Exploratory Data Analysis for Machine Learning

Finn Tomasula Martin
COSC-4557

1 Introduction

An important aspect of machine learning is the data. Without any data there is nothing to learn from and thus no learning can happen. This necessity of data means that we must allocate a significant amount of our focus to the data to ensure accurate learning when tackling any machine learning problem. Unfortunately, in the real world data often won't be adequately suited for machine learning and we must take steps to modify it to be better suited for model building without introducing any bias or additional issues. These steps are commonly referred to as preprocessing and there are many established techniques that fall under this domain. In this report, we will take a look at a couple different datasets, analyze the data and then discuss the effects of various preprocessing methods, applying them to the data if applicable.

2 Data

For this report, we will take a look at two different datasets. The first, which I will call wine quality contains a collection of statistics on 1,599 red wines. This dataset contains one target variable, quality and twelve feature variables. The second dataset which I will call tumor contains a collection of statistics on 309 patients with tumors. The target variable for this dataset is primary tumor (original location of tumor in body) and there are seventeen feature variables.

3 Results of Analysis

To analyze our data we will explore three common issues found in datasets as well as the effects of possible fixes to these issues. The three issues we will examine are the possibility of missing values, the data distribution of features and multicollinearity.

3.1 Missing Values

The first thing we will look at is the possibility of missing values. Missing values may occur for some observations in a data set and they can lead to many problems including errors or inaccurate predictions. We can check our two datasets for missing values in R by combining the sum function and the is.na function, to count how many observations equal NA.

```
numberMissing <- sum(is.na(df))
```

When we apply this check to our wine quality dataset, it results in 0 which is great and means we do not have to worry about missing values in that case. But when we apply it to our primary tumor dataset we get a result of 207, which is quite significant (Note: For the dataset I am using the missing values were represented using “”, so I had to replace all instances of “” with NA like so, df[df == “”] <- NA. In practice, you may need to look at your data to determine what is being used to represent a missing observation and adjust accordingly). There are several things we can do to account for these missing values such as fill them in with the mean or mode value or using an algorithm that accounts for missing values. In our case we will apply the simplest solution which is just removing any columns rows that contain missing values. This can be done in R with the na.omit function.

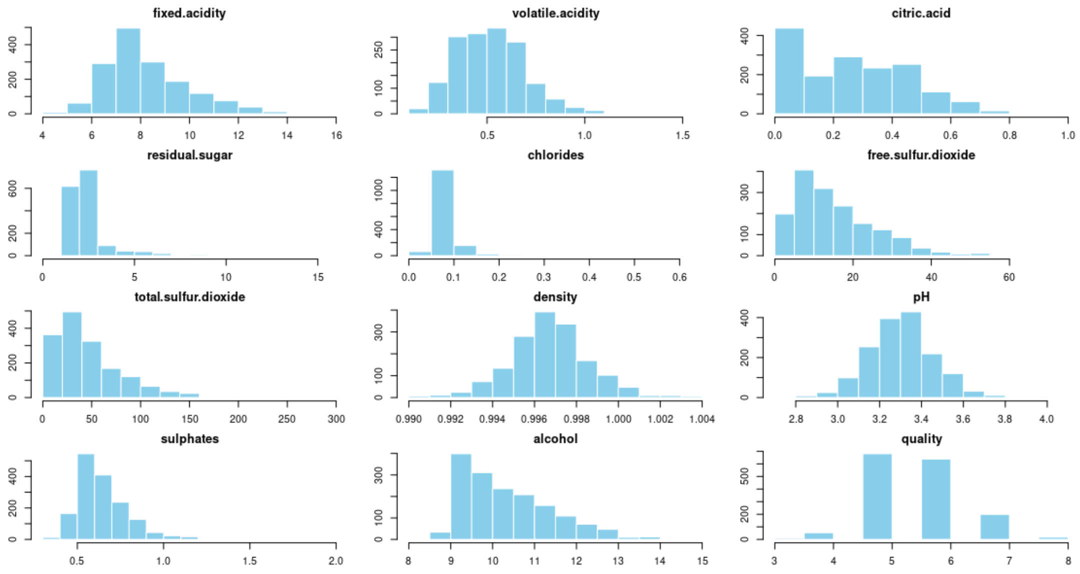
```
df <- na.omit(df)
```

Like any other method there are advantages and disadvantages to fully dropping rows. It our case, it reduced the number of observations from 309 to 120 which may result in a pretty significant loss of information. On the other hand it avoids adding in any bias or skewing our results more dramatically in any which way.

