```
# Pseudo-Python code for calculating threshold P_jr

# Inputs:
# X_class : list of M vectors belonging to a class C
# beta : confidence probability (e.g., 0.95)
# n : number of attributes per vector

def calculate_cluss_threshold(X_class, beta):
    M = len(X_class)  # number of vectors in class

    # Step 1: Compute centroid C_d
    C_d = [sum(X_class[k][i] for k in range(M)) / M for i in range(n)]

    # Step 2: Compute distances d_k between vectors and centroid
    d_k = [sqrt(sum((X_class[k][i] - C_d[i])**2 for i in range(n))) for k in range(M)]

    # Step 3: Compute mean distance
    d_mean = sum(d_k) / M

    # Step 4: Compute sample standard deviation
    sigma_d = sqrt(sum((dk - d_mean)**2 for dk in d_k) / (M - 1))

    # Step 5: Confidence probability beta is given as input

    # Step 6: Compute confidence interval
    from scipy.stats import t
    if M > 10:
        # Normal distribution approximation
        t_beta = t.ppf((1 + beta)/2, df=M-1)  # two-tailed
    else:
        # Student's t-distribution
        t_beta = t.ppf((1 + beta)/2, df=M-1)

    margin = sigma_d / sqrt(M) * t_beta
    I_beta = (d_mean - margin, d_mean + margin)

    # Step 7: Threshold is right bound of confidence interval
    P_jr = I_beta[1]

    return P_jr
```