1 Import Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats.mstats import winsorize
import seaborn as sns
import pickle
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split, StratifiedKFold
import random
from models.GeneralizedLearningVectorQuantization import GeneralizedLearningVectorQuantizat
from models.OutliersHandling import OutliersHandling
```

In [2]:

```
random_state = 45
random.seed(random_state)
```

2 Read Dataset

Import data latih yang sudah dibagi

In [3]:

```
dataset = pickle.load(open("../notebook/results/dataset.pkl",'rb'))
training = dataset['training']['data']
testing = dataset['testing']['data']
features = dataset['features_name']
n_rows, n_cols = training.shape
```

3 Outliers Analysis

3.1 Trimming Method

3.1.1 Find outliers index in data using IQR (threshold 1.5 & 3)

In [4]:

```
outliers_index_iqr_3 = OutliersHandling().find_outliers_iqr(training, n_cols)
outliers_index_iqr_15 = OutliersHandling().find_outliers_iqr(training, n_cols, threshold=1.
```

3.1.2 Delete data outliers in training

In [5]:

```
training_trim_3 = np.delete(training, outliers_index_iqr_3, axis=0)
training_trim_15 = np.delete(training, outliers_index_iqr_15, axis=0)
```

3.2 Winsorizing Method

In [6]:

```
training_win_5 = OutliersHandling().winsorizing(training,n_cols,5)
training_win_10 = OutliersHandling().winsorizing(training,n_cols,10)
training_win_20 = OutliersHandling().winsorizing(training,n_cols,20)
```

3.3 Store all data after outliers handling process

In [7]:

```
inputs_data = {
#     'Original data without outliers handling': training,
    'Data with Trimming Outliers (IQR threshold=1.5)': training_trim_15,
    'Data with Trimming Outliers (IQR threshold=3)': training_trim_3,
    'Data with Winsorizing Outliers (threshold = 5)': training_win_5,
    'Data with Winsorizing Outliers (threshold = 10)': training_win_10,
    'Data with Winsorizing Outliers (threshold = 20)': training_win_20,
}
```

4 Main Process

Tuning Optimal GLVQ Parameter First Using Data Original For Feature Selection Purpose

1. Input original data
2. Transformasi
3. K-fold cross validation (k=5) for tuning parameters
4. Output optimal GLVQ parameter

Main Pipeline

1. Choose input data that will be analyze
2. Train validate split
3. Transformation
4. Feature reduction using PCA
5. Stratified K-Fold Cross-validation the model
6. Train Test Model
7. Output the results

4.1 Tuning Optimal GLVQ Parameter First

4.1.1 Split X and Y in training set original (without outliers handling)

In [8]:

```
X_train_original, y_train_original = training[:,0:-1], training[:,-1]
```

In [9]:

```
X_train_original.shape, y_train_original.shape
```

Out[9]:

```
((455, 30), (455,))
```

4.1.2 Data Standardization using StandardScaler

In [10]:

```
scaler = StandardScaler()
X_train_original = scaler.fit_transform(X_train_original)
```

4.1.3 Tune GLVQ Parameter using 5-fold cross-validation

In [11]:

```
n_splits = 5
strkfold = StratifiedKFold(n_splits=n_splits, shuffle=True,random_state=random_state)
```

In [12]:

```
codebooks = [1,2,3,4,5]
alphas = [round(i, 2) for i in np.arange(0.1, 1, 0.1)]
max_epochs = [100]
min_errors = [0.000001]
cross_val_results = list()
for codebook in codebooks:
    for alpha in alphas:
        for max_epoch in max_epochs:
            for min_error in min_errors:
                accuracy_score_list_per_combination = list()
                combination_name = "Codebook--"+str(codebook)+"_Alpha--"+str(alpha)+"_MaxEp
                accuracy_score_list_per_combination.append(combination_name)
                sum_acc = 0
                for train_index, validation_index in strkfold.split(X=X_train_original, y=y
                    glvq = GLVQ(
                        alpha=alpha,
                        max_epoch=max_epoch,
                        min_error=min_error,
                        n_codebooks=codebook
                    )
                    glvq.fit(X_train_original[train_index], y_train_original[train_index])
                    y_pred_val_glvq = glvq.predict(X_train_original[validation_index])
                    acc = metrics.accuracy_score(y_train_original[validation_index], y_pred
                    sum_acc += acc
                    accuracy_score_list_per_combination.append(acc)
                mean_accuracy_cross_validation = sum_acc/n_splits
                accuracy_score_list_per_combination.append(mean_accuracy_cross_validation)
                print(combination_name, mean_accuracy_cross_validation)
                cross_val_results.append(accuracy_score_list_per_combination)
```

