# rsr

February 17, 2019

# Road Sign Recognition system

```
In [89]: import pathlib as path
         import skimage as skimg
         import matplotlib.pyplot as plt
         from functional import seq
         from random import sample, seed
         import numpy as np
         from rsr import Data
         from loguru import logger
         import Augmentor
         from sklearn.manifold import TSNE
         from sklearn.decomposition import PCA
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.model_selection import cross_val_score
         from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import StandardScaler
         from sklearn.neural_network import MLPClassifier
         from sklearn.metrics import confusion_matrix
         from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
         from sklearn import svm as SVM
         from mpl_toolkits import mplot3d
         from sklearn.metrics import classification_report

         logger.remove()
```

# Loading dataset Load the data from the `/cropped/` directory, convert the directory name to integer and used them as labels

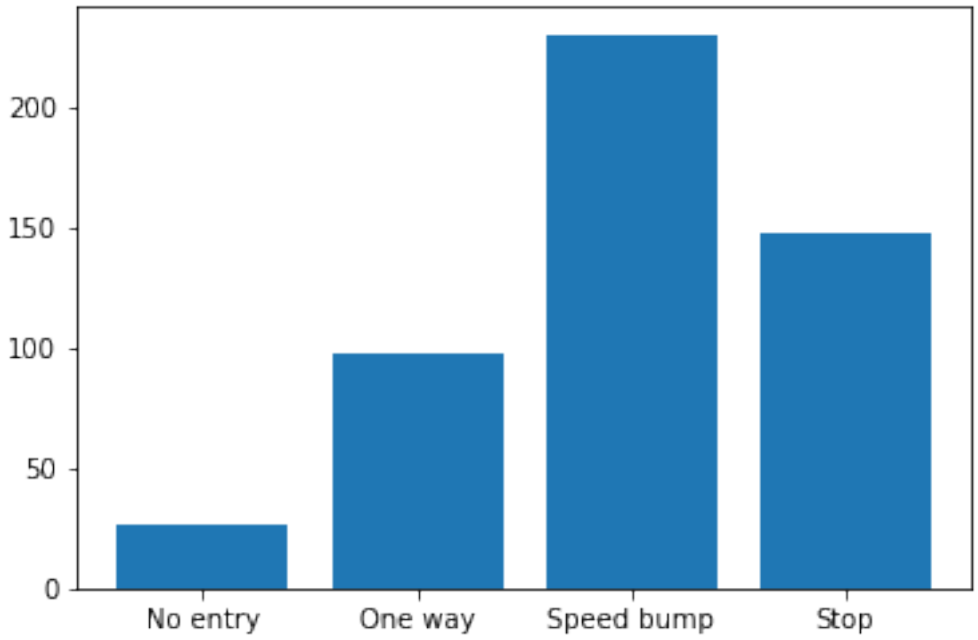
```
In [12]: def loadDataset():
             paths = path.Path('.').glob('./cropped/**/*.jpg')
             dataset = []
             for p in paths:
                 dataset.append(Data(p))
             return dataset

         def groupDataset(dataset):
             groups = seq(dataset).group_by(lambda data: data.label)
```

```
            return groups

        dataset = loadDataset()
        groups = groupDataset(dataset)
```

# Dataset distribution

```
In [13]: def printStat(dataset, groups):
            for group in groups:
                print(f'{Data.fromIndex(group[0])}: {len(group[1])}')
            print(f'Total: {len(dataset)}')

            plt.bar(seq(groups).select(lambda group: Data.fromIndex(group[0])).to_list(), seq
            plt.show()

        printStat(dataset, groups)
```

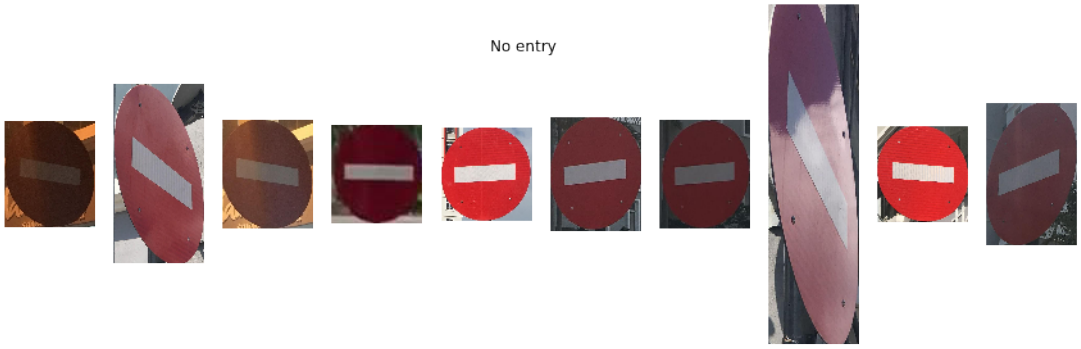
```
No entry: 26
One way: 97
Speed bump: 230
Stop: 147
Total: 500
```

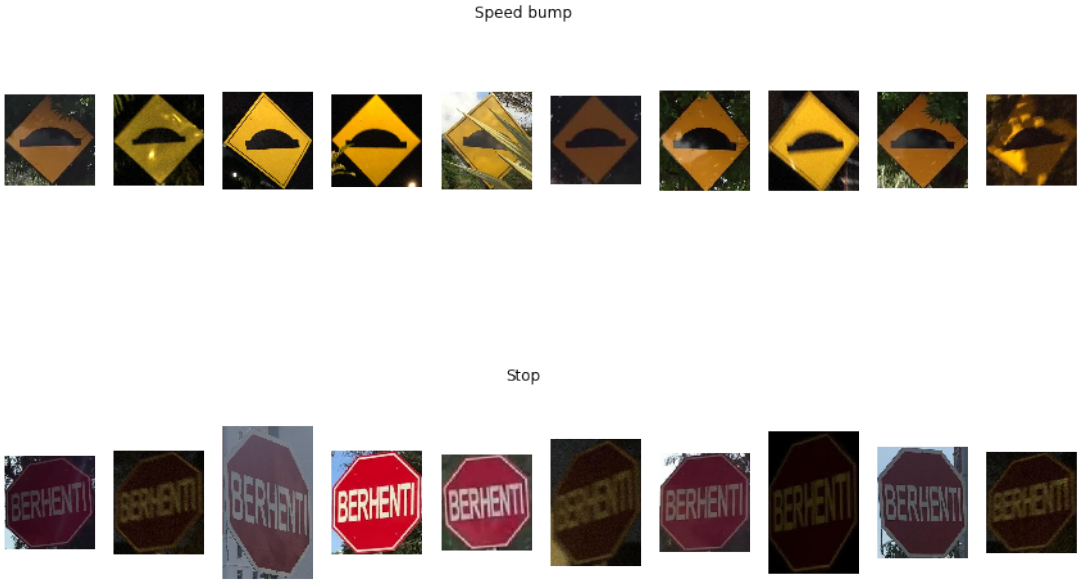



# Visualizing dataset Displaying 10 random samples for each label
*Here, the random seed is set to 0*

In [14]:
```
seed(0)

def showImages(images, title, cols = 10, figsize = (15, 15), y = 0.6):
    rows = len(images) // cols
    fig, axes = plt.subplots(rows, cols, figsize=figsize)
    cmap = None
    for axe, image in zip(axes.ravel(), images):
        axe.axis('off')
        if len(image.shape) < 3 or image.shape[-1] < 3:
            cmap = "gray"
        axe.imshow(image, cmap = cmap)

    fig.suptitle(title, y=y)
    plt.show()
    return

def showDataset(dataset, title, cols = 10, figsize = (15, 15), y = 0.6):
    showImages(seq(dataset).select(lambda data: data.image).to_list(), title, cols, fi

def sampleDataset(dataset, count=10):
    groups = groupDataset(dataset)
    for group in groups:
        showDataset(sample(group[1], count), Data.fromIndex(group[0]), cols=count)
    return
```

In [15]:
```
sampleDataset(dataset)
```




# Image preprocessing

```
In [16]: def minMaxAverage(title, items):
             print(f'Minimum {title}: {seq(items).min()}')
             print(f'Maximum {title}: {seq(items).max()}')
             print(f'Average {title}: {seq(items).average()}')
```

Average width

```
In [17]: minMaxAverage("width", seq(dataset).select(lambda data: data.image.shape[0]))
```

```
Minimum width: 74
Maximum width: 1367
Average width: 457.67
```

Average height

```
In [18]: minMaxAverage("height", seq(dataset).select(lambda data: data.image.shape[1]))
```

```
Minimum height: 65
Maximum height: 2527
Average height: 464.48
```

# Validation set Here, we only sample 100 images for testing out the preprocessing steps

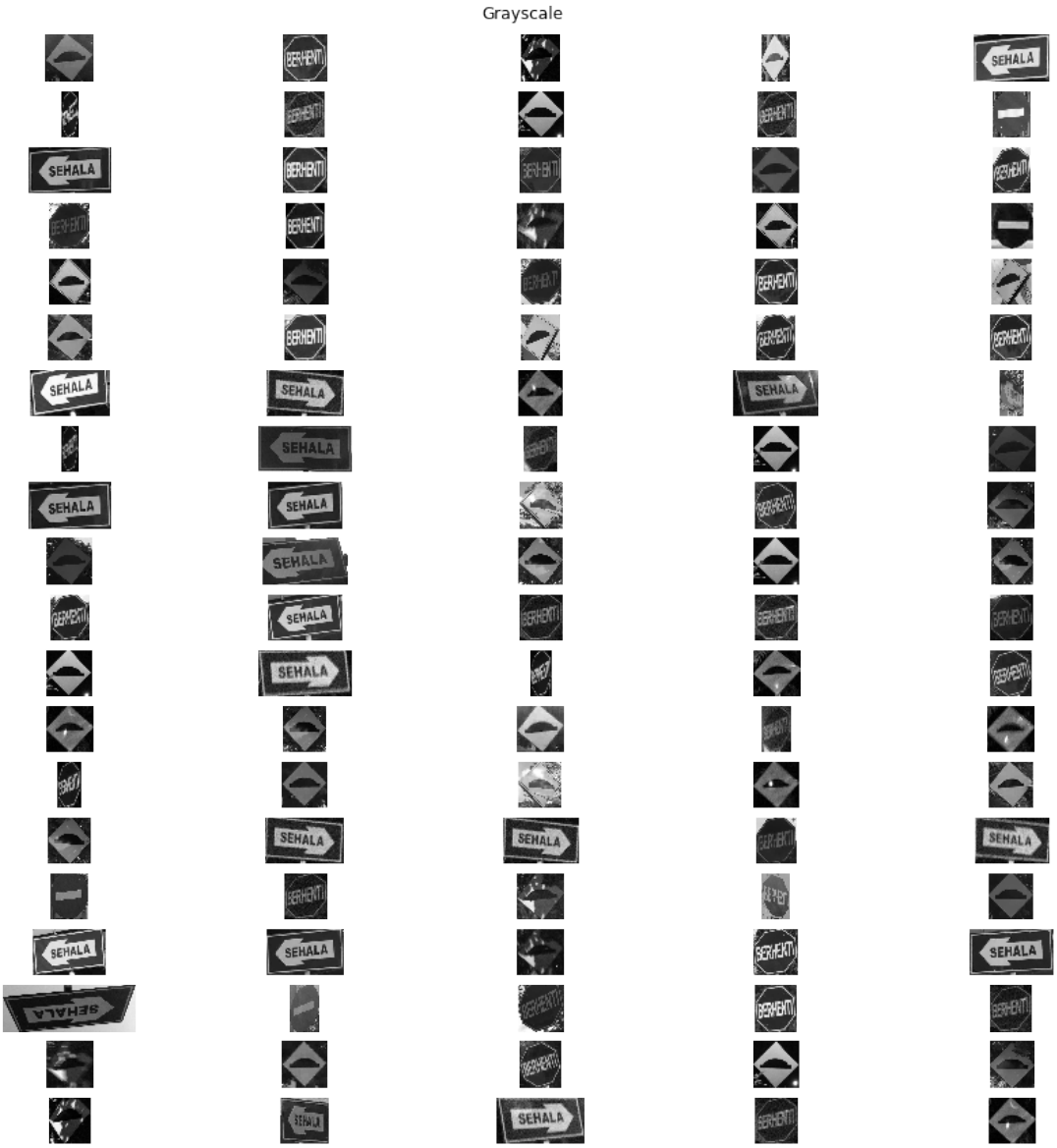
```
In [19]: # validation set
         validationSet = sample(dataset, 100)
         showDataset(validationSet, "Validation set", 5, y = 0.9)
```


Grayscale

```
In [20]: for data in validationSet:
             data.grayscale()

         showDataset(validationSet, "Grayscale", 5, y = 0.9)
```

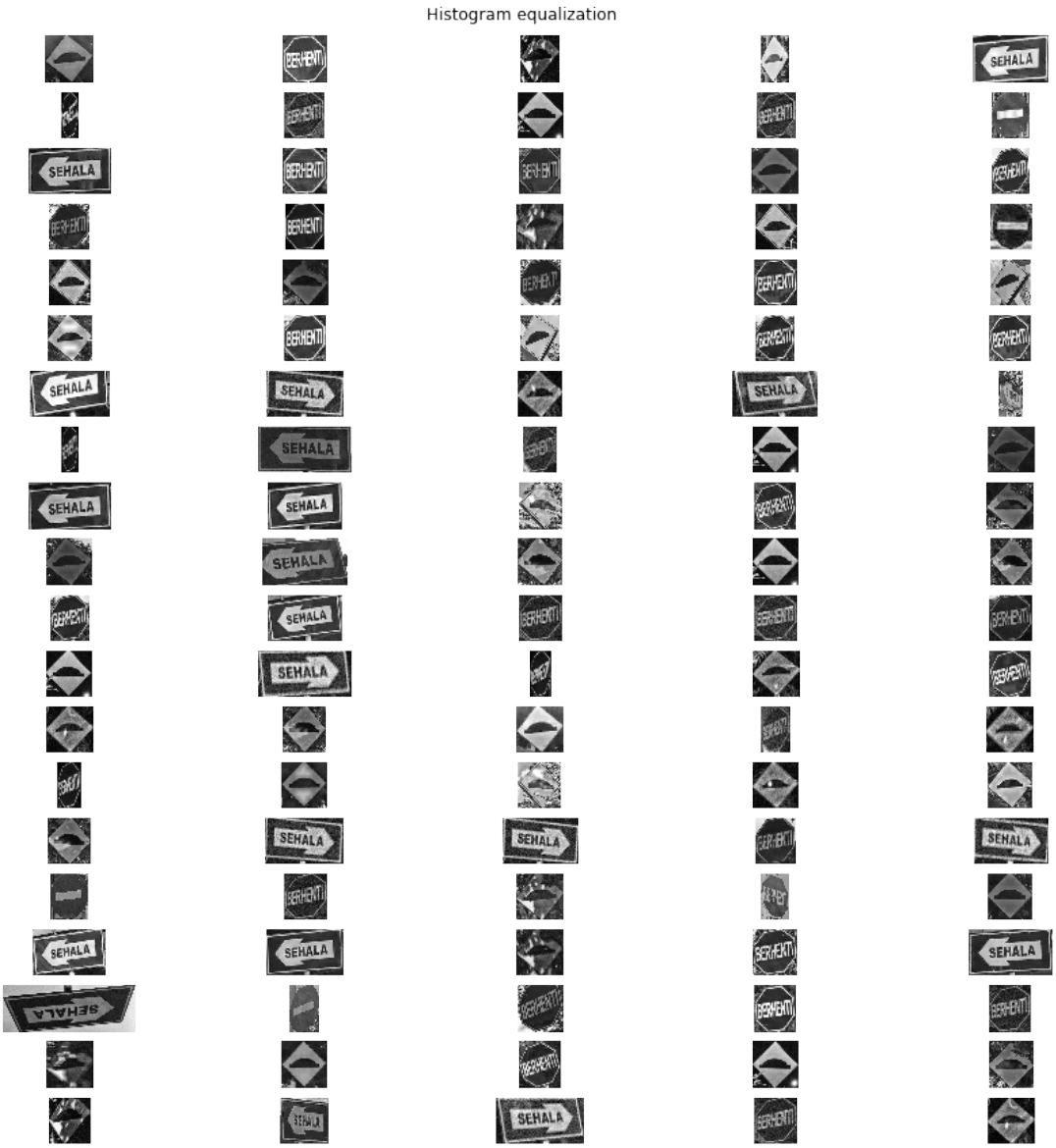
```
C:\Users\GaryNg\Anaconda3\lib\site-packages\skimage\util\dtype.py:141: UserWarning: Possible pr
```

```
.format(dtypeobj_in, dtypeobj_out))
```