3.2 Feature Distribution

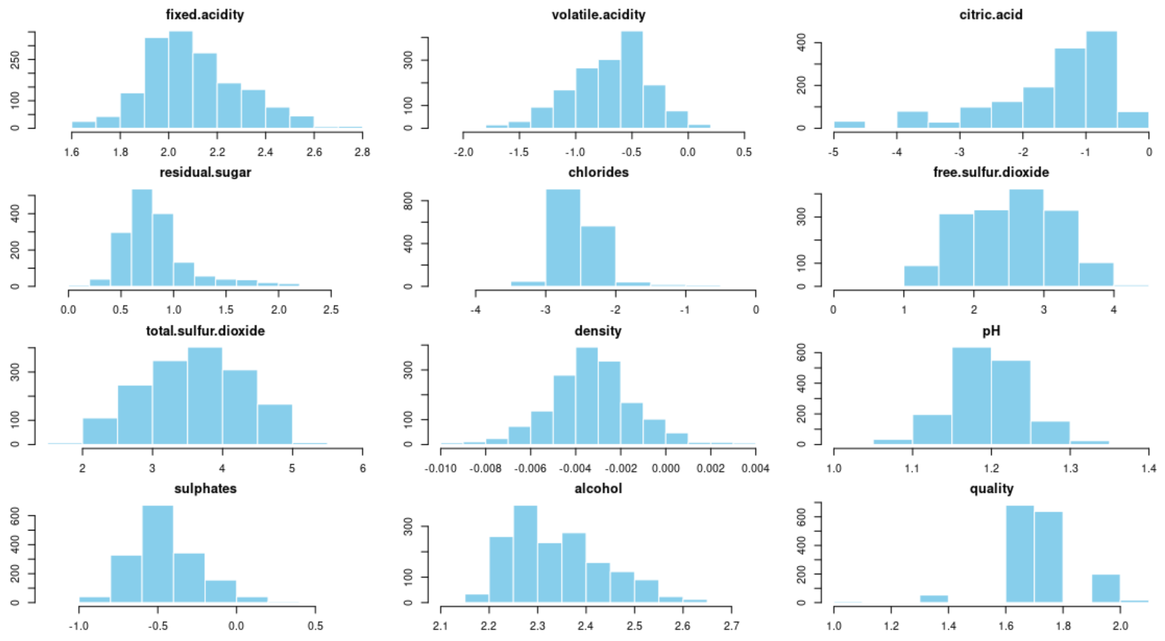
The next issue we will take a look at is the distribution of features. If all features do not have similar distribution or scale it can be difficult to use them in the same model or use them to make similar predictions about our target. Lets take a look at the distribution of all the features from the wine quality dataset. We can do this in R by looping through the columns of our dataframe and using the hist function.

```
for (col in names(df)) {
  hist(df[[col]], main = col, xlab = col, col = "skyblue", border = "white")
}
```



