Conformal Prediction Functions

Inductive Conformal Prediction (ICP)

Algorithm 1 Compute ICP Conformity Scores

function ComputeICP(X)
    $mean\_val \leftarrow \text{mean}(X, \text{axis} = 0)$
    return $\text{norm}(X - mean\_val, \text{axis} = 1)$
end function

Mondrian Conformal Prediction

Algorithm 2 Compute Mondrian Conformity Scores

function ComputeMondrian($X, labels$)
    $unique\_classes \leftarrow \text{unique}(labels)$
    $scores \leftarrow \text{zeros}(labels.\text{shape}[0])$
    for all $cls \in unique\_classes$ **do**
        $class\_data \leftarrow X[labels == cls]$
        $mean\_val \leftarrow \text{mean}(class\_data, \text{axis} = 0)$
        $scores[labels == cls] \leftarrow \text{norm}(class\_data - mean\_val, \text{axis} = 1)$
    end for
    return $scores$
end function

Cross-Conformal Prediction

Algorithm 3 Compute Cross-Conformal Scores

function ComputeCrossConformal($X, labels, k = 5$)
    $folds \leftarrow \text{array\_split}(\text{arange}(len(X)), k)$
    $all\_scores \leftarrow []$
    for $i = 0$ **to** $k - 1$ **do**
        $train\_idx \leftarrow \bigcup_{j \neq i} folds[j]$
        $calib\_idx \leftarrow folds[i]$
        $mean\_val \leftarrow \text{mean}(X[train\_idx], \text{axis} = 0)$
        $scores \leftarrow \text{norm}(X[calib\_idx] - mean\_val, \text{axis} = 1)$
        $all\_scores.\text{append}(scores)$
    end for
    return $\text{concatenate}(all\_scores)$
end function

Venn-Abers Predictors

Algorithm 4 Compute Venn-Abers Scores

function ComputeVennAbers($X, labels$)
    $model \leftarrow$ IsotonicRegression()
    $model$.fit($X, labels$)
    return $model$.predict(X)
end function

Compute Conformal Intervals

Algorithm 5 Compute Conformal Prediction Intervals

function ComputeConformalIntervals($synthetic\_data, \alpha$)
    $intervals \leftarrow []$
    for all $sample \in synthetic\_data$ **do**
        $lower\_bound \leftarrow$ percentile($sample, \alpha \times 100$)
        $upper\_bound \leftarrow$ percentile($sample, (1 - \alpha) \times 100$)
        $intervals$.append(($lower\_bound, upper\_bound$))
    end for
    return $intervals$
end function