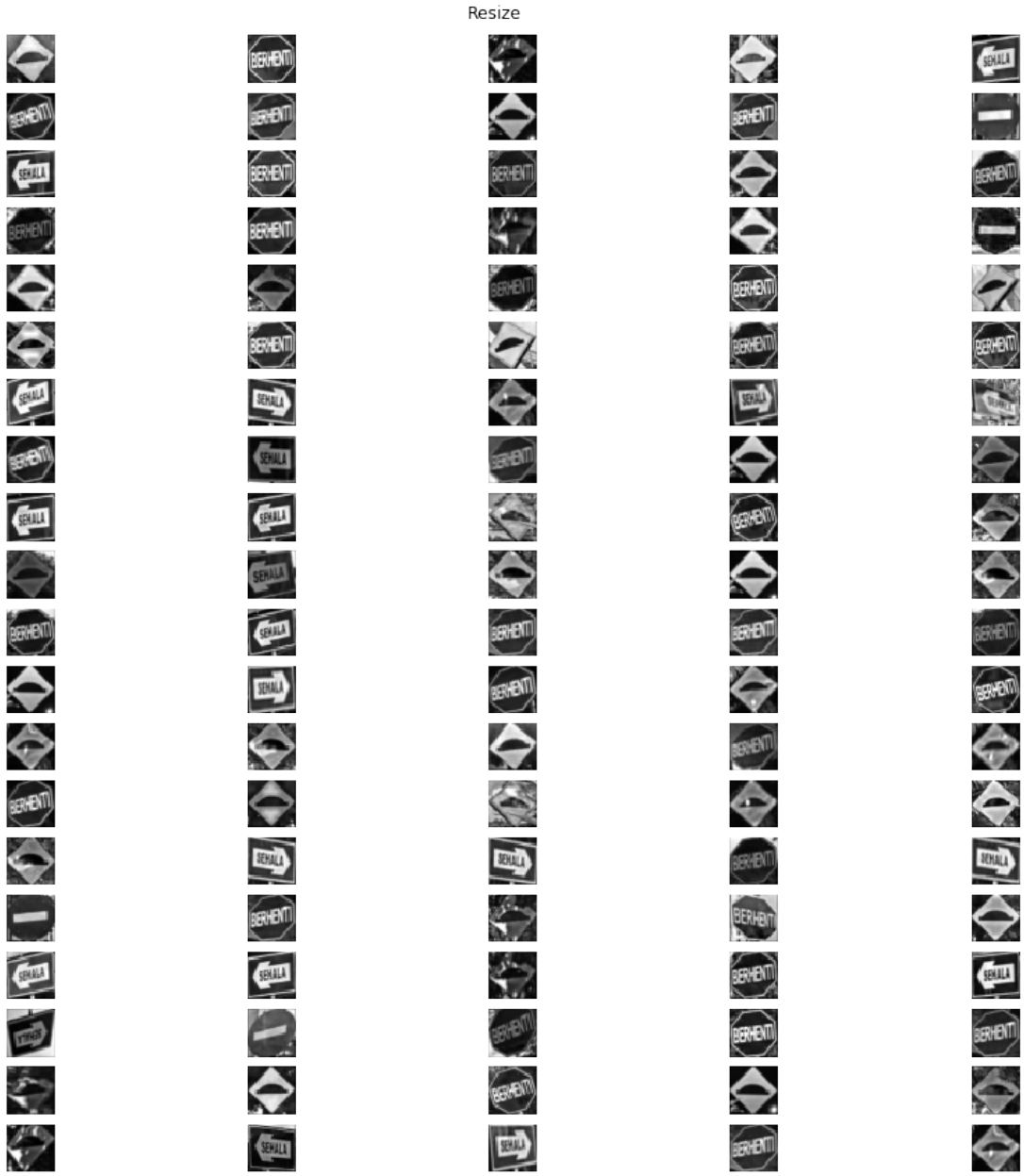

Histogram equalization

```
In [21]: for data in validationSet:
             data.equalize()

         showDataset(validationSet, "Histogram equalization", 5, y = 0.9)
```




Resize

```
In [22]: for data in validationSet:
             data.resize()

         showDataset(validationSet, "Resize", 5, y = 0.9)
```




Final width

```
In [23]: minMaxAverage("width", seq(validationSet).select(lambda data: data.image.shape[0]))

Minimum width: 32
Maximum width: 32
Average width: 32.0
```

Final height

```
In [24]: minMaxAverage("height", seq(validationSet).select(lambda data: data.image.shape[1]))
```

```
Minimum height: 32
Maximum height: 32
Average height: 32.0
```

Pickling preprocessed data

```
In [25]: import pickle

         def pickleData():
             dataset = loadDataset()
             for data in dataset:
                 data.preprocess()

             images, labels = getDataLabelsFromDataset(dataset)

             print(images.shape)
             print(labels.shape)

             pickle.dump(images, open('images.pkl', 'wb'))
             pickle.dump(labels, open('labels.pkl', 'wb'))
```

```
In [26]: def unpickleData():
             images = pickle.load(open("./pkl/images.pkl", "rb"))
             labels = pickle.load(open("./pkl/labels.pkl", "rb"))
             return (images, labels)
```

```
In [27]: def getData():
             """Return unpicked data seperated into training and testing sets"""
             return train_test_split(*unpickleData(), random_state=123)
```

```
In [28]: def scatter3d(x, y, z, labels):
             ax = plt.axes(projection='3d')
             ax.scatter(x, y, z, c = labels)
             ax.set_xticklabels([])
             ax.set_yticklabels([])
             ax.set_zticklabels([])
             plt.show()
```

### 0.1 Model 1: k-NN

```
In [29]: def crossValidateKnn(data, labels):
             scores = []
             neighbours = seq(range(1, 50)).filter(lambda i: i % 2 == 1).to_list()
             for k in neighbours:
                 knn = KNeighborsClassifier(n_neighbors=k)
                 # 10-fold
```

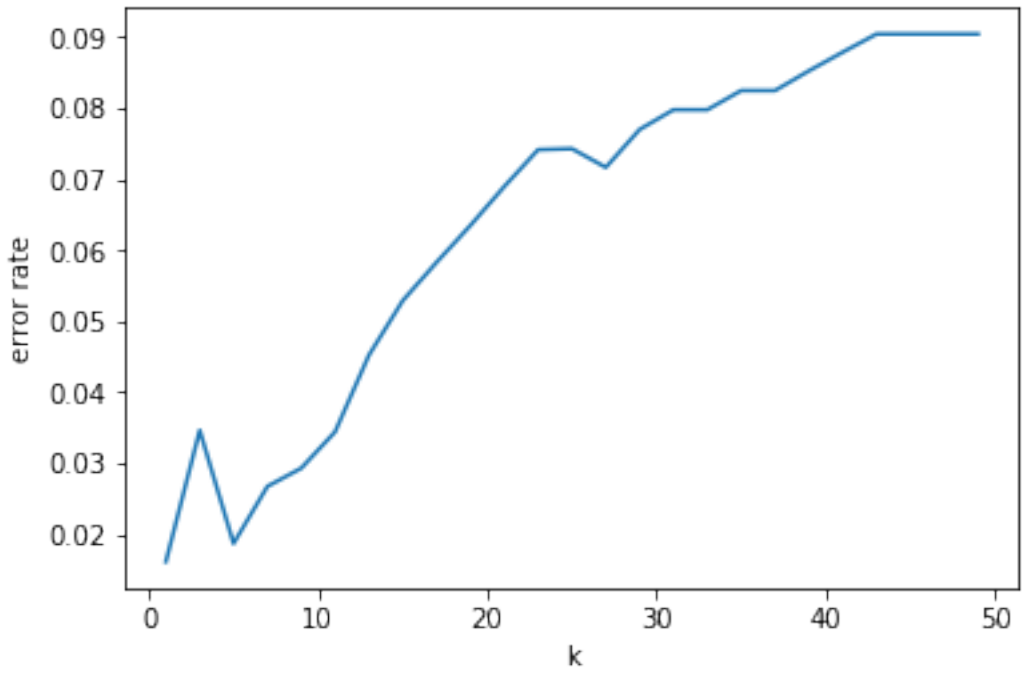
```
            scores.append(cross_val_score(knn, data, labels, cv=10, scoring='accuracy').me

        mse = [1 - x for x in scores] # miss-classification error
        plt.xlabel('k')
        plt.ylabel('error rate')
        plt.plot(neighbours, mse)

        optimalK = neighbours[mse.index(min(mse))]
        print(f'optimal k: {optimalK}')

        return optimalK
```

In [30]:
```
def trainKnn(trainingData, testingData, trainingLabels, testingLabels):

    print(f'trainingData: {trainingData.shape}, trainingLabels: {trainingLabels.shape}
    print(f'testingData: {testingData.shape}, testingLabels: {testingLabels.shape}')

    k = crossValidateKnn(trainingData, trainingLabels)

    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(trainingData, trainingLabels)
    print(f'training accuracy: {knn.score(trainingData, trainingLabels)}')
    print(f'testing accuracy: {knn.score(testingData, testingLabels)}')

    predictedLabels = knn.predict(testingData)
    print(f'confusion matrix: \n {confusion_matrix(testingLabels, predictedLabels)}')
    print(classification_report(testingLabels, predictedLabels))
```

In [31]:
```
trainKnn(*getData())
```

```
trainingData: (375, 1024), trainingLabels: (375,)
testingData: (125, 1024), testingLabels: (125,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.984
confusion matrix:
 [[ 7  0  0  1]
 [ 1 22  0  0]
 [ 0  0 67  0]
 [ 0  0  0 27]]
              precision    recall  f1-score   support

           0       0.88      0.88      0.88         8
           1       1.00      0.96      0.98        23
           2       1.00      1.00      1.00        67
           3       0.96      1.00      0.98        27

   micro avg       0.98      0.98      0.98       125
```

```
   macro avg       0.96      0.96      0.96       125
weighted avg       0.98      0.98      0.98       125
```



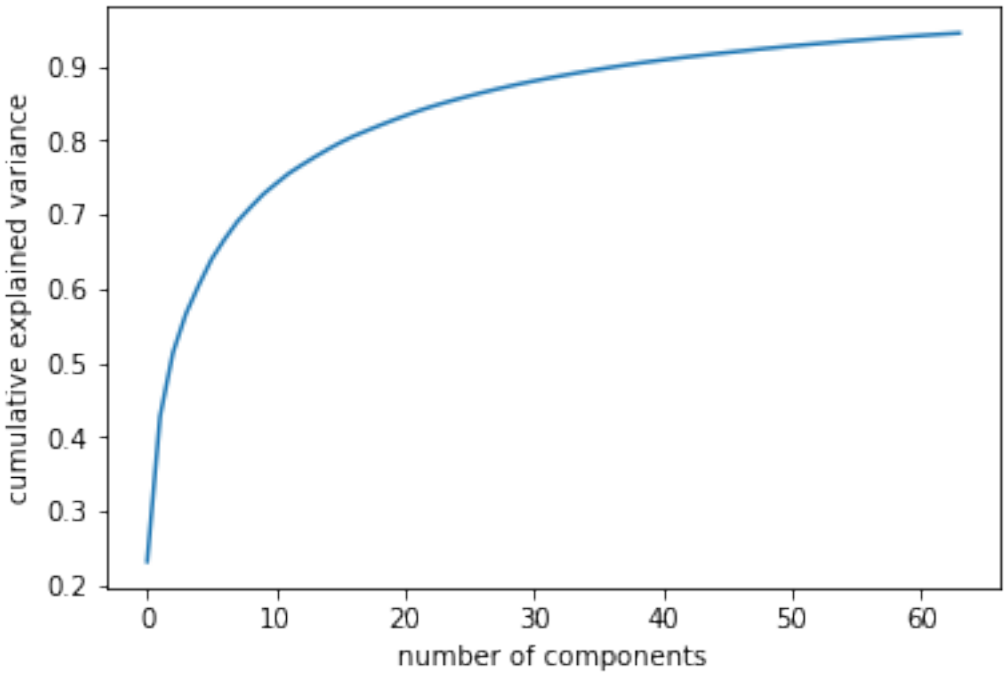

# 1 Model 2: PCA-kNN

```
In [32]: def fitPca(trainingData, testingData, trainingLabels, testingLabels, components=64, s
             print(f'trainingData: {trainingData.shape}, trainingLabels: {trainingLabels.shape
             print(f'testingData: {testingData.shape}, testingLabels: {testingLabels.shape}')

             scaler = StandardScaler()
             trainingData = scaler.fit_transform(trainingData)
             testingData = scaler.fit_transform(testingData)

             pca = PCA(n_components=components, svd_solver='randomized', whiten=True)
             pca.fit(trainingData)

             print(f'total explained variance: {pca.explained_variance_ratio_[:components].sum
             plt.plot(np.cumsum(pca.explained_variance_ratio_))
             plt.xlabel('number of components')
             plt.ylabel('cumulative explained variance')
             plt.show()
```

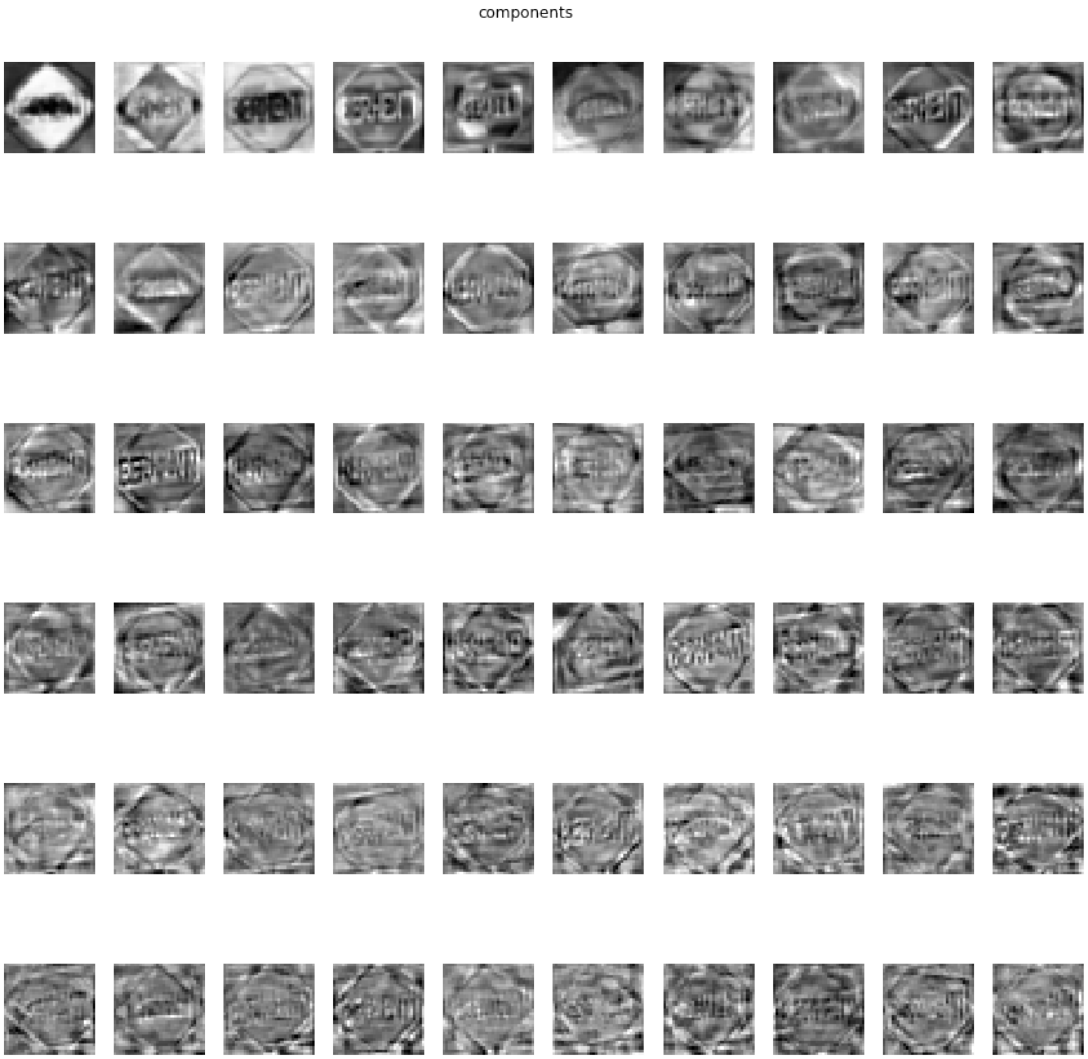
```
    print(f'pca components: {pca.components_.shape}')
    showImages(pca.components_.reshape(-1, size[0], size[1]), "components", y=0.9)

    trainingData = pca.transform(trainingData)
    testingData = pca.transform(testingData)

    return (trainingData, testingData, trainingLabels, testingLabels)
```

In [33]:
```
trainKnn(*fitPca(*getData()))
```

```
trainingData: (375, 1024), trainingLabels: (375,)
testingData: (125, 1024), testingLabels: (125,)
total explained variance: 0.9442995795943376
```

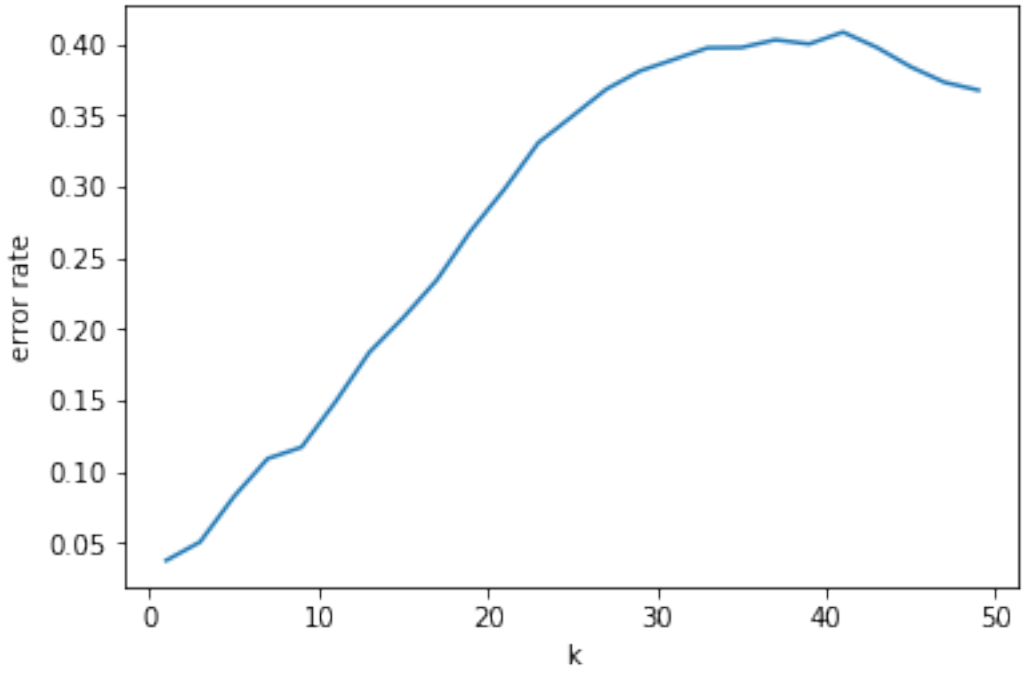



```
pca components: (64, 1024)
```




```
trainingData: (375, 64), trainingLabels: (375,)
testingData: (125, 64), testingLabels: (125,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.984
confusion matrix:
 [[ 7  0  0  1]
 [ 0 22  0  1]
 [ 0  0 67  0]
 [ 0  0  0 27]]
              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       1.00      0.96      0.98        23
```

```
           2       1.00      1.00      1.00        67
           3       0.93      1.00      0.96        27

   micro avg       0.98      0.98      0.98       125
   macro avg       0.98      0.96      0.97       125
weighted avg       0.99      0.98      0.98       125
```



