Bonn-Aachen International Center for Information Technology

Introduction to Machine Learning Tutorial 06

Bonn-Aachen International Center for Information Technology

Data Preprocessing

Machine Learning methods **do not perform well when the input values have different scales**.

If we look at our data in heart.csv. and diabetes.csv, we see that the data is differently scaled.

- KNN
- SVM
- K Means

Are three ML methods that are sensitive to scaling.

Feature Scaling

Then let's apply scaling to our data! We are going to scale our features (columns) and this is why this step I also called feature scaling.

One can distinguish:

- Normalisation
- Standardisation

Normalisation

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

where x is an original value, x‘ is the normalized value

This is also termed Minimax normalisation.

Normalisation

Sklearn provides predefined code for this. It is the **`MinMaxScaler`**.

The is termed a transformer as it is able to transform (here: normalize data).

The input data is transformed to values between 0 and 1.

This transformer has a **`feature_range`** hyperparameter that allows you to change if you do not want the range from 0 to 1 for some reason.

Bonn-Aachen International Center for Information Technology

Standardisation

Standardisation is quite different:

$$x' = \frac{x - \bar{x}}{\sigma}$$

Where x is the original feature vector, x-bar is the mean of that feature vector, and σ is its standard deviation.

Normalisation v Standardisation

The following table demonstrate the difference between the two feature scaling, standardization and normalization on a sample dataset from 0 to 5:

input	standardized	normalized
0.0	-1.336306	0.0
1.0	-0.801784	0.2
2.0	-0.267261	0.4
3.0	0.267261	0.6
4.0	0.801784	0.8
5.0	1.336306	1.0

Bonn-Aachen International Center for Information Technology

Normalisation v Standardisation

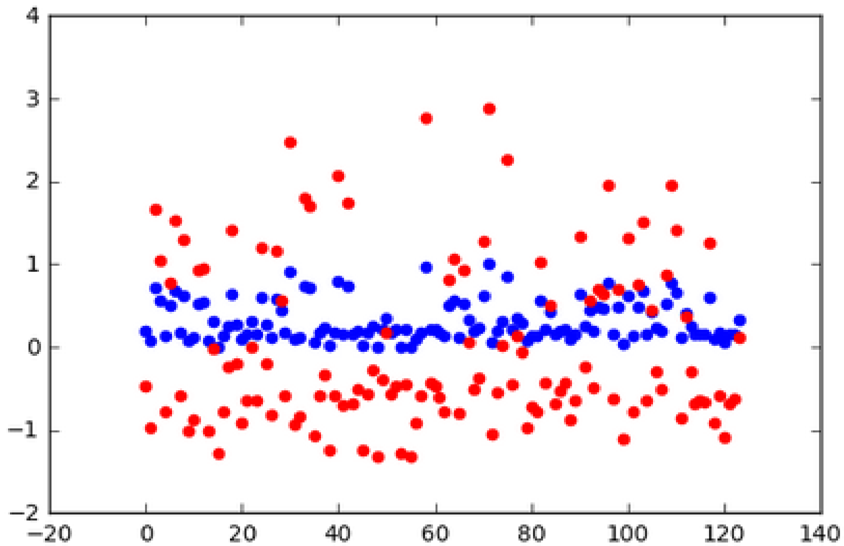
The following table demonstrate the difference between the two feature scaling, standardization and normalization on a sample dataset from 0 to 5:

```
xx = np.arange(len(X_train_std))
yy1 = X_train_norm[:,1]
yy2 = X_train_std[:,1]
scatter(xx, yy1, color='b')
scatter(xx, yy2, color='r')
```

```
<matplotlib.collections.PathCollection at 0x7f7456ea2f50>
```

Blue: normalised data

Red: standardised data



Standardisation

We are therefore going to apply standardisation to our ML algorithm SVM and kNN.

```
from sklearn.preprocessing import StandardScaler
x_sc= StandardScaler()

features_train_scaled = x_sc.fit_transform(features_train)
features_test_scaled = x_sc.transform(features_test)
```

Implementation

Create the test and training set first:

```
# Create training and test set
from sklearn.model_selection import train_test_split
features_train, features_test, labels_train, labels_test =
train_test_split(features, labels, test_size=0.25)
```

then implement the StandardScaler:

```
# Import the StandardScaler
from sklearn.preprocessing import StandardScaler
x_sc= StandardScaler()

features_train_scaled = x_sc.fit_transform(features_train)
features_test_scaled = x_sc.transform(features_test)
```

Implementation

`x_sc.fit_transform`

Contains two methods:

1. Fit()
2. Transform()

Instead of calling **`fit()`** first and then **`transform()`** one can code **`fit_transform()`.**

Notice: `Fit_transform()` is only applied to the training data. The **transform method** is only applied to the test data.

Implementation

`x_sc.fit_transform`

Contains two methods:

1. Fit()
2. Transform()

Instead of calling **`fit()`** first and then **`transform()`** one can code **`fit_transform()`**.

Notice: `Fit_transform()` is only applied to the training data. The **transform method** is only applied to the test data.

Bonn-Aachen
International Center for
Information Technology

Implementation

Here the **fit** method, when applied to the training dataset, learns the model parameters (for example, mean and standard deviation).

We then need to apply the **transform** method on the training dataset to get the transformed (scaled) training dataset.

We could also perform both of this steps in one step by applying **fit_transform** on the training dataset.

We do not want to apply fit on the test set as the test set should be an unknowm data set with unknown distributions. We want to test the ML ability to generalize (on unknown/unseen data).