ALGORITMO KTREE COM SLIDING WINDOW

KTE-SW

Algorithm KTE-SW // *Supervised **K**Trees with Concept-Drift Adapting **E**ntropy-Based from stream **S**liding **W**indow-Based*

Input: ensemble of CART → **E**; window → **W(X)**; data → **X**; label → **y**; n_k_trees → **k**

Fit(E, k, Wi(X), y) → Output- None:

1. Calculate entropy for the window W(X) → H(Wi(X))
2. Choose k trees from E → ktree(E, X_tray, y_train)

Predict ($W_{(i+1)}$(X)) → Output- y_label_predict: // *Pre-quential*

1. For each Xi in $W_{(i+1)}$ (X)
   1.1 Predict with majority voting ktree → y_label_predict
2. Calculate entropy for the window $W_{(i+1)}$(X) → H($W_{(i+1)}$(X))
   2.1 Compare H(Wi(X)) with H($W_{(i+1)}$(X)) using Kullback-Leibler → KLB(H(Wi(X), H($W_{(i+1)}$(X)))
   2.2 If KLB(H(Wi(X), H($W_{(i+1)}$(X))) ≈ 1 then
      2.2.1 **retraining ktree**
      2.2.2 update H(Wi(X)) = H($W_{(i+1)}$(X))
3. Return y_label_predict

```
method Fit(E, k, Wi(X), y) → Output- None:
    1. Calculate entropy for the window Wi(X) → H(Wi(X))
      1.1 For each Wi(X):
       entropy_per_feature[n_feature] = H(Wi(X)) = Σ_{j=1}^{n_feature} p(x_j) * log_e(1/p(x_j))
      1.2 For each individual_tree(E):
       feature_importance[n_estimator, n_features] = estimator(E)
      1.3 change_factor = feature_importance x entropy_per_feature
      1.4 set KTE-SW.H(Wi(X))_ = change_factor
     2. Choose k trees from E → ktree(E, k, X_tray, y_train)
      2.1 For each individual_tree(E):
       y_pred[n_estimator, predict] = predict(E, Wi(X))
      2.2 for each individual_tree(E):
       list_evaluate[n_estimator, metric] = compute_metric(y_true, y_pred(E))
      2.3 sort_evaluate[n_estimator, metric] = sort_descending(list_evaluate)
      2.4 set KTE-SW.estimators_ = choose_k_tree(k, sort_evaluate)
```

$$H(\text{Wi}(X)) = \sum_{j=1}^{n\_feature} p(x_j) * log_e\left(\frac{1}{p(x_j)}\right)$$

```
method Predict (W(i+1)(X)) → Output- y_label_predict: // Pre-quential
  1. For each Xi in W(i+1) (X)
     1.1 For each individual_tree(KTE-SW.estimators_):
         y_pred[n_estimator, predict] = predict(E, W(i+1)(X))
     1.2 y_label_predict[n_instance] = majority_voting_scheme(y_pred)
  2. Calculate entropy for the window W(i+1) (X) → H(W(i+1) (X))
     2.1 Compare H(Wi(X)) with H(W(i+1) (X)) using Kullback-Leibler → KLB(H(Wi(X), H(W(i+1) (X)))
        2.1.1 get KTE-SW.H(Wi(X))_
        2.1.2 For each W(i+1)(X):
```

$$\text{entropy\_per\_feature[n\_feature]} = H(W_{(i+1)}(X)) = \sum_{j=1}^{n\_feature} p(x_j) * log_e\left(\frac{1}{p(x_j)}\right)$$

```
        2.1.3 For each individual_tree(KTE-SW.estimators_):
              feature_importance[n_estimator, n_features] = estimator(KTE-SW.estimators_)
        2.1.4 current_change_factor = feature_importance x entropy_per_feature
        2.1.5 Compare KTE-SW.H(Wi(X))_ with current_change_factor
```

$$D_{kl} = KLB = \sum_i p(i) * \log_e \frac{p(i)}{q(i)} = \sum_i \text{KTE-SW.H(Wi(X))\_} * \log_e \frac{\text{KTE-SW.H(Wi(X))\_}}{\text{current\_change\_factor}}$$

```
     2.2 If KLB(KTE-SW.H(Wi(X))_, current_change_factor) ≈ 1 then
        2.2.1 retraining ktree
        2.2.2 update H(Wi(X)) = H(W(i+1) (X))
  3. Return y_label_predict
```