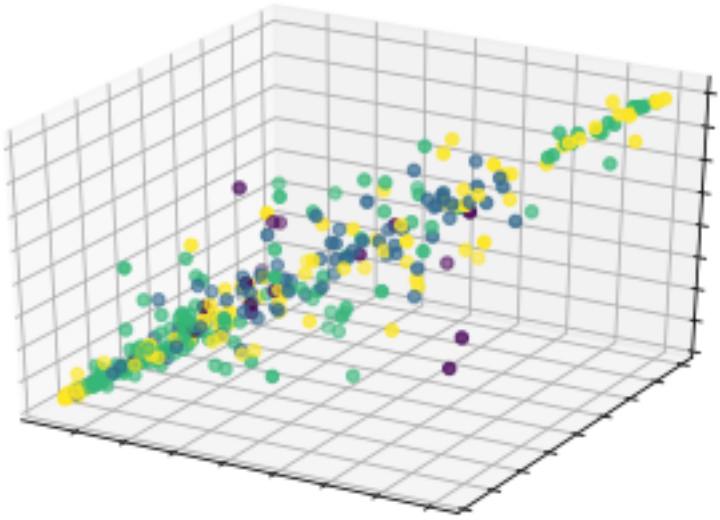

## 2 Model 3: LDA-kNN

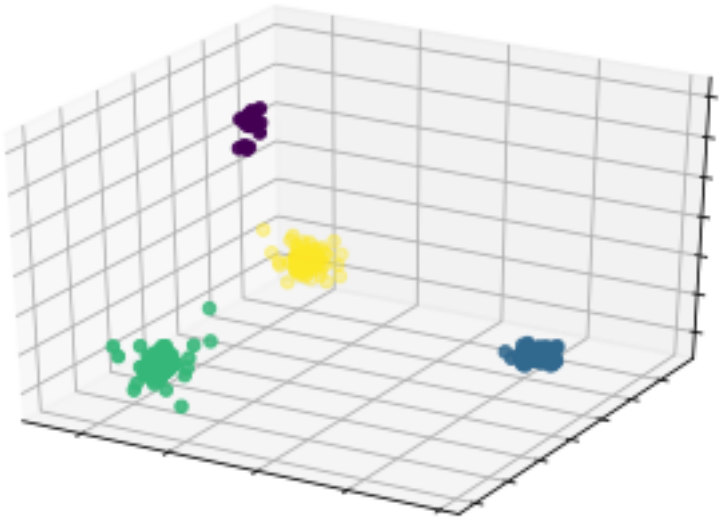
```
In [34]: def fitLda(trainingData, testingData, trainingLabels, testingLabels):
             print(f'trainingData: {trainingData.shape}, trainingLabels: {trainingLabels.shape}
             print(f'testingData: {testingData.shape}, testingLabels: {testingLabels.shape}')

             scaler = StandardScaler()
             trainingData = scaler.fit_transform(trainingData)
             testingData = scaler.fit_transform(testingData)

             lda = LDA(n_components=3)
             lda.fit(trainingData, trainingLabels)

             scatter3d(trainingData[:, 0], trainingData[:, 1],trainingData[:, 2], trainingLabel
```

```
    trainingData = lda.transform(trainingData)
    testingData = lda.transform(testingData)

    scatter3d(trainingData[:, 0], trainingData[:, 1],trainingData[:, 2], trainingLabel

    return (trainingData, testingData, trainingLabels, testingLabels)
```

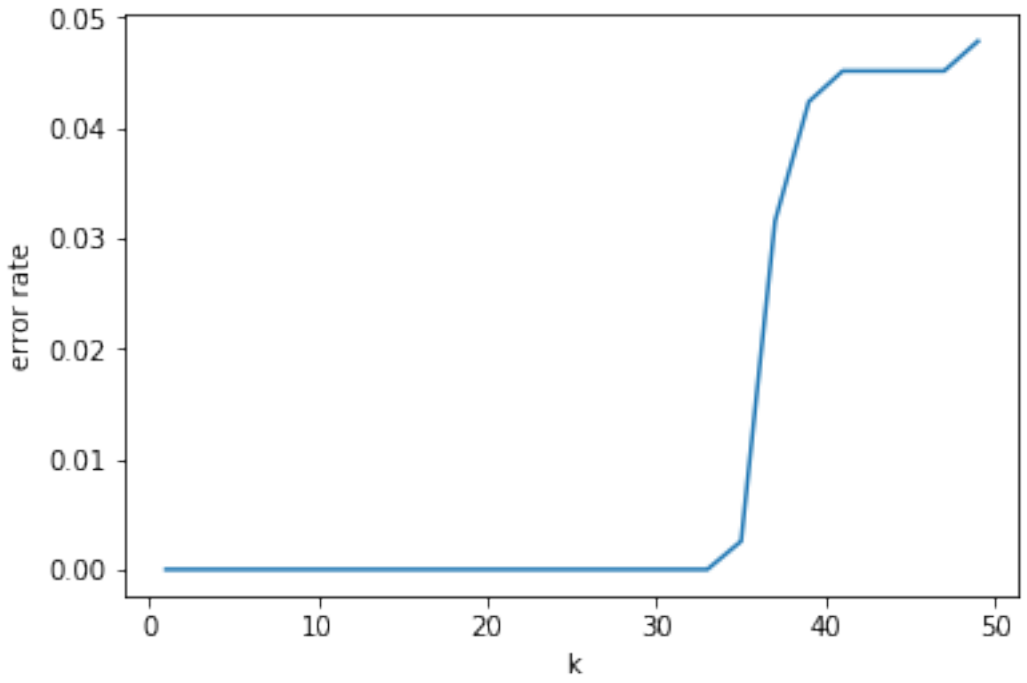
```
In [35]: trainKnn(*fitLda(*getData()))
```

```
trainingData: (375, 1024), trainingLabels: (375,)
testingData: (125, 1024), testingLabels: (125,)


C:\Users\GaryNg\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning:
  warnings.warn("Variables are collinear.")
```



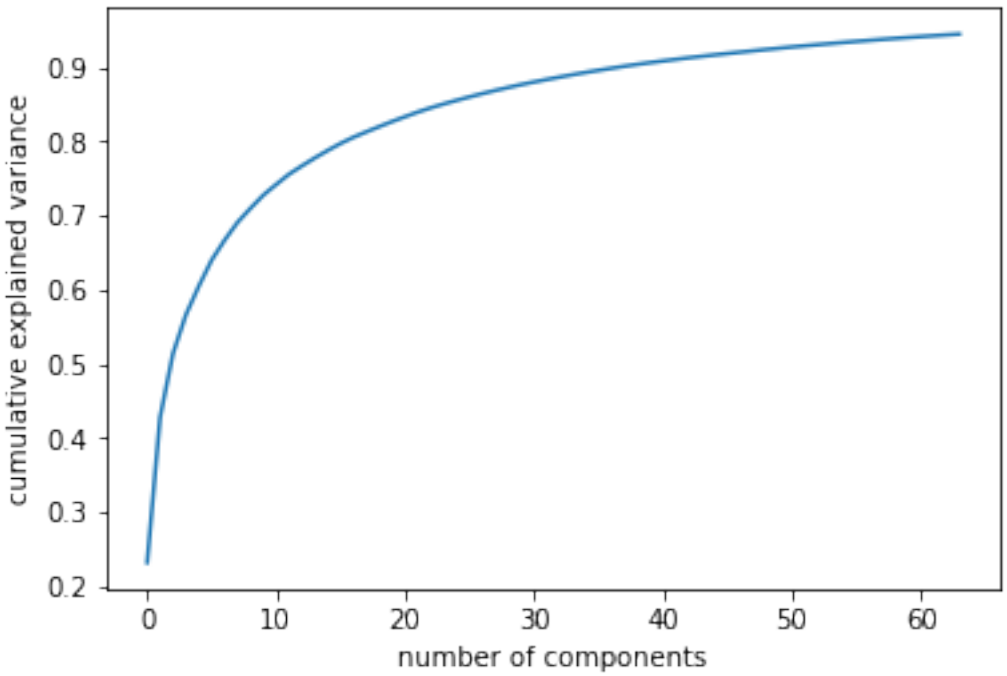


```
trainingData: (375, 3), trainingLabels: (375,)
testingData: (125, 3), testingLabels: (125,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.96
confusion matrix:
 [[ 7  0  0  1]
 [ 1 21  1  0]
 [ 1  1 65  0]
 [ 0  0  0 27]]
              precision    recall  f1-score   support

           0       0.78      0.88      0.82         8
           1       0.95      0.91      0.93        23
           2       0.98      0.97      0.98        67
           3       0.96      1.00      0.98        27

   micro avg       0.96      0.96      0.96       125
   macro avg       0.92      0.94      0.93       125
weighted avg       0.96      0.96      0.96       125
```

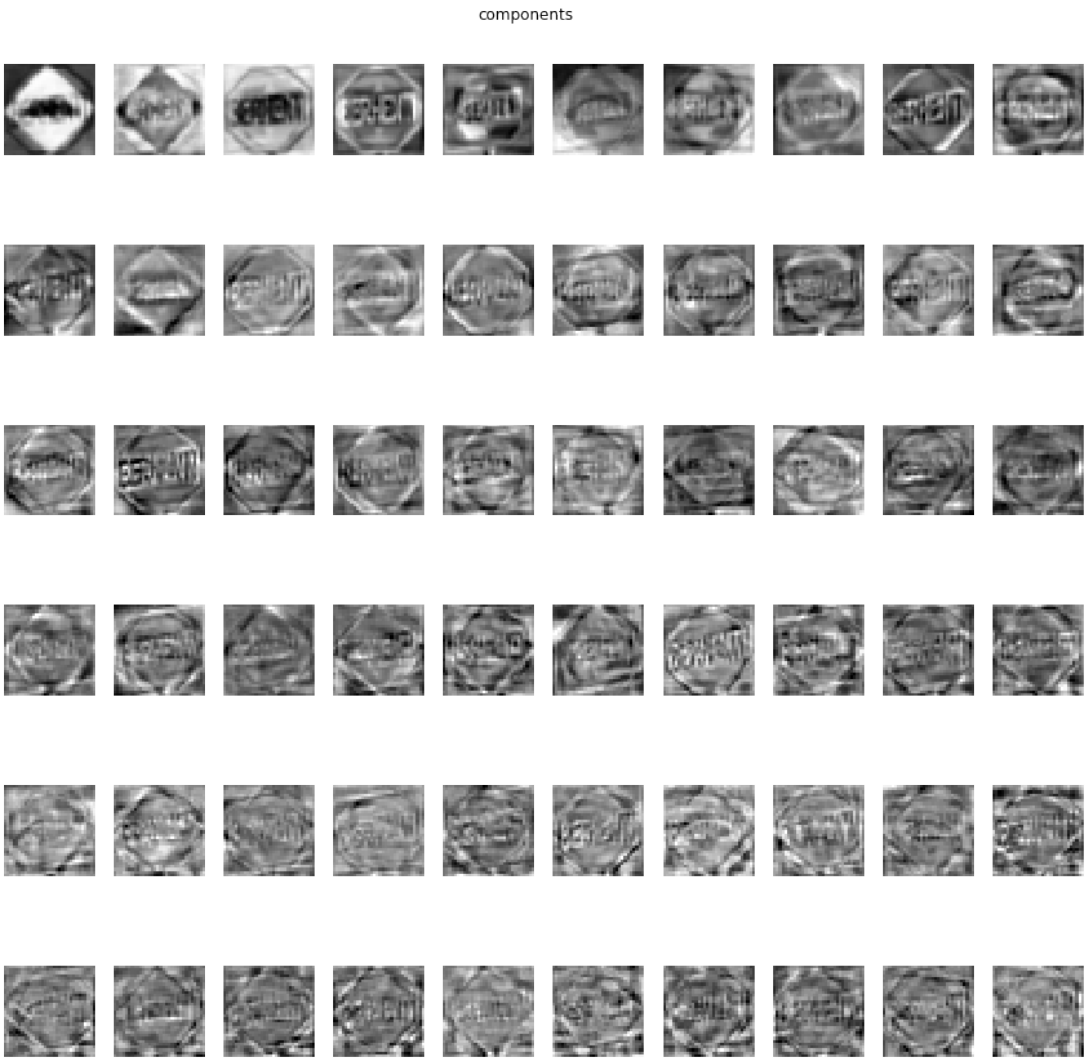



## 3 Model 3: PCA-LDA-kNN

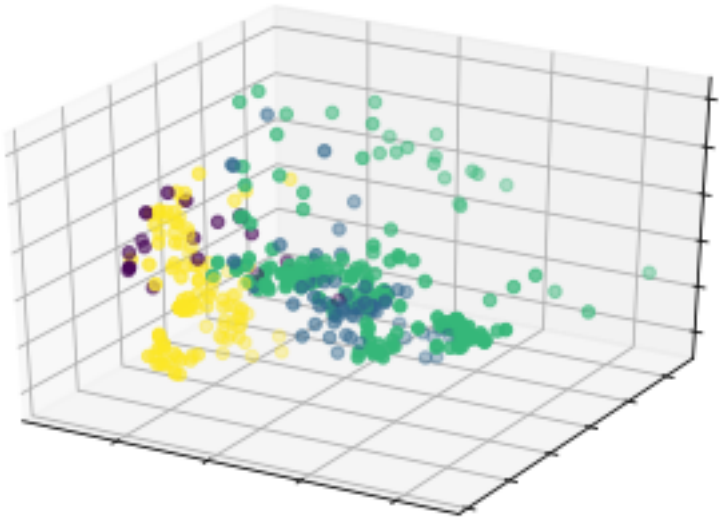
```
In [36]: trainKnn(*fitLda(*fitPca(*getData())))

trainingData: (375, 1024), trainingLabels: (375,)
testingData: (125, 1024), testingLabels: (125,)
total explained variance: 0.9443089031170777
```

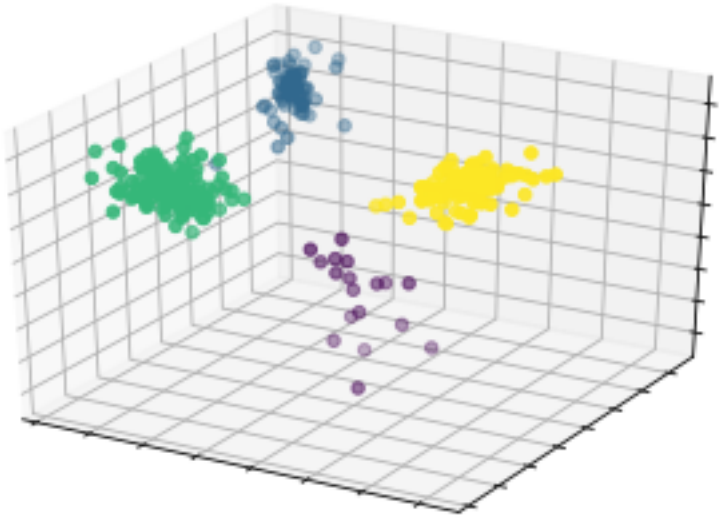



```
pca components: (64, 1024)
```



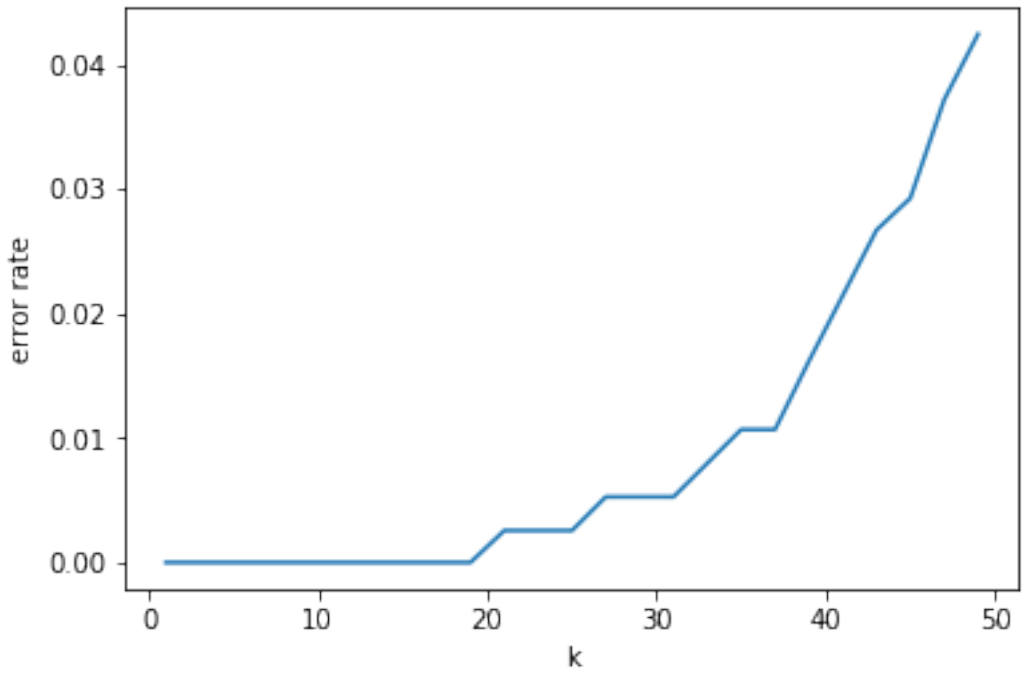

```
trainingData: (375, 64), trainingLabels: (375,)
testingData: (125, 64), testingLabels: (125,)
```





```
trainingData: (375, 3), trainingLabels: (375,)
testingData: (125, 3), testingLabels: (125,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.992
confusion matrix:
```

```
[[ 7  0  0  1]
 [ 0 23  0  0]
 [ 0  0 67  0]
 [ 0  0  0 27]]
              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       1.00      1.00      1.00        23
           2       1.00      1.00      1.00        67
           3       0.96      1.00      0.98        27

   micro avg       0.99      0.99      0.99       125
   macro avg       0.99      0.97      0.98       125
weighted avg       0.99      0.99      0.99       125
```