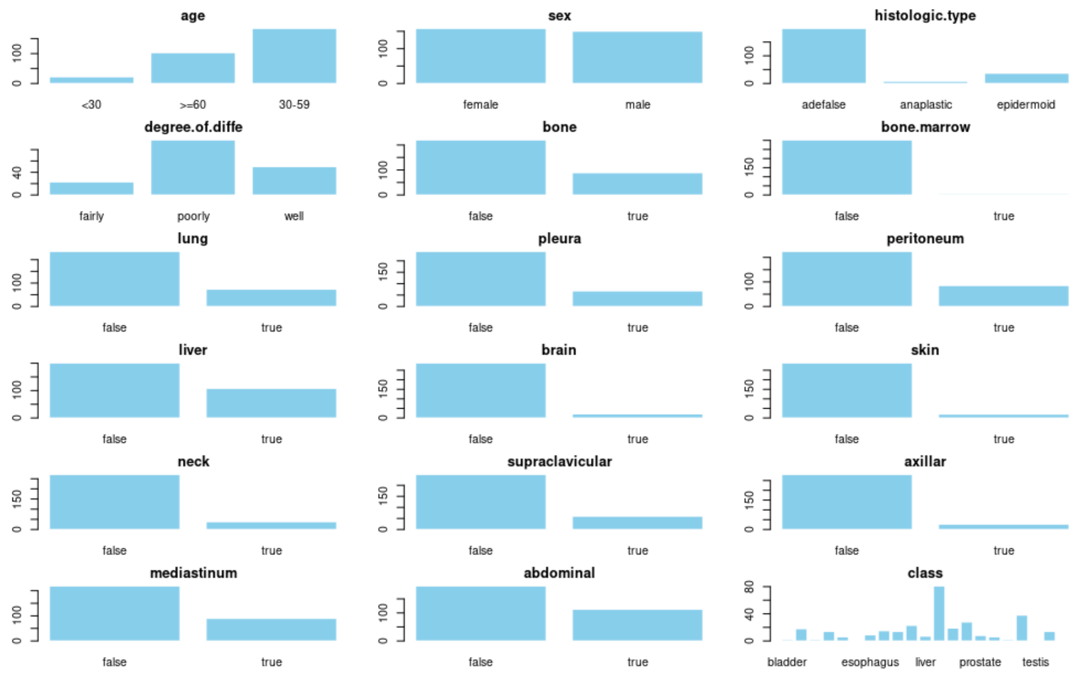
As you can see the features have some varying distributions. Whether this fact actually has any adverse effects on your model often depends on the question you are trying to answer and you may not be able to tell until the model is already built. If we do decide that the lack of uniform distribution is a bad thing we can solve it in several ways but the most common is normalizing. Normalizing data is when you perform some sort of transformation on every observation in order to fit it all in a normal distribution. If we perform this on all of our variables then they will all have a normal distribution and our lack of uniform distribution will be dealt with. We can do this in R by using the as.data.frame function to log all observations in our dataframe.

```
normaldf <- as.data.frame(log(df))
```



Now our features have a more uniform distribution. They now may all be more applicable in our model as well as easier to use to make predictions once our model is built. On the other had, this can cause more issues such as inhibiting our ability to make claims about the direct effect that a feature may have on our target since both are modified. Next, lets check out the distribution of the features for the primary tumor dataset. This dataset's features are primarily categorical variables. That means we have to use the barplot R function in place of the hist one.

```
for (col in names(df)) {
  tab <- table(df[[col]])
  barplot(tab, main = col, xlab = col, col = "skyblue", border = "white")
}
```



You may notice that this result appears quite a bit different than the result for the wine quality. This is due to the fact that the features are all categorical variables and primarily binary. As a result of this fact the distribution has less of an effect on the usability of our model. If we were to try to normalize this data it would end up changing the results of the actual data which is absolutely not what we want to do. So, in this case we will leave the data as is and we can move on to modeling or additional preprocessing.

3.3 Multicollinearity

Another issue that may be present in our data is multicollinearity. Multicollinearity is when features of our data may be highly correlated with each other. This can add redundancy to our model as well as potentially cause issues in the interpretation of predictions. For example if features A and B are highly correlated then predictions based on B may actually be a result of A. Lets take a look at our wine quality dataset and check for multicollinearity. We can generate a correlation matrix in R by using the cor function.

```
cor(df)
```

This will give us the correlation between all variables in the df. Then we need to figure our what level of correlation is high enough to be concerned about. For our purposes we will say that a correlation of +- 0.8 is significant. The good news for us is that none of the features have that level of correlation so we do not need to worry about it. Now we can do a similar check with our primary tumor dataset. Since this dataset contains categorical variables, we will not be able to use the cor function. Instead we can use the polycor library and the hetcor function.

```
library(polycor)
hetcor(df)
```

Similar to the wine quality we did not find any multicollinearity to be concerned about, but what would we do if we had. Like other issues there are several techniques for dealing with multicollinearity such as removing one of the highly correlated variables or combining them into a single variable.

Sources:

https://github.com/datasets/primary-tumor?tab=readme-ov-file

https://archive.ics.uci.edu/dataset/186/wine+quality

https://towardsdatascience.com/7-ways-to-handle-missing-values-in-machine-learning-1a6326adf79e