```
Codebook--1_Alpha--0.1_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--1_Alpha--0.2_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--1_Alpha--0.3_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--1_Alpha--0.4_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--1_Alpha--0.5_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--1_Alpha--0.6_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--1_Alpha--0.7_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--1_Alpha--0.8_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--1_Alpha--0.9_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--2_Alpha--0.1_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--2_Alpha--0.2_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--2_Alpha--0.3_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--2_Alpha--0.4_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--2_Alpha--0.5_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--2_Alpha--0.6_MaxEpoch--100_MinError--1e-06 0.9406593406593406
Codebook--2_Alpha--0.7_MaxEpoch--100_MinError--1e-06 0.9406593406593406
Codebook--2_Alpha--0.8_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--2_Alpha--0.9_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--3_Alpha--0.1_MaxEpoch--100_MinError--1e-06 0.9472527472527472
Codebook--3_Alpha--0.2_MaxEpoch--100_MinError--1e-06 0.9472527472527472
Codebook--3_Alpha--0.3_MaxEpoch--100_MinError--1e-06 0.9472527472527472
Codebook--3_Alpha--0.4_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--3_Alpha--0.5_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--3_Alpha--0.6_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--3_Alpha--0.7_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--3_Alpha--0.8_MaxEpoch--100_MinError--1e-06 0.9472527472527472
Codebook--3_Alpha--0.9_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--4_Alpha--0.1_MaxEpoch--100_MinError--1e-06 0.945054945054945
```

```
Codebook--4_Alpha--0.2_MaxEpoch--100_MinError--1e-06 0.9406593406593406
Codebook--4_Alpha--0.3_MaxEpoch--100_MinError--1e-06 0.9472527472527472
Codebook--4_Alpha--0.4_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--4_Alpha--0.5_MaxEpoch--100_MinError--1e-06 0.9516483516483516
Codebook--4_Alpha--0.6_MaxEpoch--100_MinError--1e-06 0.9516483516483516
Codebook--4_Alpha--0.7_MaxEpoch--100_MinError--1e-06 0.9494505494505494
Codebook--4_Alpha--0.8_MaxEpoch--100_MinError--1e-06 0.9472527472527472
Codebook--4_Alpha--0.9_MaxEpoch--100_MinError--1e-06 0.9428571428571428
Codebook--5_Alpha--0.1_MaxEpoch--100_MinError--1e-06 0.9384615384615385
Codebook--5_Alpha--0.2_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--5_Alpha--0.3_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--5_Alpha--0.4_MaxEpoch--100_MinError--1e-06 0.9406593406593406
Codebook--5_Alpha--0.5_MaxEpoch--100_MinError--1e-06 0.9472527472527472
Codebook--5_Alpha--0.6_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--5_Alpha--0.7_MaxEpoch--100_MinError--1e-06 0.9406593406593406
Codebook--5_Alpha--0.8_MaxEpoch--100_MinError--1e-06 0.945054945054945
Codebook--5_Alpha--0.9_MaxEpoch--100_MinError--1e-06 0.9472527472527472
```

In [13]:

```
separator_parameter = "_"
separator_value = "--"
columns_name = ['combination_name'] + ["Fold-"+str(i+1) for i in range(n_splits)] + ['mean_

glvq_tuning_parameters_results = pd.DataFrame(data=cross_val_results, columns=columns_name)
glvq_tuning_parameters_results['codebook'] = glvq_tuning_parameters_results.loc[:,'combinat
glvq_tuning_parameters_results['alpha'] = glvq_tuning_parameters_results.loc[:,'combination
glvq_tuning_parameters_results['max_epoch'] = glvq_tuning_parameters_results.loc[:,'combina
glvq_tuning_parameters_results['min_error'] = glvq_tuning_parameters_results.loc[:,'combina
```

In [14]:

```
# save GLVQ tuning parameter results from original data
glvq_tuning_parameters_results.to_csv("informations/glvq_tuning_results_train_original.csv"
```

4.1.3.1 Find GLVQ optimal parameter from tuning parameter results

Find the combination that provide the maximum mean accuracy for 5-fold

In [23]:

```
best_glvq_results = glvq_tuning_parameters_results[
    (glvq_tuning_parameters_results['mean_accuracy']==glvq_tuning_parameters_results['mean_
].reset_index(drop=True)
p=0
optimal_codebook = int(best_glvq_results.loc[p,'codebook'])
optimal_alpha = float(best_glvq_results.loc[p,'alpha'])
optimal_max_epoch = int(best_glvq_results.loc[p,'max_epoch'])
optimal_min_error = float(best_glvq_results.loc[p,'min_error'])
```

In [25]:

```
best_glvq_results
```

Out[25]:

	combination_name	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5	mean_accuracy	codeb
0	Codebook--4_Alpha--0.5_MaxEpoch--100_MinError-...	0.945055	0.956044	0.967033	0.934066	0.956044	0.951648	
1	Codebook--4_Alpha--0.6_MaxEpoch--100_MinError-...	0.934066	0.967033	0.978022	0.934066	0.945055	0.951648	

4.2 Main pipeline

In [26]:

```
hasilPengolahanData = dict()
```

In [27]:

```
for label, data in inputs_data.items():
    hasilPengolahanData[label] = dict()

    # 2- train validate split
    X, y = data[:, 0:-1], data[:, -1]
    X_train, X_val, y_train, y_val = train_test_split(
        X, y, test_size=0.2, random_state=random_state, stratify=y
    )