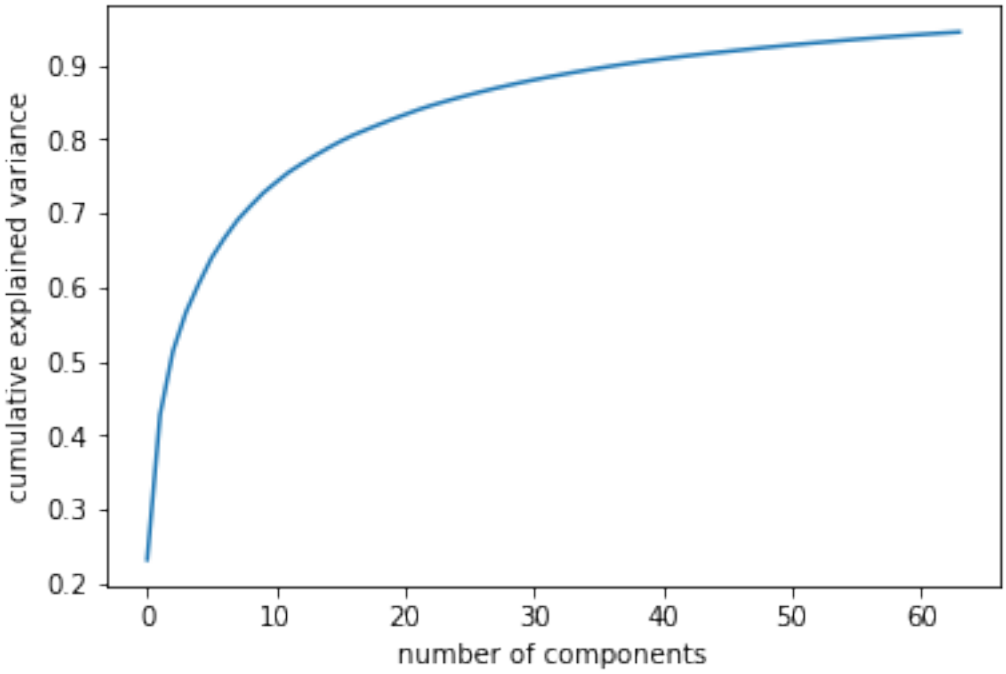

## 4 Model 4: PCA-SVM

```
In [37]: def trainSvm(trainingData, testingData, trainingLabels, testingLabels):
             print(f'trainingData: {trainingData.shape}, trainingLabels: {trainingLabels.shape
             print(f'testingData: {testingData.shape}, testingLabels: {testingLabels.shape}')

             svm = SVM.SVC(gamma='auto', kernel='linear', random_state=0)
```

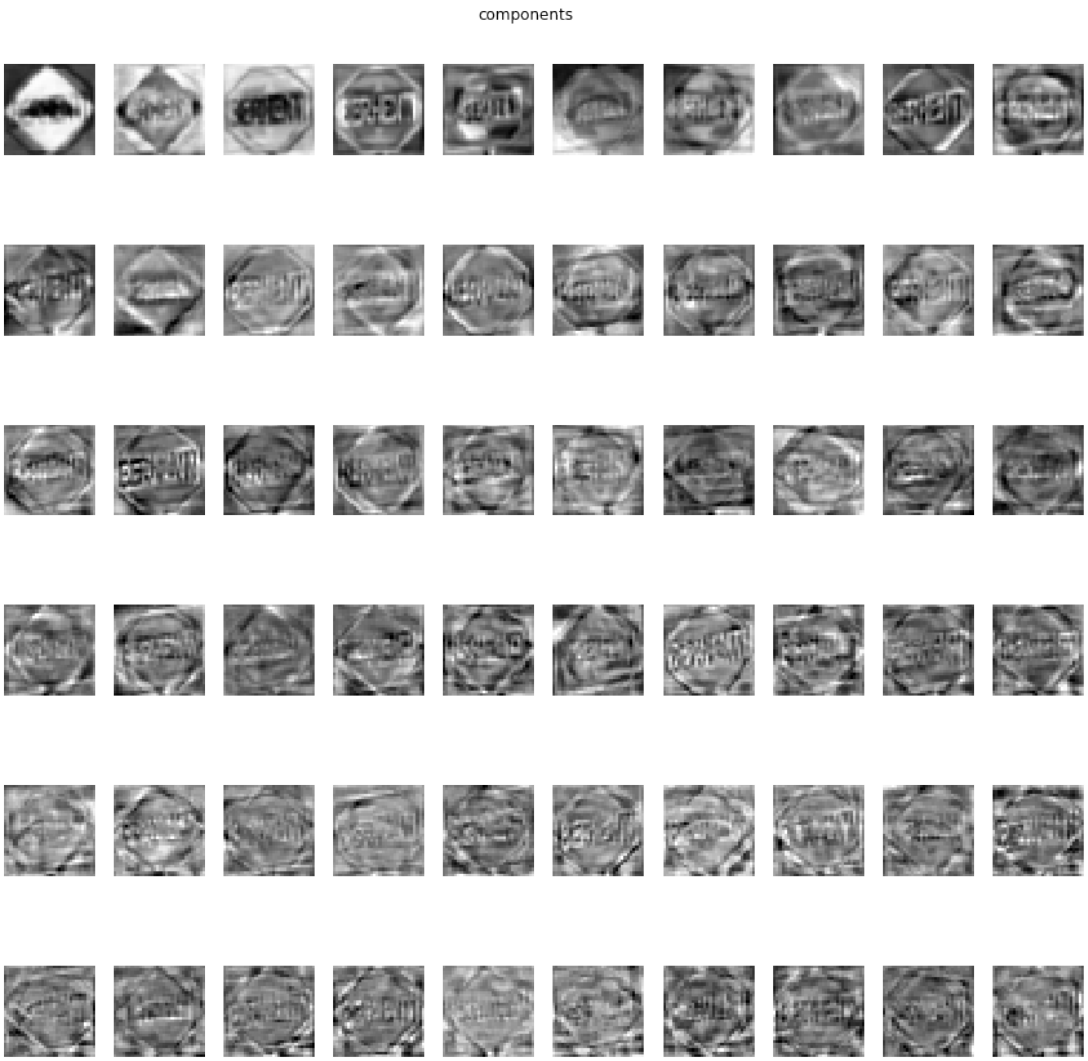
```
    svm.fit(trainingData, trainingLabels)

    print(f'training accuracy: {svm.score(trainingData, trainingLabels)}')
    print(f'testing accuracy: {svm.score(testingData, testingLabels)}')

    predictedLabels = svm.predict(testingData)
    print(f'confusion matrix: \n {confusion_matrix(testingLabels, predictedLabels)}')
    print(classification_report(testingLabels, predictedLabels))
```

```
In [38]: trainSvm(*fitPca(*getData()))
```

```
trainingData: (375, 1024), trainingLabels: (375,)
testingData: (125, 1024), testingLabels: (125,)
total explained variance: 0.9443228014007938
```




```
pca components: (64, 1024)
```



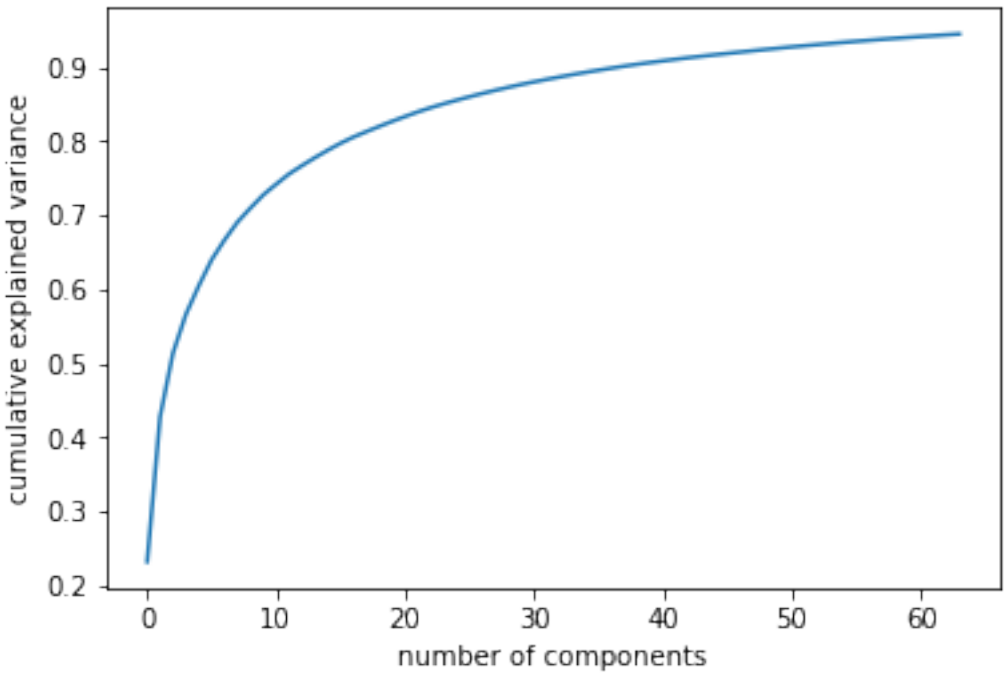

```
trainingData: (375, 64), trainingLabels: (375,)
testingData: (125, 64), testingLabels: (125,)
training accuracy: 1.0
testing accuracy: 0.992
confusion matrix:
 [[ 7  0  0  1]
 [ 0 23  0  0]
 [ 0  0 67  0]
 [ 0  0  0 27]]
              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       1.00      1.00      1.00        23
           2       1.00      1.00      1.00        67
```

```
           3       0.96      1.00      0.98        27

   micro avg       0.99      0.99      0.99       125
   macro avg       0.99      0.97      0.98       125
weighted avg       0.99      0.99      0.99       125
```

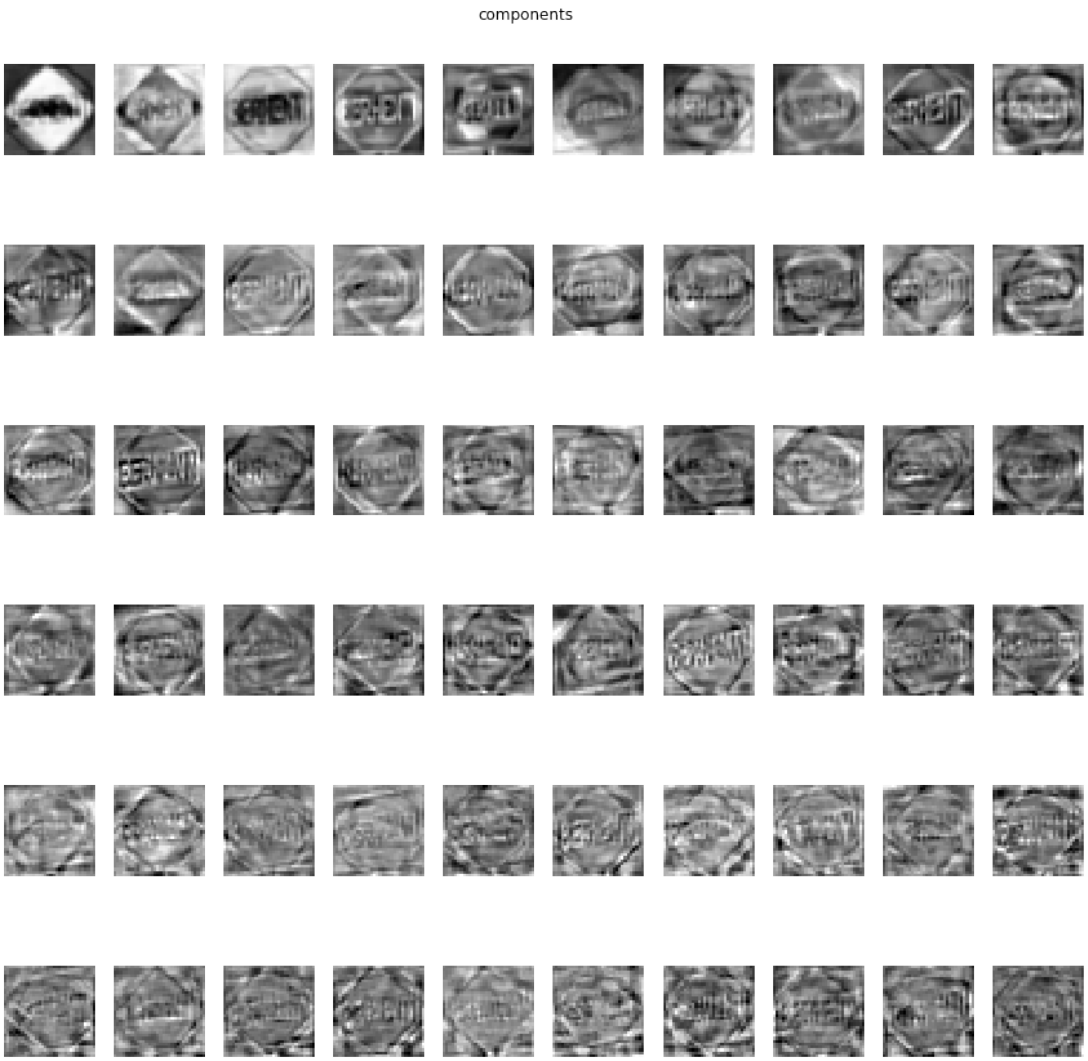
# 5 Model 6: PCA-LDA-SVM

```
In [39]: trainSvm(*fitLda(*fitPca(*getData())))
```

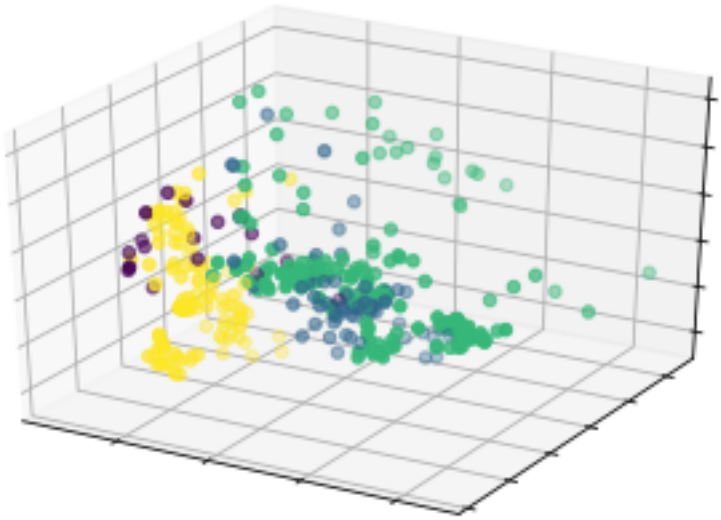
```
trainingData: (375, 1024), trainingLabels: (375,)
testingData: (125, 1024), testingLabels: (125,)
total explained variance: 0.9443203098661364
```

