Detecting and Handling Outliers

One of the most crucial steps of data preprocessing is detecting and handling outliers. But, what are outliers, in the first place? Simply put, in a feature, the values that stand out of the crowd are outliers. For example: consider a list of ages of a sample of Netflix customers, [24, 34, 20, 16, 25, 30, 75, 21, 30, 15]. Clearly 75 is an outlier.

Outliers can affect the model’s performance drastically. It is important to handle the outliers carefully.

Boxplot is the best tool to visualize outliers in a dataset. Other ways are Inter-Quartile Range and Z-score.

Ways to handle outliers:

1. Remove the outliers: Before deleting the entries having outliers, make sure the they aren’t adding any significant information and also that the dataset is not too small as deleting rows from datasets with a smaller number of records would mean losing vital information.
2. Imputation with Mean/Median/Mode: Replace the outliers with any of the measures of central tendency just as we perform missing value imputation.
3. Quantile based flooring: This method is like squeezing the far-off values so that they fall within the range.

Example of quantile-based flooring:

Algorithm:

1. Calculate 25 percentile
2. Calculate 75 percentile
3. Calculate Inter-quartile range
4. Calculate upper bound
5. Calculate lower bound
6. Replace the values above upper bound with upper bound value
7. Replace the values below lower bound with lower bound value

Implementation:

def outliers(df, feature):

    '''Quantile based capping of outliers in a dataframe'''

    upper = lower = per25 = per75 = iqr = 0

    per25 = np.percentile(df[feature], 25)

    per75 = np.percentile(df[feature], 75)

    iqr = per75 - per25

    lower = per25 - (iqr\*1.5)

    upper = per75 + (iqr\*1.5)

    df[feature] = df[feature].apply(lambda x: lower if x<lower else x)

    df[feature] = df[feature].apply(lambda x: upper if x>upper else x)

    return df