    # 3- standardization
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_val = scaler.transform(X_val)

    # 4- feature reduction using PCA
    pca_ = PCA()
    pca_.fit(X_train)
    threshold_cumsum = 0.8
    cumsum_eigenvalue_ratio = pca_.explained_variance_ratio_.cumsum()
    n_components = len(cumsum_eigenvalue_ratio)
    best_pca_component = 0
    for i in range(n_components):
        if cumsum_eigenvalue_ratio[i] >= threshold_cumsum:
            # index start from 0
            best_pca_component = i+1
            break
    pca_scaler = PCA(n_components=best_pca_component)
    pca_scaler.fit(X_train)
    X_train = pca_scaler.transform(X_train)
    X_val = pca_scaler.transform(X_val)
    hasilPengolahanData[label]['feature_reduction'] = {
        'PCA': pca_scaler,
        'threshold_cumsum': threshold_cumsum,
        'best_component': best_pca_component,
    }

    # 5- K-Fold Cross-validation
    strkfold = StratifiedKFold(n_splits=n_splits, shuffle=True,random_state=random_state)
    sum_acc = 0
    hasilPengolahanData[label]['Kfold_results'] = dict()
    for idx_fold, (train_index, validation_index) in enumerate(strkfold.split(X=X_train, y=
        glvq = GLVQ(
            alpha=optimal_alpha,
            max_epoch=optimal_max_epoch,
            min_error=optimal_min_error,
            n_codebooks=optimal_codebook
        )
        glvq.fit(X_train[train_index], y_train[train_index])
        y_pred_val_glvq = glvq.predict(X_train[validation_index])
        acc = metrics.accuracy_score(
            y_train[validation_index], y_pred_val_glvq)
        hasilPengolahanData[label]['Kfold_results']['Fold'+str(idx_fold)] = acc
        sum_acc += acc
    mean_accuracy_cross_validation = sum_acc/n_splits
    hasilPengolahanData[label]['Kfold_results']['mean_accuracy'] = mean_accuracy_cross_vali

    # 6 - Train test model
    glvq = GLVQ(
        alpha=optimal_alpha,
```

```
        max_epoch=optimal_max_epoch,
        min_error=optimal_min_error,
        n_codebooks=optimal_codebook
    )
    glvq.fit(X_train, y_train)
    y_pred_val = glvq.predict(X_val)
    y_pred_train = glvq.predict(X_train)
    hasilPengolahanData[label]['training'] = {
        'model': glvq,
        'y_pred': y_pred_train,
        'accuracy': metrics.accuracy_score(y_pred_train,y_train)
    }
    hasilPengolahanData[label]['validation'] = {
        'y_pred': y_pred_val,
        'accuracy': metrics.accuracy_score(y_pred_val,y_val)
    }

    print(label, ".............Done")
```

```
Data with Trimming Outliers (IQR threshold=1.5) .............Done
Data with Trimming Outliers (IQR threshold=3) .............Done
Data with Winsorizing Outliers (threshold = 5) .............Done
Data with Winsorizing Outliers (threshold = 10) .............Done
Data with Winsorizing Outliers (threshold = 20) .............Done
```

4.2.1 Save the processing results

In [28]:

```
filename = 'informations/hasilPengolahanData.pkl'
pickle.dump(hasilPengolahanData, open(filename, 'wb'))
```

5 Results analysis

In [29]:

```
df_data = {
    'labels': list(),
    'Fold0': list(),
    'Fold1': list(),
    'Fold2': list(),
    'Fold3': list(),
    'Fold4': list(),
    'mean_accuracy': list(),
    'validation_accuracy': list(),
    'training_accuracy': list(),
}
for label, results in hasilPengolahanData.items():
    df_data['labels'].append(label)
    for fold, acc in results['Kfold_results'].items():
        df_data[fold].append(acc)
    df_data['training_accuracy'].append(results['training']['accuracy'])
    df_data['validation_accuracy'].append(results['validation']['accuracy'])
outliers_analysis_results = pd.DataFrame(df_data)
```

In [30]:

```
outliers_analysis_results
```

Out[30]:

	labels	Fold0	Fold1	Fold2	Fold3	Fold4	mean_accuracy	validation_a
0	Data with Trimming Outliers (IQR threshold=1.5)	0.980769	0.901961	0.901961	0.941176	0.960784	0.937330	
1	Data with Trimming Outliers (IQR threshold=3)	0.954545	0.939394	0.923077	0.969231	0.969231	0.951096	
2	Data with Winsorizing Outliers (threshold = 5)	0.972603	0.958904	0.986301	0.958904	0.916667	0.958676	
3	Data with Winsorizing Outliers (threshold = 10)	0.958904	0.958904	0.986301	0.931507	0.916667	0.950457	
4	Data with Winsorizing Outliers (threshold = 20)	0.945205	0.958904	0.986301	0.931507	0.930556	0.950495	

5.1 Save outliers analysis results

In [31]:

```
outliers_analysis_results.to_csv('informations/outliers_analysis_results.csv', index=False)
```

In [ ]:

```

```