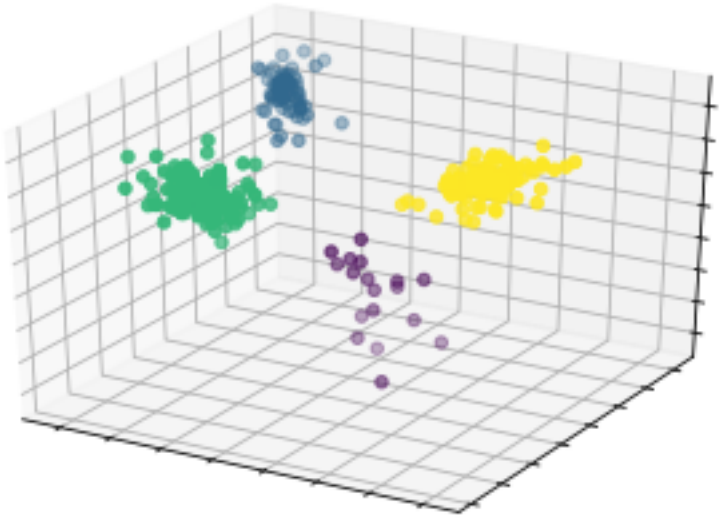



```
pca components: (64, 1024)
```




```
trainingData: (375, 64), trainingLabels: (375,)
testingData: (125, 64), testingLabels: (125,)
```





```
trainingData: (375, 3), trainingLabels: (375,)
testingData: (125, 3), testingLabels: (125,)
training accuracy: 1.0
testing accuracy: 0.992
confusion matrix:
 [[ 7  0  0  1]
```

```
 [ 0 23  0  0]
 [ 0  0 67  0]
 [ 0  0  0 27]]
              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       1.00      1.00      1.00        23
           2       1.00      1.00      1.00        67
           3       0.96      1.00      0.98        27

   micro avg       0.99      0.99      0.99       125
   macro avg       0.99      0.97      0.98       125
weighted avg       0.99      0.99      0.99       125
```

# 6 Model 7: MLP

```
In [40]: def trainMlp(trainingData, testingData, trainingLabels, testingLabels, layers=(1024,

             print(f'trainingData: {trainingData.shape}, trainingLabels: {trainingLabels.shape
             print(f'testingData: {testingData.shape}, testingLabels: {testingLabels.shape}')

             scaler = StandardScaler()
             trainingData = scaler.fit_transform(trainingData)
             testingData = scaler.fit_transform(testingData)

             mlp = MLPClassifier(random_state=0, verbose=True, hidden_layer_sizes=layers)
             mlp.fit(trainingData, trainingLabels)

             print(mlp)
             print(f'training accuracy: {mlp.score(trainingData, trainingLabels)}')
             print(f'testing accuracy: {mlp.score(testingData, testingLabels)}')

             predictedLabels = mlp.predict(testingData)
             print(f'confusion matrix: \n {confusion_matrix(testingLabels, predictedLabels)}')
             print(classification_report(testingLabels, predictedLabels))

In [41]: trainMlp(*getData())

trainingData: (375, 1024), trainingLabels: (375,)
testingData: (125, 1024), testingLabels: (125,)
Iteration 1, loss = 0.98445011
Iteration 2, loss = 0.08688085
Iteration 3, loss = 0.06993920
Iteration 4, loss = 0.00259222
Iteration 5, loss = 0.00732444
Iteration 6, loss = 0.00837116
```

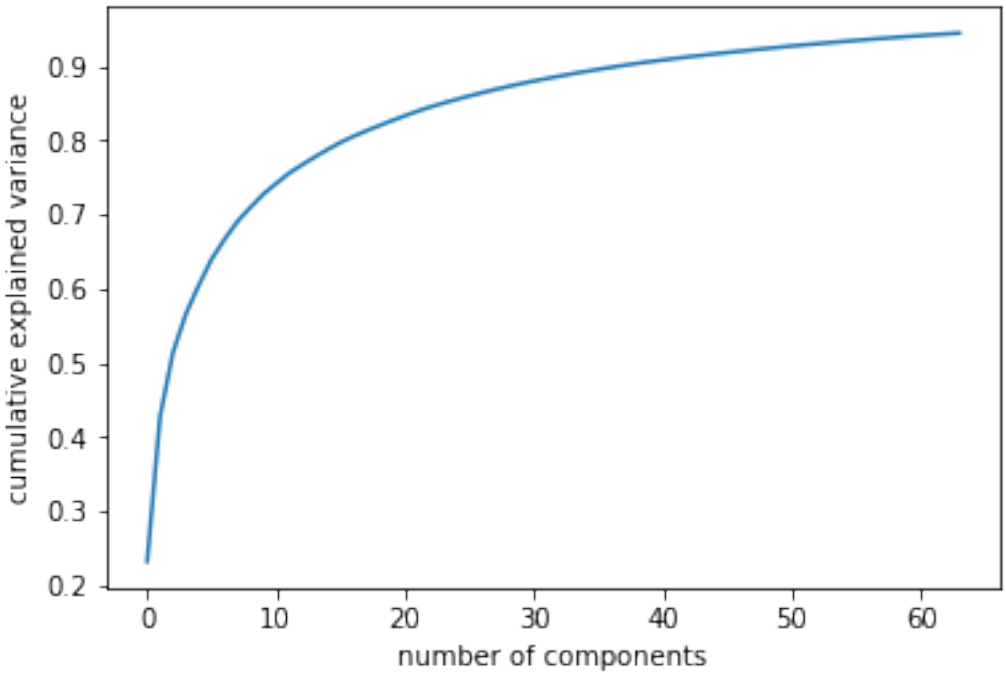
```
Iteration 7, loss = 0.00230393
Iteration 8, loss = 0.00108071
Iteration 9, loss = 0.00069248
Iteration 10, loss = 0.00068976
Iteration 11, loss = 0.00069921
Iteration 12, loss = 0.00068605
Iteration 13, loss = 0.00070754
Iteration 14, loss = 0.00071357
Iteration 15, loss = 0.00070887
Iteration 16, loss = 0.00067750
Iteration 17, loss = 0.00064647
Iteration 18, loss = 0.00062245
Iteration 19, loss = 0.00060275
Iteration 20, loss = 0.00059064
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.992
confusion matrix:
 [[ 7  0  0  1]
 [ 0 23  0  0]
 [ 0  0 67  0]
 [ 0  0  0 27]]
              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       1.00      1.00      1.00        23
           2       1.00      1.00      1.00        67
           3       0.96      1.00      0.98        27

   micro avg       0.99      0.99      0.99       125
   macro avg       0.99      0.97      0.98       125
weighted avg       0.99      0.99      0.99       125
```

# 7 Model 8: PCA-MLP

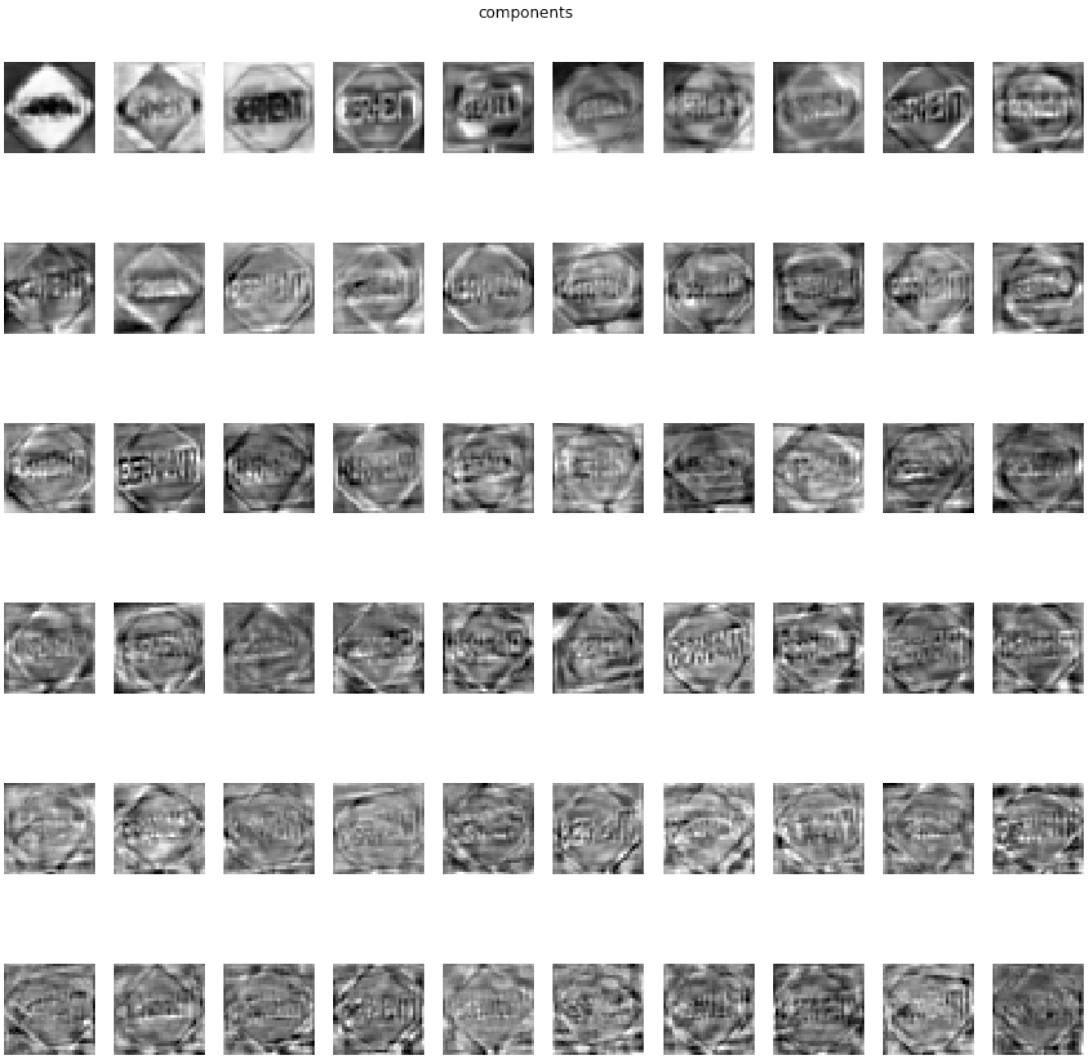
```
In [42]: trainMlp(*fitPca(*getData()), layers=(64, 64))
```

```
trainingData: (375, 1024), trainingLabels: (375,)
testingData: (125, 1024), testingLabels: (125,)
```

```
total explained variance: 0.9443533218455223
```




```
pca components: (64, 1024)
```




```
trainingData: (375, 64), trainingLabels: (375,)
testingData: (125, 64), testingLabels: (125,)
Iteration 1, loss = 1.61309297
Iteration 2, loss = 1.49522447
Iteration 3, loss = 1.39456534
Iteration 4, loss = 1.30605626
Iteration 5, loss = 1.23478735
Iteration 6, loss = 1.16834564
Iteration 7, loss = 1.11125695
Iteration 8, loss = 1.06012242
Iteration 9, loss = 1.01319540
Iteration 10, loss = 0.96900562
Iteration 11, loss = 0.92744500
Iteration 12, loss = 0.88579616
```

```
Iteration 13, loss = 0.84561256
Iteration 14, loss = 0.80534216
Iteration 15, loss = 0.76612544
Iteration 16, loss = 0.72674336
Iteration 17, loss = 0.68932938
Iteration 18, loss = 0.64962758
Iteration 19, loss = 0.61179703
Iteration 20, loss = 0.57423252
Iteration 21, loss = 0.53786412
Iteration 22, loss = 0.50174423
Iteration 23, loss = 0.46710909
Iteration 24, loss = 0.43282183
Iteration 25, loss = 0.40073867
Iteration 26, loss = 0.36958091
Iteration 27, loss = 0.33980666
Iteration 28, loss = 0.31196384
Iteration 29, loss = 0.28501822
Iteration 30, loss = 0.26042973
Iteration 31, loss = 0.23734103
Iteration 32, loss = 0.21544692
Iteration 33, loss = 0.19588478
Iteration 34, loss = 0.17806760
Iteration 35, loss = 0.16172652
Iteration 36, loss = 0.14657222
Iteration 37, loss = 0.13300500
Iteration 38, loss = 0.12058662
Iteration 39, loss = 0.10922476
Iteration 40, loss = 0.09920773
Iteration 41, loss = 0.09016440
Iteration 42, loss = 0.08213478
Iteration 43, loss = 0.07470895
Iteration 44, loss = 0.06812687
Iteration 45, loss = 0.06205975
Iteration 46, loss = 0.05670362
Iteration 47, loss = 0.05204235
Iteration 48, loss = 0.04762277
Iteration 49, loss = 0.04397917
Iteration 50, loss = 0.04055181
Iteration 51, loss = 0.03751541
Iteration 52, loss = 0.03476756
Iteration 53, loss = 0.03227943
Iteration 54, loss = 0.03008537
Iteration 55, loss = 0.02814247
Iteration 56, loss = 0.02631354
Iteration 57, loss = 0.02467790
Iteration 58, loss = 0.02324352
Iteration 59, loss = 0.02187501
Iteration 60, loss = 0.02068822
```

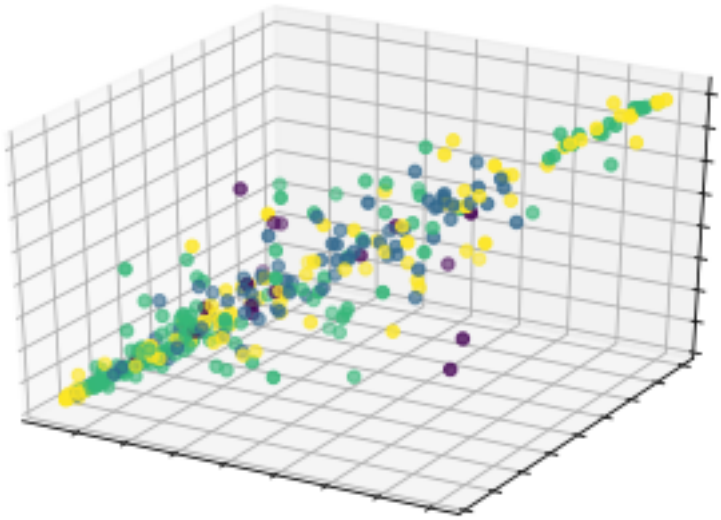
```
Iteration 61, loss = 0.01960116
Iteration 62, loss = 0.01854961
Iteration 63, loss = 0.01760003
Iteration 64, loss = 0.01674325
Iteration 65, loss = 0.01596269
Iteration 66, loss = 0.01524824
Iteration 67, loss = 0.01457148
Iteration 68, loss = 0.01394195
Iteration 69, loss = 0.01337174
Iteration 70, loss = 0.01281246
Iteration 71, loss = 0.01228604
Iteration 72, loss = 0.01182083
Iteration 73, loss = 0.01138216
Iteration 74, loss = 0.01095674
Iteration 75, loss = 0.01055550
Iteration 76, loss = 0.01018033
Iteration 77, loss = 0.00984592
Iteration 78, loss = 0.00950551
Iteration 79, loss = 0.00919773
Iteration 80, loss = 0.00889841
Iteration 81, loss = 0.00861309
Iteration 82, loss = 0.00834611
Iteration 83, loss = 0.00808245
Iteration 84, loss = 0.00784832
Iteration 85, loss = 0.00761591
Iteration 86, loss = 0.00739551
Iteration 87, loss = 0.00718939
Iteration 88, loss = 0.00698775
Iteration 89, loss = 0.00679726
Iteration 90, loss = 0.00661457
Iteration 91, loss = 0.00643816
Iteration 92, loss = 0.00626981
Iteration 93, loss = 0.00610679
Iteration 94, loss = 0.00595372
Iteration 95, loss = 0.00580335
Iteration 96, loss = 0.00566139
Iteration 97, loss = 0.00552619
Iteration 98, loss = 0.00539490
Iteration 99, loss = 0.00526264
Iteration 100, loss = 0.00514543
Iteration 101, loss = 0.00502622
Iteration 102, loss = 0.00491154
Iteration 103, loss = 0.00480080
Iteration 104, loss = 0.00469765
Iteration 105, loss = 0.00459589
Iteration 106, loss = 0.00449375
Iteration 107, loss = 0.00440062
Iteration 108, loss = 0.00430759
```

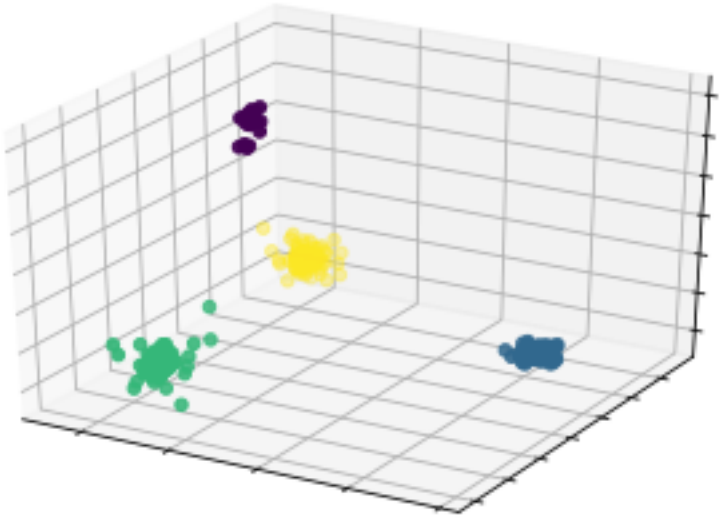
```
Iteration 109, loss = 0.00421617
Iteration 110, loss = 0.00413471
Iteration 111, loss = 0.00405013
Iteration 112, loss = 0.00396748
Iteration 113, loss = 0.00388823
Iteration 114, loss = 0.00381485
Iteration 115, loss = 0.00373782
Iteration 116, loss = 0.00366949
Iteration 117, loss = 0.00359932
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(64, 64), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.992
confusion matrix:
 [[ 7  0  0  1]
 [ 0 23  0  0]
 [ 0  0 67  0]
 [ 0  0  0 27]]
              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       1.00      1.00      1.00        23
           2       1.00      1.00      1.00        67
           3       0.96      1.00      0.98        27

   micro avg       0.99      0.99      0.99       125
   macro avg       0.99      0.97      0.98       125
weighted avg       0.99      0.99      0.99       125
```

