = ['PAY\_AMT1', 'PAY\_AMT2', 'PAY\_AMT3', 'PAY\_AMT4', 'PAY\_AMT5', 'PAY\_AMT6']

1. Use .describe() to examine statistical summaries of the bill amount features. Reflect on what you see. Does it make sense?

Use the following code to view the summary:

df[bill\_feats].describe()

The output should appear as follows:

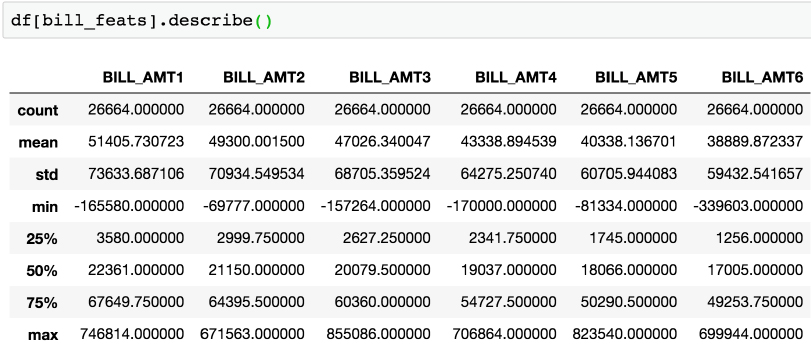


Figure 6.41: Statistical description of bill amounts for the past 6 months

We see that the average monthly bill is roughly 40,000 to 50,000 NT dollars. The reader is encouraged to examine the conversion rate to their local currency. For example, 1 US dollar ~= 30 NT dollars. Do the conversion and ask yourself, is this a reasonable monthly payment? We should also confirm this with the client, but it seems reasonable.

We also notice there are some negative bill amounts. This seems reasonable because of possible overpayment of the previous months' bill, perhaps in anticipation of a purchase that would show up on the current months' bill. A scenario like this would leave that account with a negative balance, in the sense of a credit to the account holder.

1. Visualize the bill amount features using a 2 by 3 grid of histogram plots using the following code:

df[bill\_feats].hist(bins=20, layout=(2,3))

The graph should look like this:

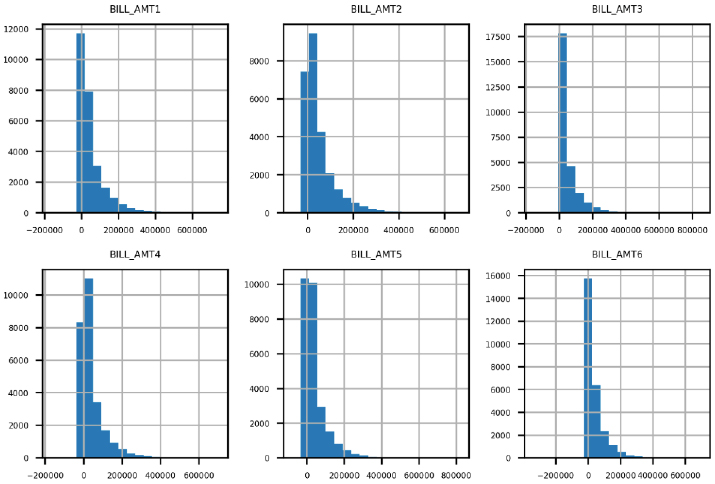


Figure 6.42: Histograms of bill amounts

The histogram plots in Figure 6.42 make sense in several respects. Most accounts have relatively small bills. There is a steady decrease in the number of accounts as the amount of the bill increases. It also appears that the distribution of billed amounts is roughly similar month-to-month, so we don't notice any data inconsistency issues as we did with the payment status features. This feature appears to pass our data quality inspection. Now, we move on to the final set of features.

1. Use the .describe() method to obtain a summary of the payment amount features using the following code:

df[pay\_amt\_feats].describe()

The output should appear thus:

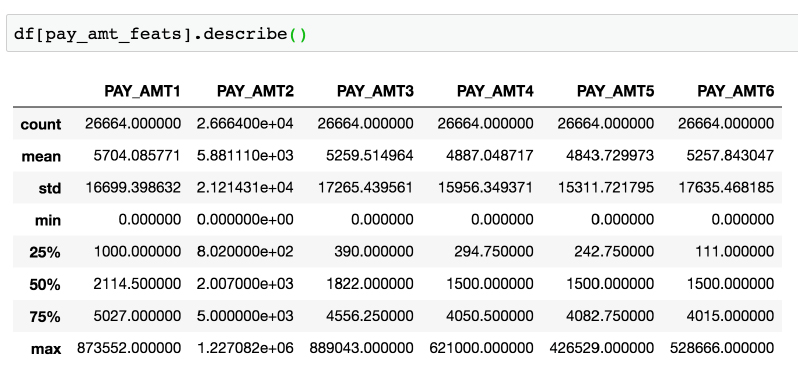


Figure 6.43: Statistical description of bill payment amounts for the past 6 months

The average payment amounts are about an order of magnitude (power of 10) lower than the average bill amounts we summarized earlier in the Activity. This means that the "average case" is an account that is not paying off its entire balance from month to month. This makes sense in light of our exploration of the PAY\_1 feature, for which the most prevalent value was 0 (account made at least the minimum payment but did not pay off the whole balance). There are no negative payments, which also seems right.

