Working with feature Reduction using PCA

Sometimes you may be force to use all feature columns and you need to get a great model of it.

Let’s assume that you tried all methods and you are not getting the required accuracy. In that case, we recommend you use feature reduction.

**PCA(Principal Component Analysis):**

Let’s say we have our featureset, we pass this featureset and label to our PCA algorithm.

PCA returns something called as principalComponents. This principalComponents access features which can be given to our model(whatever algorithms you are choosing) to get the FinalTrainedModel.

Featureset ==🡺PCA ====🡺 principalComponents ===🡺 modelAlgo ===🡺 FinalTrainedModel

Label ========================================🡺

When you are working on feature selection models. In feature selection you may introduce some algorithms this between.

But when you use feature reduction methods, during development you need to use even PCA object.

principalComponents is ==🡺 eigen vector representation of your original feature set.

(important patterns that can make the algorithm understand the population)

Using PCA we are extracting important information out of featureset. The information that can help the model understand the pattern.

When it comes to applying StandardScaling on feature set, there are three areas where this step is mandatory:

1. When performing Feature Reduction (PCA)
2. When using Neural Network (ANN)
3. When you use Genetic algorithm for optimizations

PCA Guidelines:

1. Standardization(StandardScaler) is mandatory.

2. Hyperparameter for PCA (n\_components <= no of features)

3. n\_components can be judged using:

count based on variance(component value should be greater than or equal to 75%)

In a step of PCA algorithm, we need to find number of component, after having the array from:

#Identify the ideal n\_components

principalComponents.explained\_variance\_ratio\_

the result array:

array([0.72770452, 0.23030523, 0.03683832, 0.00515193])

1. (method one) n\_components will be : the number of components greater than equal 0.75.

Here because none of them are >= 0.75, we consider n\_components = 1.

If the array was like this: [0.82, 0.76, 0.72, 0.5] , n\_components = 2.

1. Bruteforcing, try all 1..n components where n is number of features. For example if the dataset has 10 features, you will create 10 models and whichever model gives you best model accuracy, you will choose that as a final one.(for loops)