Feature Scaling in Machine Learning

Feature Scaling is a technique often applied as part of data preparation for machine learning. The goal of scaling (sometimes also referred as normalization) is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information.

If one component (e.g. human height) varies less than another does (e.g. weight) because of their respective scales (meters vs. kilos), an algorithm say PCA might determine that the direction of maximal variance more closely corresponds with the 'weight' axis, if those features are not scaled. Thus leading to incorrect results.

Scaling avoids these problems by creating new values that maintain the general distribution and ratios in the source data, while keeping values within a scale applied across all numeric columns used in the model. Essentially avoiding the results being heavily influenced by the High magnitude features.

Scaling is advised in the following situations-

1. When using ***Distance or Variance Based Algorithms*** -
   Scaling is essential in Distance based (for eg. k-nearest neighbours with a Euclidean distance measure) or variance based models (eg. linear discriminant analysis, principal component analysis) if we want all the features to contribute equally in the final output.

2. ***Gradient Descent optimizations***-
   When the feature data value is large, scaling can be used to obtain better Convergence in lesser number of iterations and speed up the overall iterative process.

Types of Scaling –

The two most commonly used scaling techniques are **Z-score** and **Min-Max** Scaling.

1. **Z-Score Scaling** – A method in which all the values are converted to z-scores.

Z-score removes the mean i.e. brings the mean to zero and scales the data to unit variance. The scaling shrinks the range of the feature values, for eg. if the original range was say 500 to 50,000 with a mean of 20,000 the new range could be shrunk to -5 to +5 with mean approximately equal to 0.

Z-scores are calculated using the following formula -

$$z = \frac{\mathrm{x} - mean(x)}{stdev(x)}$$

One thing to note at this point is, the expression of z-score makes use of mean and std. deviation of the distribution, which are both affected by the presence of outlier values. This leads to an imbalanced feature scales in presence of significant outlier values. Thus being aware of the outliers and performing appropriate treatment in accordance with the business becomes very crucial.

2. **Min-Max Scaling**: The MinMax method linearly rescales every feature to the [0,1] interval. The presence of this bounded range - in contrast to z-score scaling - is that we will end up with smaller standard deviations, which can suppress the effect of outliers.

   The values in the column are obtained using the following formula:

$$z = \frac{\mathrm{x} - min(x)}{[\max(x) - min(x)]}$$

   Similar to the z-score scaling, Min-Max scaling is also affected by the outliers due to presence of Min and Max value (as Outliers would be the Min Max values in the features) in the expression. Reinforcing the need to inspect for outliers for both the scaling methods.

Choosing between Z-score standardization and Min-Max scaling:

When it comes to choosing the appropriate scaling method for the dataset, there are no hard coded rules instead; it depends on the case study in hand.

That being said, you can use the below pointers to arrive at the decision best suited for your use-case:

1. Min-Max is good to use when you know that the distribution of your data does not follow a Gaussian distribution. In addition, this can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbours and Neural Networks.
2. Z-Score on the other hand, can be helpful in cases where the data follows a Normal (or near normal) distribution or the algorithm demands a Normal distribution of the features.

3. Min-Max can be a better option when you are able to accurately estimate the minimum and maximum observable values. For example in image processing, where pixel intensities are known to have the range of 0 to 255 for the RGB colour range.

Inverse Scaling -

There might be certain scenarios, where interpretability of the features is very important. For instance, let's suppose we have trained machine learning models on the scaled data and we want to present our results/findings to the stake holders. Now, the stakeholders will be more comfortable with the original scales (units) instead of the Bounded Scaled values. Thus, inverse scaling can be used to get the values to the original scales and avoid the issues of interpretability.

Python Code snippet for scaling -

```
#Importing the necessary libraries

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler


#Splitting the dataset into Train and Test prior to scaling to avoid data leakage

#Only applicable for Supervised Learning Algorithms

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=100)


# Seperating the Continuous and Categorical variables

X_train_cont = X_train.select_dtypes(include= 'number',exclude='object')

X_train_cat = X_train.select_dtypes(include='object',exclude='number')


X_test_cont = X_test.select_dtypes(include= 'number',exclude='object')

X_test_cat = X_test.select_dtypes(include='object',exclude='number')

#Intilizing object of StandardScaler

zscore = StandardScaler()
```

```
#Fitting the StandardScaler object with the Train Set

zscore.fit(X_train_cont)

#Scalling the Train & Test sets using the fitted StandardScaler object

X_train_scaled= pd.DataFrame(zscore.transform(X_train_cont),columns =
X_train_cont.columns)

X_test_scaled= pd.DataFrame(zscore.transform(X_test_cont),columns = X_test_cont.columns)


#Merging the Categorical and Continuous datasets

X_train = X_train_scaled.merge(right= X_train_cat,left_index =True,right_index =True)

X_test = X_test_scaled.merge(right= X_test_cat,left_index =True,right_index =True)

#Model building Steps

…

# Inverse Scaling to obtain unscaled data

X_train_cont = pd.DataFrame(zscore.inverse_transform(X_train_scaled),columns =
X_train.columns)

X_test_cont = pd.DataFrame(zscore.inverse_transform(X_test_scaled),columns =
X_test.columns)
```

Substitution for Min-Max Scaling –

```
from sklearn.preprocessing import MinMaxScaler

#Intilizing object of Min-Max

minmax = MinMaxScaler()

#Fitting the MinMaxScaler object with the Train Set

minmax.fit(X_train_cont)
```

References –

1. [Scikit-learn Pre-processing Article]
2. **Data Preprocessing in Data Mining By -** ***Salvador García , Julián Luengo , Francisco Herrera***