1. Plot a histogram of the bill payment features similar to the bill amount features, but also apply some rotation to the x-axis labels with the xrot keyword argument so that they don't overlap. Use the xrot=<angle> keyword argument to rotate x-axis labels by a given angle in degrees using the following code:

df[pay\_amt\_feats].hist(layout=(2,3), xrot=30)

In our case, we found that 30 degrees of rotation worked well. The plot should look like this:

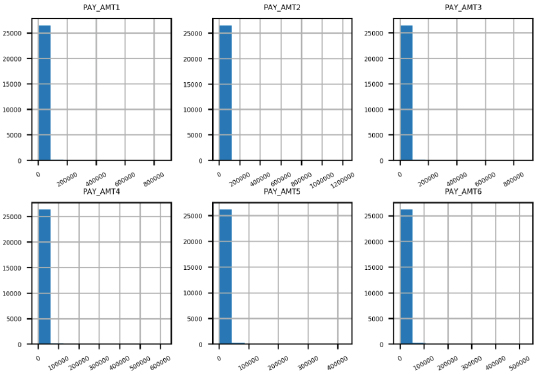


Figure 6.44: Histograms of raw payment amount data

A quick glance at this figure indicates that this is not a very informative graphic; there is only one bin in most of the histograms that is of any noticeable height. This is not an effective way to visualize these data. It appears that the monthly payment amounts are mainly in a bin that includes 0. How many are in fact 0?

1. Use a Boolean mask to see how many of the payment amount data are exactly equal to 0 using the following code: Do this with the following code:

pay\_zero\_mask = df[pay\_amt\_feats] == 0

pay\_zero\_mask.sum()

The output should look like this:

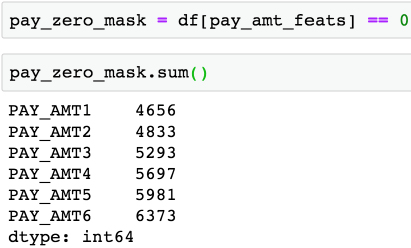


Figure 6.45: Counts of bill payments equal to 0

Does this data make sense given the histogram in the previous step?

The first line here creates a new DataFrame called pay\_zero\_mask, which is a DataFrame of True and False values according to whether the payment amount is equal to 0. The second line takes the column sums of this DataFrame, interpreting True as 1 and False as 0, so the column sums indicate how many accounts have a value of 0 for each feature.

We see that a substantial portion, roughly around 20-25% of accounts, have a bill payment equal to 0 in any given month. However, most bill payments are above 0. So, why can't we see them in the histogram? This is due to the range of values for bill payments relative to the values of the majority of the bill payments.

In the statistical summary, we can see that the maximum bill payment in a month is typically 2 orders of magnitude (100 times) larger than the average bill payment. It seems likely there are only a small number of these very large bill payments. But, because of the way the histogram is created, using equal sized bins, nearly all the data is lumped into the smallest bin, and the larger bins are nearly invisible because they have so few accounts. We need a strategy to effectively visualize this data.

1. Ignoring the payments of 0 using the mask you created in the previous step, use pandas' .apply() and NumPy's np.log10() method to plot histograms of logarithmic transformations of the non-zero payments. You can use .apply() to apply any function, including log10, to all the elements of a DataFrame. Use the following code to complete the preceding step:

df[pay\_amt\_feats][~pay\_zero\_mask].apply(np.log10).hist(layout=(2,3))

This is a relatively advanced use of pandas, so don't worry if you couldn't figure it out by yourself. However, it's good to start to get an impression of how you can do a lot in pandas with relatively little code.

The output should be as follows:

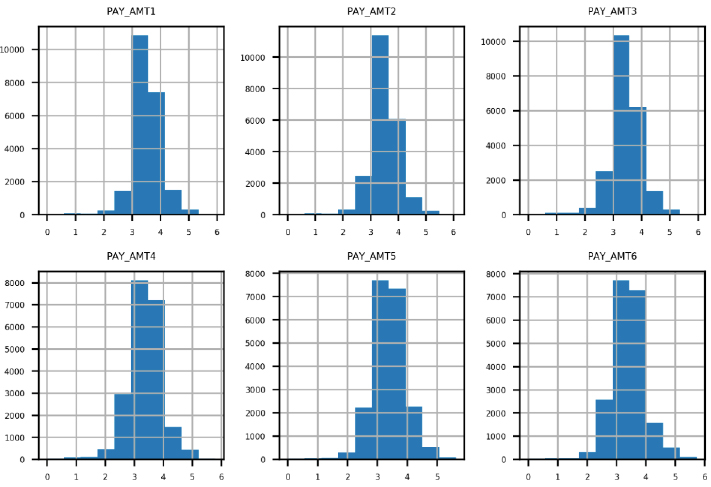


Figure 6.46: Base-10 logs of non-zero bill payment amounts

While we could have tried to create variable width bins for better visualization of the payment amounts, a more convenient approach that is often used to visualize, and sometimes even model, data that has a few values on a much different scale than most of the values, is a logarithmic transformation, or log transform. We used a base-10 log transform. Roughly speaking, this transform tells us the number of zeros in a value. In other words, a balance of at least a million dollars, but less than ten million, would have a log transform of at least 6 but less than 7, because 106 = 1,000,000 (and conversely log10(1,000,000) = 6) while 107 = 10,000,000.

To apply this transformation to our data, first, we needed to mask out the zero payments, because log10(0) is undefined. We did this with the Python logical not operator ~ and the zero mask we created already. Then we used the pandas .apply() method, which applies any function we like to the data we have selected. In this case, we wished to apply a base-10 logarithm, calculated by np.log10. Finally, we made histograms of these values.

The result is a more effective data visualization: the values are spread in a more informative way across the histogram bins. We can see that the most commonly occurring bill payments are in the range of thousands (log10(1,000) = 3), which matches what we observed for the mean bill payment in the statistical summary. There are some pretty small bill payments, and also a few pretty large ones. Overall, the distribution of bill payments appears pretty consistent from month to month, so we don't see any potential issues with these data.