# 8 Model 9: LDA-MLP

```
In [43]: trainMlp(*fitLda(*getData()), layers=(100, ))
```

```
trainingData: (375, 1024), trainingLabels: (375,)
testingData: (125, 1024), testingLabels: (125,)


C:\Users\GaryNg\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning:
  warnings.warn("Variables are collinear.")
```





```
trainingData: (375, 3), trainingLabels: (375,)
testingData: (125, 3), testingLabels: (125,)
Iteration 1, loss = 1.24642156
Iteration 2, loss = 1.20766327
Iteration 3, loss = 1.16916265
Iteration 4, loss = 1.13157888
```

```
Iteration 5, loss = 1.09481276
Iteration 6, loss = 1.05887311
Iteration 7, loss = 1.02351347
Iteration 8, loss = 0.98894744
Iteration 9, loss = 0.95526744
Iteration 10, loss = 0.92229549
Iteration 11, loss = 0.89014602
Iteration 12, loss = 0.85843406
Iteration 13, loss = 0.82774935
Iteration 14, loss = 0.79756763
Iteration 15, loss = 0.76816404
Iteration 16, loss = 0.73945753
Iteration 17, loss = 0.71123262
Iteration 18, loss = 0.68390903
Iteration 19, loss = 0.65738788
Iteration 20, loss = 0.63139907
Iteration 21, loss = 0.60608811
Iteration 22, loss = 0.58139568
Iteration 23, loss = 0.55773859
Iteration 24, loss = 0.53451055
Iteration 25, loss = 0.51227390
Iteration 26, loss = 0.49063805
Iteration 27, loss = 0.46971541
Iteration 28, loss = 0.44959529
Iteration 29, loss = 0.42991543
Iteration 30, loss = 0.41124498
Iteration 31, loss = 0.39306782
Iteration 32, loss = 0.37556378
Iteration 33, loss = 0.35869606
Iteration 34, loss = 0.34251960
Iteration 35, loss = 0.32694702
Iteration 36, loss = 0.31220002
Iteration 37, loss = 0.29797791
Iteration 38, loss = 0.28445979
Iteration 39, loss = 0.27151712
Iteration 40, loss = 0.25906555
Iteration 41, loss = 0.24731196
Iteration 42, loss = 0.23600601
Iteration 43, loss = 0.22542229
Iteration 44, loss = 0.21508474
Iteration 45, loss = 0.20555641
Iteration 46, loss = 0.19628320
Iteration 47, loss = 0.18761959
Iteration 48, loss = 0.17937304
Iteration 49, loss = 0.17144706
Iteration 50, loss = 0.16404640
Iteration 51, loss = 0.15699385
Iteration 52, loss = 0.15028594
```

```
Iteration 53, loss = 0.14397628
Iteration 54, loss = 0.13793953
Iteration 55, loss = 0.13228959
Iteration 56, loss = 0.12691102
Iteration 57, loss = 0.12179806
Iteration 58, loss = 0.11695535
Iteration 59, loss = 0.11236139
Iteration 60, loss = 0.10806002
Iteration 61, loss = 0.10392528
Iteration 62, loss = 0.10003751
Iteration 63, loss = 0.09633711
Iteration 64, loss = 0.09275747
Iteration 65, loss = 0.08947118
Iteration 66, loss = 0.08631043
Iteration 67, loss = 0.08327735
Iteration 68, loss = 0.08041043
Iteration 69, loss = 0.07769071
Iteration 70, loss = 0.07506908
Iteration 71, loss = 0.07260748
Iteration 72, loss = 0.07025615
Iteration 73, loss = 0.06800326
Iteration 74, loss = 0.06587353
Iteration 75, loss = 0.06383621
Iteration 76, loss = 0.06187619
Iteration 77, loss = 0.06000821
Iteration 78, loss = 0.05823968
Iteration 79, loss = 0.05652664
Iteration 80, loss = 0.05490243
Iteration 81, loss = 0.05333078
Iteration 82, loss = 0.05183574
Iteration 83, loss = 0.05038837
Iteration 84, loss = 0.04900651
Iteration 85, loss = 0.04766006
Iteration 86, loss = 0.04637817
Iteration 87, loss = 0.04512090
Iteration 88, loss = 0.04392167
Iteration 89, loss = 0.04275499
Iteration 90, loss = 0.04163698
Iteration 91, loss = 0.04054665
Iteration 92, loss = 0.03952381
Iteration 93, loss = 0.03852174
Iteration 94, loss = 0.03755340
Iteration 95, loss = 0.03663458
Iteration 96, loss = 0.03574419
Iteration 97, loss = 0.03488530
Iteration 98, loss = 0.03405849
Iteration 99, loss = 0.03325895
Iteration 100, loss = 0.03249614
```

```
Iteration 101, loss = 0.03175111
Iteration 102, loss = 0.03104091
Iteration 103, loss = 0.03033543
Iteration 104, loss = 0.02967610
Iteration 105, loss = 0.02903085
Iteration 106, loss = 0.02839986
Iteration 107, loss = 0.02780152
Iteration 108, loss = 0.02721441
Iteration 109, loss = 0.02663707
Iteration 110, loss = 0.02609297
Iteration 111, loss = 0.02556145
Iteration 112, loss = 0.02504154
Iteration 113, loss = 0.02454590
Iteration 114, loss = 0.02405318
Iteration 115, loss = 0.02358003
Iteration 116, loss = 0.02312670
Iteration 117, loss = 0.02267615
Iteration 118, loss = 0.02225061
Iteration 119, loss = 0.02182595
Iteration 120, loss = 0.02142284
Iteration 121, loss = 0.02101833
Iteration 122, loss = 0.02063882
Iteration 123, loss = 0.02025978
Iteration 124, loss = 0.01989246
Iteration 125, loss = 0.01953941
Iteration 126, loss = 0.01919579
Iteration 127, loss = 0.01885874
Iteration 128, loss = 0.01852950
Iteration 129, loss = 0.01820935
Iteration 130, loss = 0.01790113
Iteration 131, loss = 0.01759064
Iteration 132, loss = 0.01730091
Iteration 133, loss = 0.01700965
Iteration 134, loss = 0.01672813
Iteration 135, loss = 0.01645865
Iteration 136, loss = 0.01618627
Iteration 137, loss = 0.01592477
Iteration 138, loss = 0.01567023
Iteration 139, loss = 0.01542063
Iteration 140, loss = 0.01517890
Iteration 141, loss = 0.01494318
Iteration 142, loss = 0.01470914
Iteration 143, loss = 0.01448285
Iteration 144, loss = 0.01426301
Iteration 145, loss = 0.01404559
Iteration 146, loss = 0.01383544
Iteration 147, loss = 0.01362795
Iteration 148, loss = 0.01342539
```

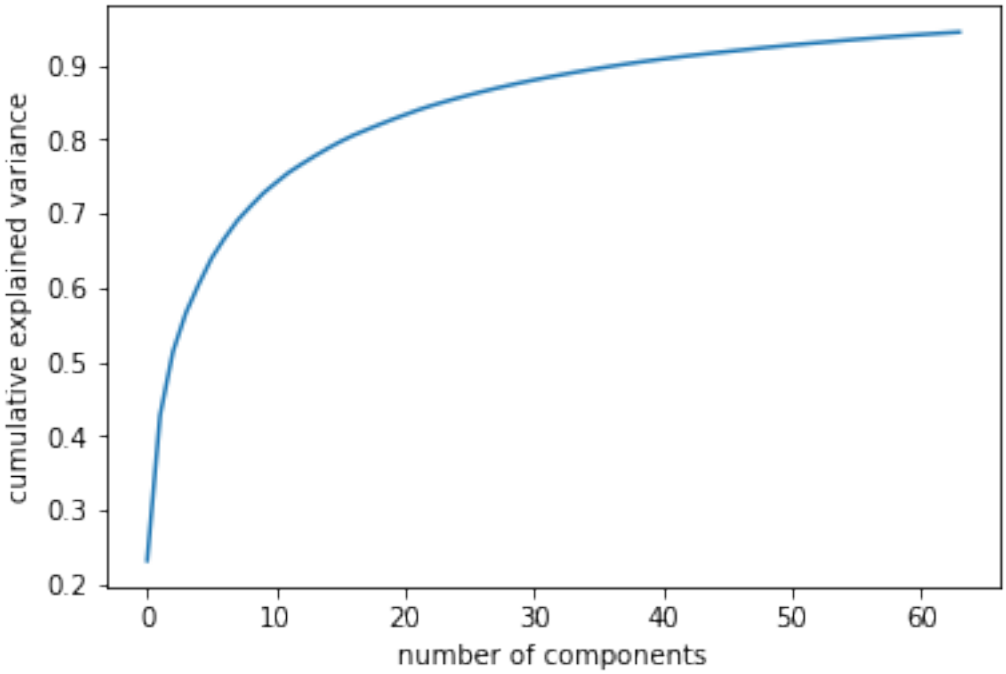
```
Iteration 149, loss = 0.01322640
Iteration 150, loss = 0.01303662
Iteration 151, loss = 0.01284556
Iteration 152, loss = 0.01266305
Iteration 153, loss = 0.01247950
Iteration 154, loss = 0.01230014
Iteration 155, loss = 0.01213018
Iteration 156, loss = 0.01195982
Iteration 157, loss = 0.01179521
Iteration 158, loss = 0.01163075
Iteration 159, loss = 0.01147182
Iteration 160, loss = 0.01131513
Iteration 161, loss = 0.01116506
Iteration 162, loss = 0.01101393
Iteration 163, loss = 0.01086719
Iteration 164, loss = 0.01072226
Iteration 165, loss = 0.01058251
Iteration 166, loss = 0.01044402
Iteration 167, loss = 0.01030851
Iteration 168, loss = 0.01017738
Iteration 169, loss = 0.01004415
Iteration 170, loss = 0.00991845
Iteration 171, loss = 0.00979223
Iteration 172, loss = 0.00967064
Iteration 173, loss = 0.00954967
Iteration 174, loss = 0.00943072
Iteration 175, loss = 0.00931595
Iteration 176, loss = 0.00920128
Iteration 177, loss = 0.00908997
Iteration 178, loss = 0.00897981
Iteration 179, loss = 0.00887083
Iteration 180, loss = 0.00876653
Iteration 181, loss = 0.00866169
Iteration 182, loss = 0.00856083
Iteration 183, loss = 0.00845872
Iteration 184, loss = 0.00836224
Iteration 185, loss = 0.00826380
Iteration 186, loss = 0.00816953
Iteration 187, loss = 0.00807536
Iteration 188, loss = 0.00798486
Iteration 189, loss = 0.00789246
Iteration 190, loss = 0.00780404
Iteration 191, loss = 0.00771887
Iteration 192, loss = 0.00763099
Iteration 193, loss = 0.00754850
Iteration 194, loss = 0.00746544
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
```

```
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.976
confusion matrix:
 [[ 7  0  0  1]
 [ 0 22  1  0]
 [ 1  0 66  0]
 [ 0  0  0 27]]
              precision    recall  f1-score   support

           0       0.88      0.88      0.88         8
           1       1.00      0.96      0.98        23
           2       0.99      0.99      0.99        67
           3       0.96      1.00      0.98        27

   micro avg       0.98      0.98      0.98       125
   macro avg       0.96      0.95      0.95       125
weighted avg       0.98      0.98      0.98       125
```

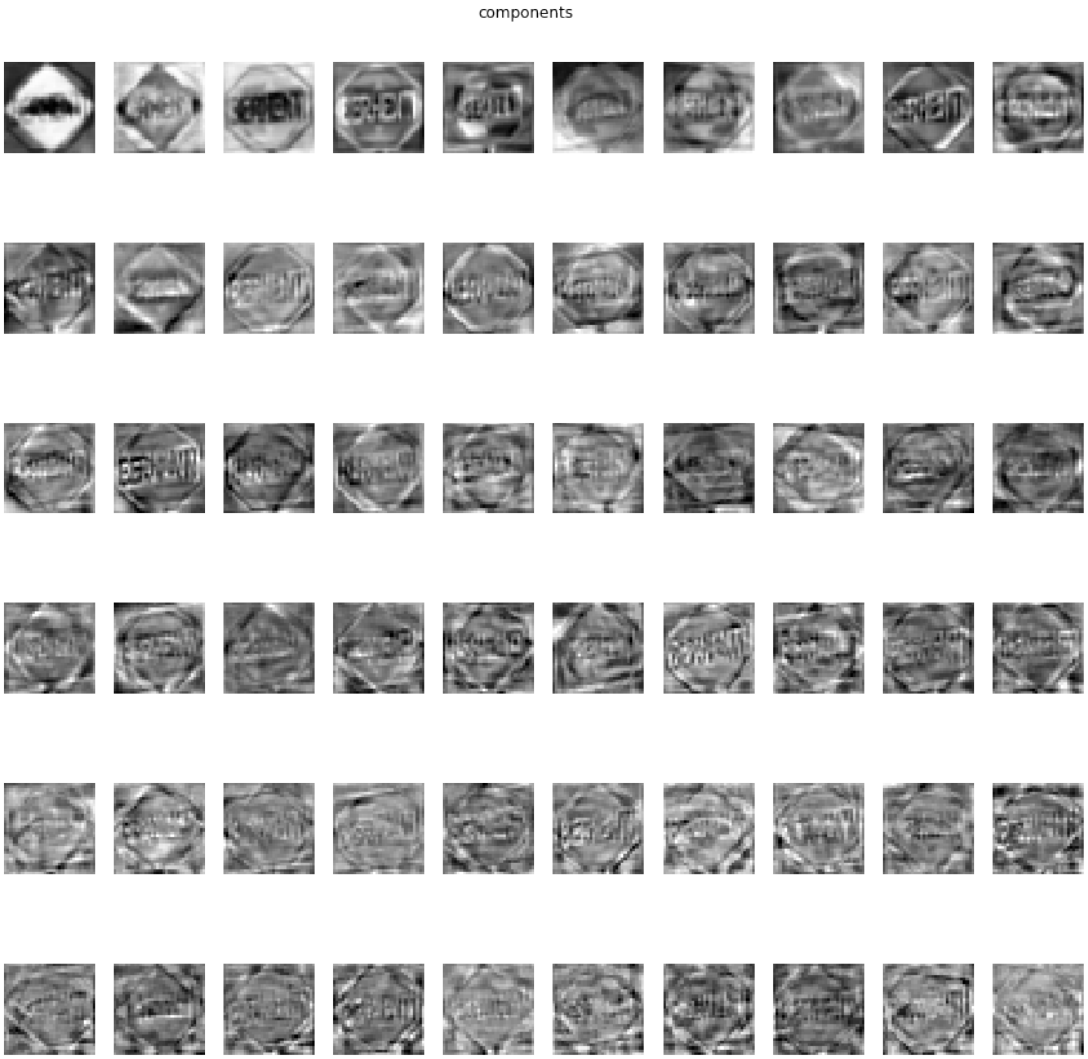
# 9 Model 10: PCA-LDA-MLP

```
In [44]: trainMlp(*fitLda(*fitPca(*getData())))
```

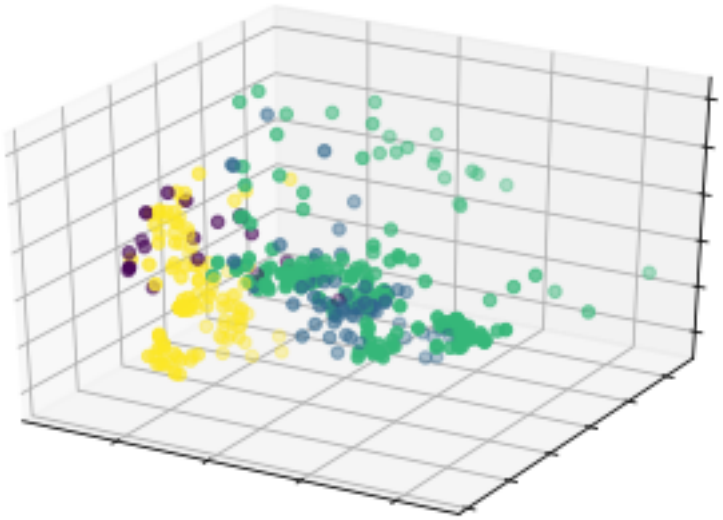
```
trainingData: (375, 1024), trainingLabels: (375,)
testingData: (125, 1024), testingLabels: (125,)
total explained variance: 0.9443220201975157
```

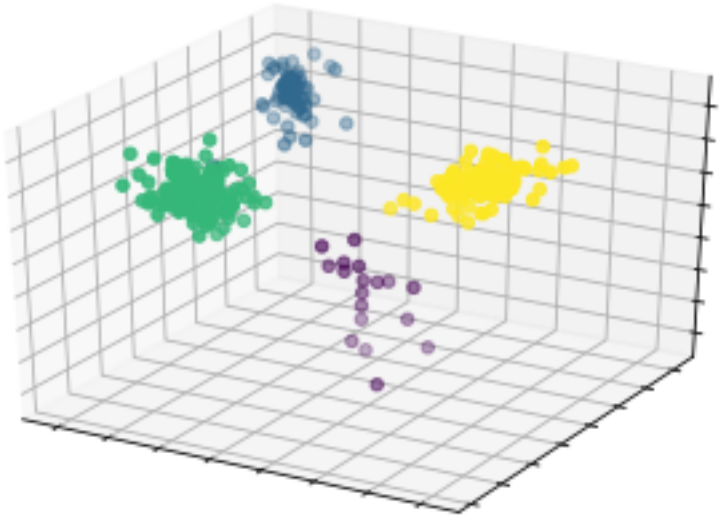



```
pca components: (64, 1024)
```




```
trainingData: (375, 64), trainingLabels: (375,)
testingData: (125, 64), testingLabels: (125,)
```





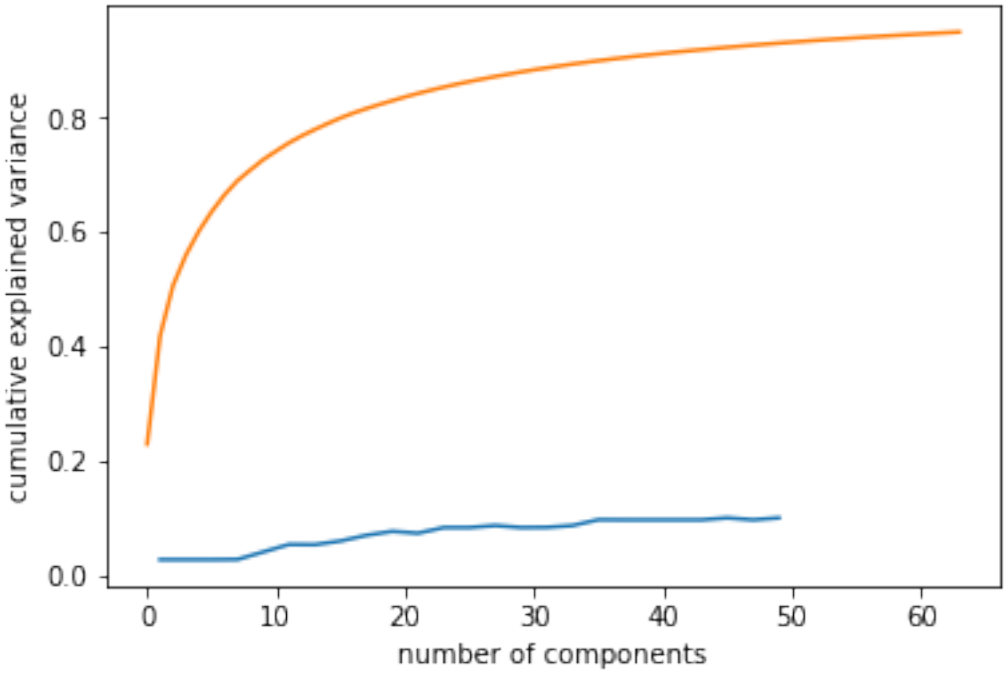
```
trainingData: (375, 3), trainingLabels: (375,)
testingData: (125, 3), testingLabels: (125,)
Iteration 1, loss = 1.20979992
Iteration 2, loss = 0.58948199
Iteration 3, loss = 0.26138793
Iteration 4, loss = 0.10523986
```

```
Iteration 5, loss = 0.04075081
Iteration 6, loss = 0.01760821
Iteration 7, loss = 0.00872802
Iteration 8, loss = 0.00467426
Iteration 9, loss = 0.00296332
Iteration 10, loss = 0.00196047
Iteration 11, loss = 0.00152160
Iteration 12, loss = 0.00116707
Iteration 13, loss = 0.00098555
Iteration 14, loss = 0.00084280
Iteration 15, loss = 0.00075269
Iteration 16, loss = 0.00069382
Iteration 17, loss = 0.00064032
Iteration 18, loss = 0.00060589
Iteration 19, loss = 0.00057353
Iteration 20, loss = 0.00055340
Iteration 21, loss = 0.00053497
Iteration 22, loss = 0.00052072
Iteration 23, loss = 0.00050604
Iteration 24, loss = 0.00049691
Iteration 25, loss = 0.00048769
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.992
confusion matrix:
 [[ 7  0  0  1]
 [ 0 23  0  0]
 [ 0  0 67  0]
 [ 0  0  0 27]]
              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       1.00      1.00      1.00        23
           2       1.00      1.00      1.00        67
           3       0.96      1.00      0.98        27

   micro avg       0.99      0.99      0.99       125
   macro avg       0.99      0.97      0.98       125
weighted avg       0.99      0.99      0.99       125
```

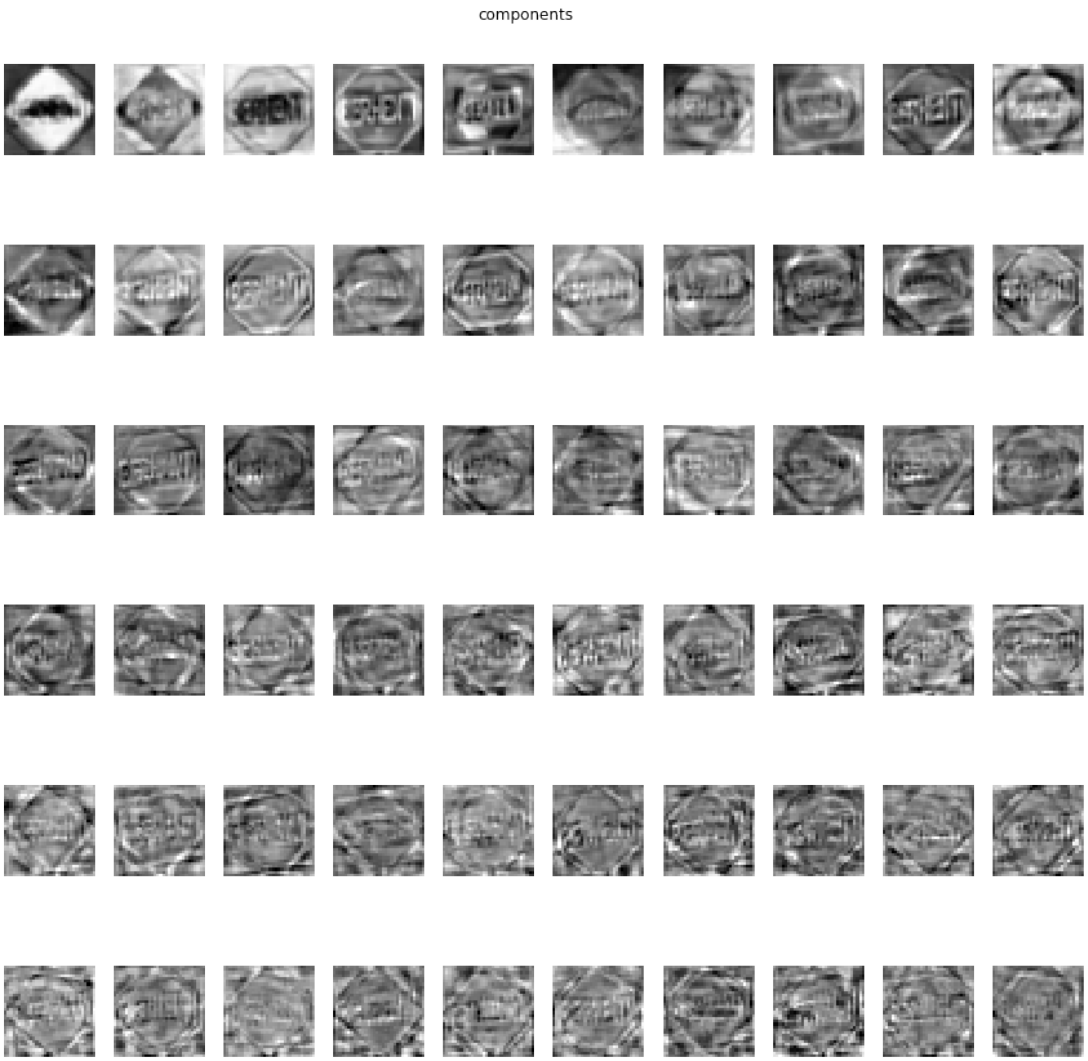
## 10 Benchmark

```
In [45]: def runAll(testSize = 0.25):
             data = train_test_split(*unpickleData(), random_state=123, test_size=testSize)
             print("--- Model 1 ---")
             trainKnn(*data)
             print("--- Model 2 ---")
             trainKnn(*fitPca(*data))
             print("--- Model 3 ---")
             trainKnn(*fitLda(*data))
             print("--- Model 4 ---")
             trainKnn(*fitLda(*fitPca(*data)))
             print("--- Model 5 ---")
             trainSvm(*fitPca(*data))
             print("--- Model 6 ---")
             trainSvm(*fitLda(*fitPca(*data)))
             print("--- Model 7 ---")
             trainMlp(*data)
             print("--- Model 8 ---")
             trainMlp(*fitPca(*data), layers=(64, 64))
             print("--- Model 9 ---")
             trainMlp(*fitLda(*data), layers=(100, ))
             print("--- Model 10 ---")
             trainMlp(*fitLda(*fitPca(*data)))

In [46]: runAll(testSize=0.4)

--- Model 1 ---
trainingData: (300, 1024), trainingLabels: (300,)
testingData: (200, 1024), testingLabels: (200,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.99
confusion matrix:
 [[ 10   0   0   1]
 [  1  30   0   0]
 [  0   0 101   0]
 [  0   0   0  57]]
              precision    recall  f1-score   support

           0       0.91      0.91      0.91        11
           1       1.00      0.97      0.98        31
           2       1.00      1.00      1.00       101
           3       0.98      1.00      0.99        57

   micro avg       0.99      0.99      0.99       200
   macro avg       0.97      0.97      0.97       200
weighted avg       0.99      0.99      0.99       200
```

```
--- Model 2 ---
trainingData: (300, 1024), trainingLabels: (300,)
testingData: (200, 1024), testingLabels: (200,)
total explained variance: 0.9493939935834859
```

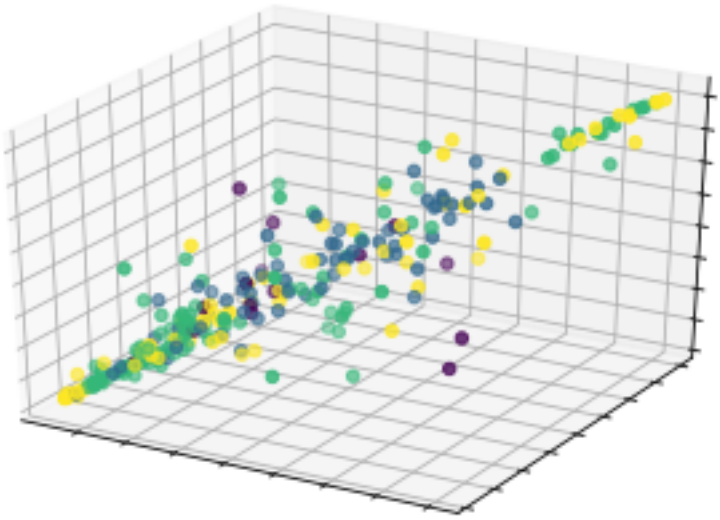


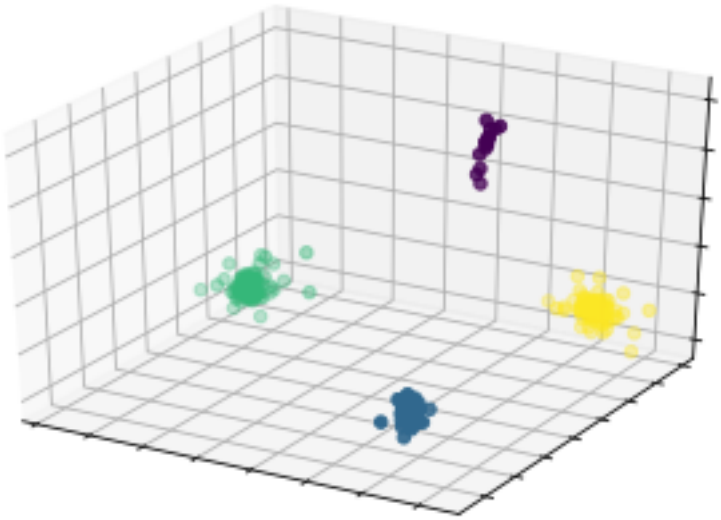

```
pca components: (64, 1024)
```



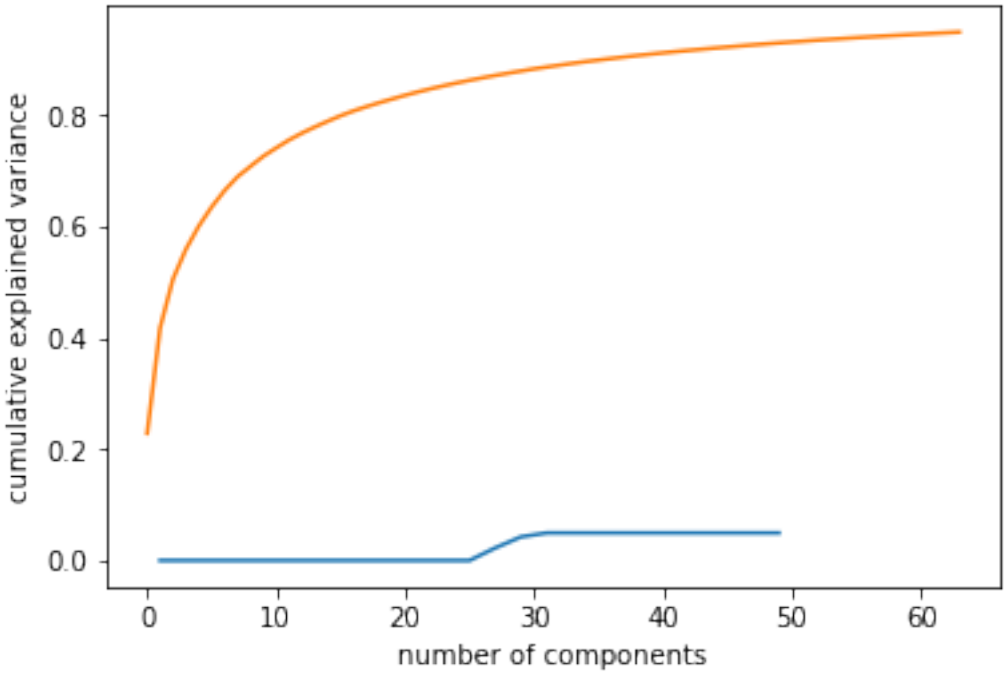

```
trainingData: (300, 64), trainingLabels: (300,)
testingData: (200, 64), testingLabels: (200,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.975
confusion matrix:
 [[ 10   0   0   1]
 [  0  29   2   0]
 [  0   0 100   1]
 [  0   0   1  56]]
              precision    recall  f1-score   support

           0       1.00      0.91      0.95        11
           1       1.00      0.94      0.97        31
```

```
           2       0.97      0.99      0.98       101
           3       0.97      0.98      0.97        57

   micro avg       0.97      0.97      0.97       200
   macro avg       0.98      0.95      0.97       200
weighted avg       0.98      0.97      0.97       200

--- Model 3 ---
trainingData: (300, 1024), trainingLabels: (300,)
testingData: (200, 1024), testingLabels: (200,)


C:\Users\GaryNg\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning:
  warnings.warn("Variables are collinear.")
C:\Users\GaryNg\Anaconda3\lib\site-packages\matplotlib\figure.py:98: MatplotlibDeprecationWarni
Adding an axes using the same arguments as a previous axes currently reuses the earlier instanc
  "Adding an axes using the same arguments as a previous axes "
```

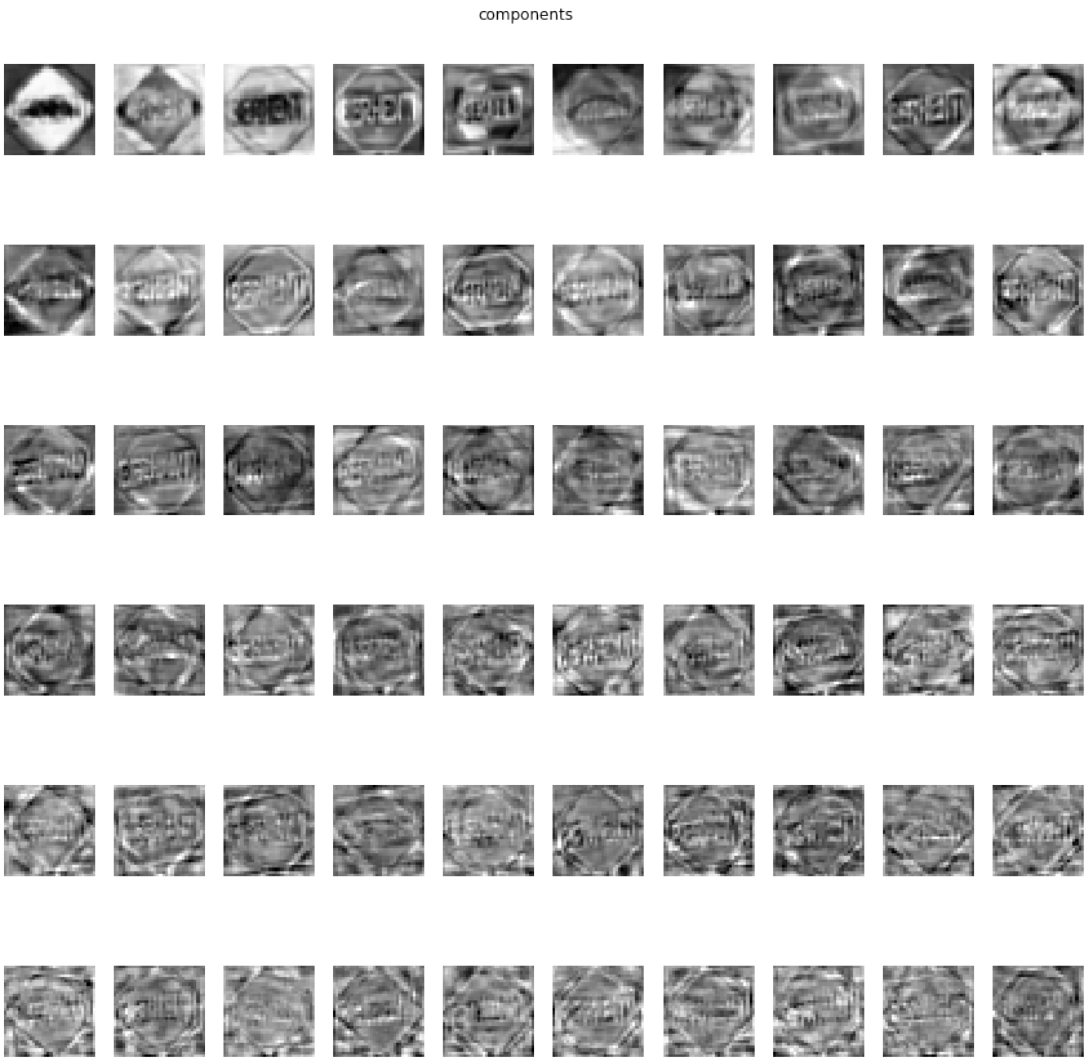




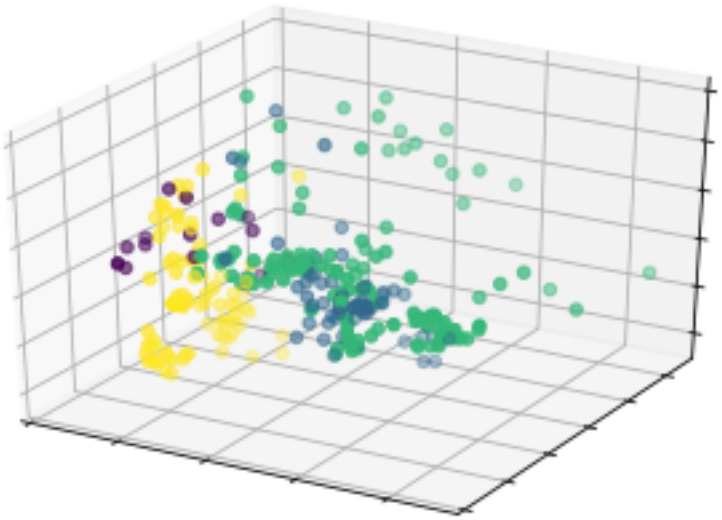
```
trainingData: (300, 3), trainingLabels: (300,)
testingData: (200, 3), testingLabels: (200,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.97
confusion matrix:
 [[10  0  1  0]
 [ 0 31  0  0]
 [ 0  1 98  2]
 [ 1  0  1 55]]
              precision    recall  f1-score   support

           0       0.91      0.91      0.91        11
           1       0.97      1.00      0.98        31
           2       0.98      0.97      0.98       101
           3       0.96      0.96      0.96        57

   micro avg       0.97      0.97      0.97       200
   macro avg       0.96      0.96      0.96       200
weighted avg       0.97      0.97      0.97       200

--- Model 4 ---
trainingData: (300, 1024), trainingLabels: (300,)
testingData: (200, 1024), testingLabels: (200,)
total explained variance: 0.9494124846073411
```

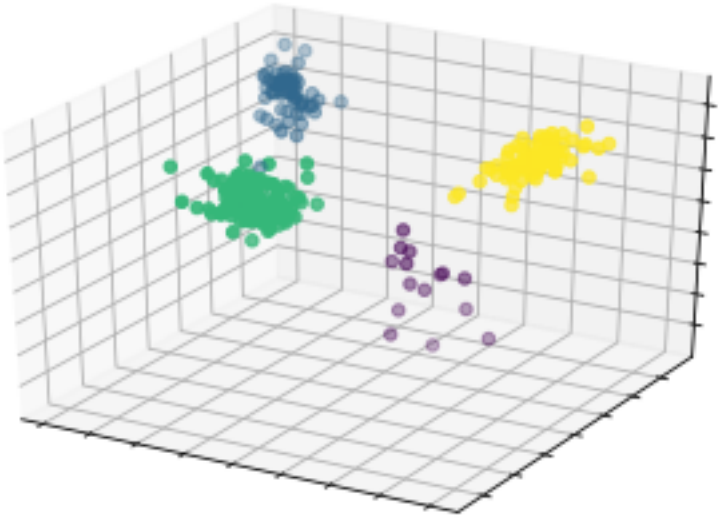



```
pca components: (64, 1024)
```



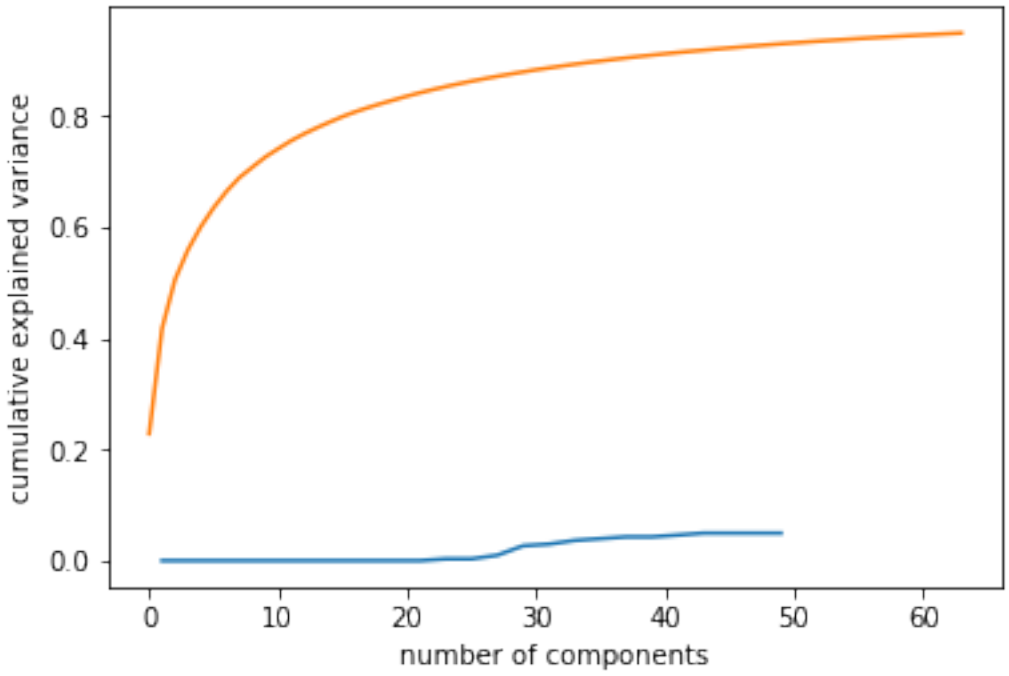

```
trainingData: (300, 64), trainingLabels: (300,)
testingData: (200, 64), testingLabels: (200,)
```



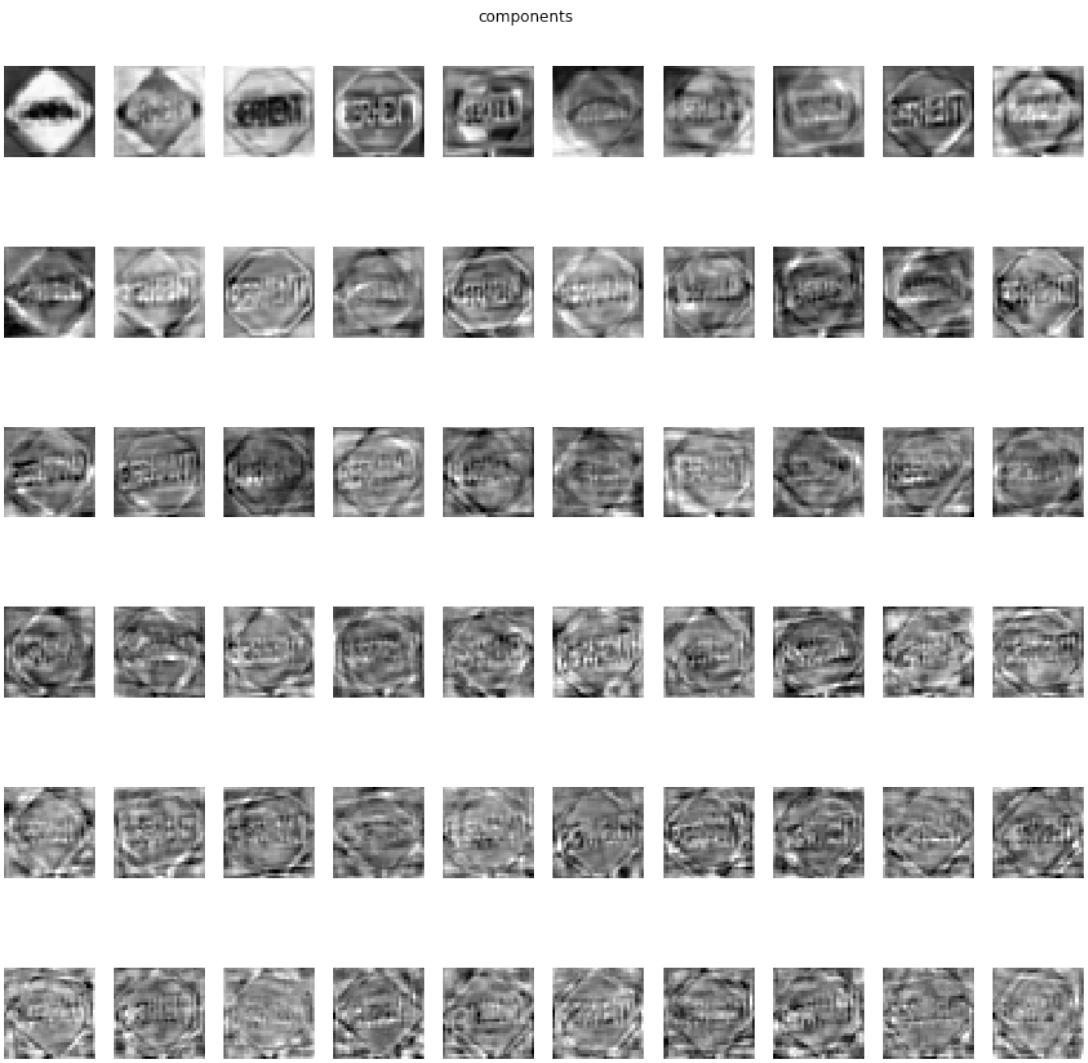


```
trainingData: (300, 3), trainingLabels: (300,)
testingData: (200, 3), testingLabels: (200,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.985
confusion matrix:
```

```
 [[10  0  0  1]
 [ 0 31  0  0]
 [ 0  2 99  0]
 [ 0  0  0 57]]
              precision    recall  f1-score   support

           0       1.00      0.91      0.95        11
           1       0.94      1.00      0.97        31
           2       1.00      0.98      0.99       101
           3       0.98      1.00      0.99        57

   micro avg       0.98      0.98      0.98       200
   macro avg       0.98      0.97      0.98       200
weighted avg       0.99      0.98      0.99       200

--- Model 5 ---
trainingData: (300, 1024), trainingLabels: (300,)
testingData: (200, 1024), testingLabels: (200,)
total explained variance: 0.9494429553419992
```

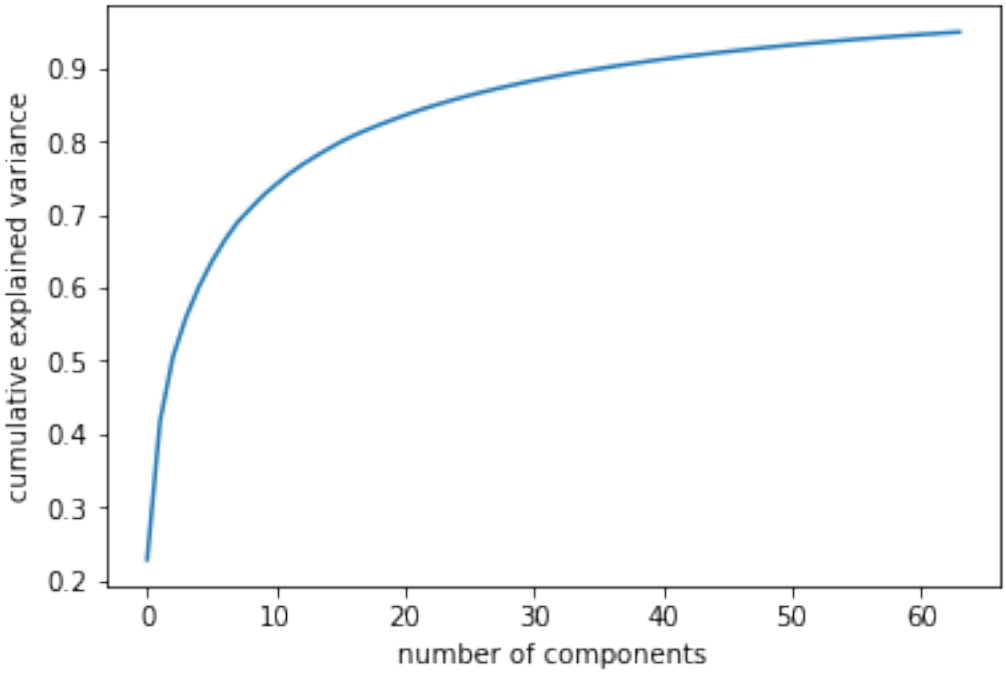



```
pca components: (64, 1024)
```

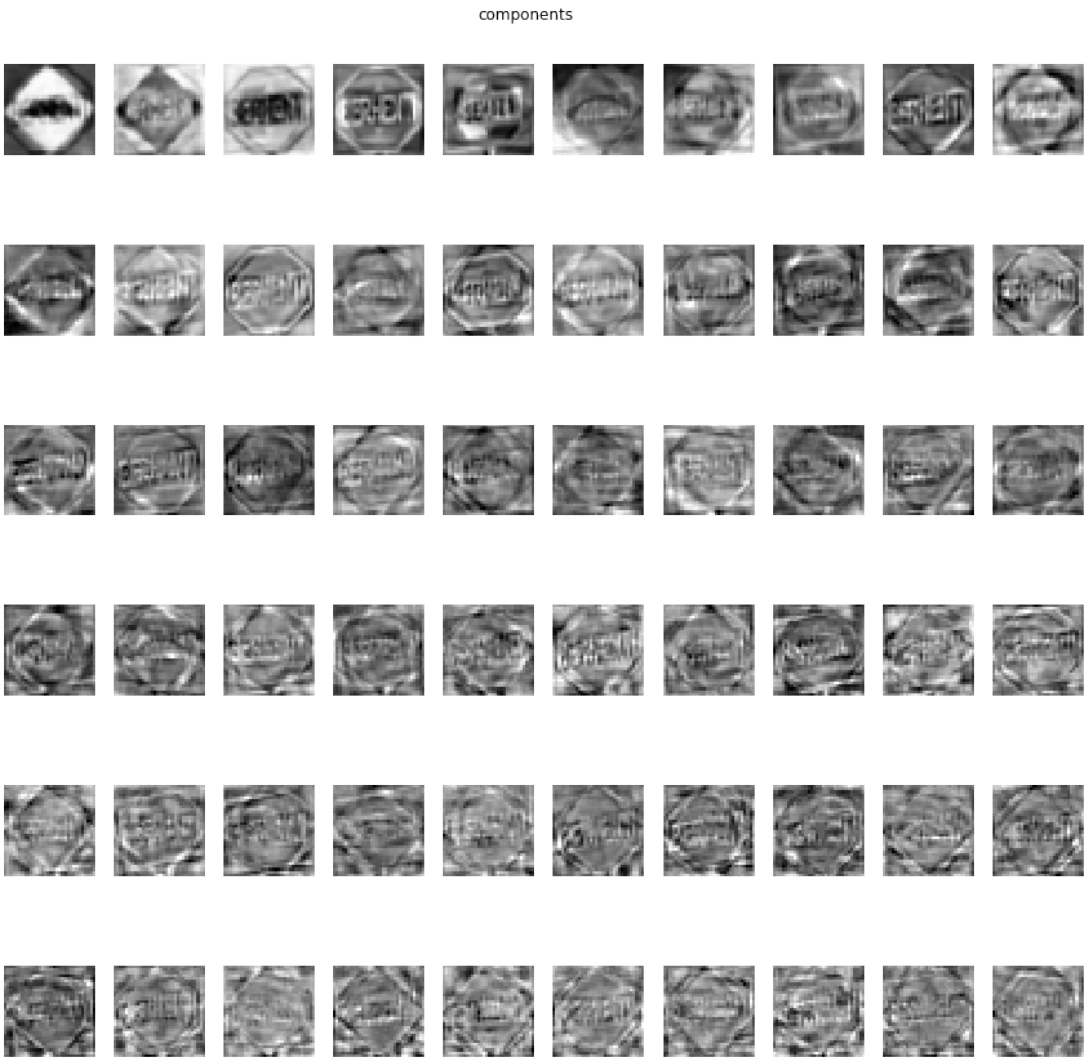



```
trainingData: (300, 64), trainingLabels: (300,)
testingData: (200, 64), testingLabels: (200,)
training accuracy: 1.0
testing accuracy: 0.99
confusion matrix:
 [[ 10   0   0   1]
 [  0  31   0   0]
 [  0   1 100   0]
 [  0   0   0  57]]
              precision    recall  f1-score   support

           0       1.00      0.91      0.95        11
           1       0.97      1.00      0.98        31
           2       1.00      0.99      1.00       101
```

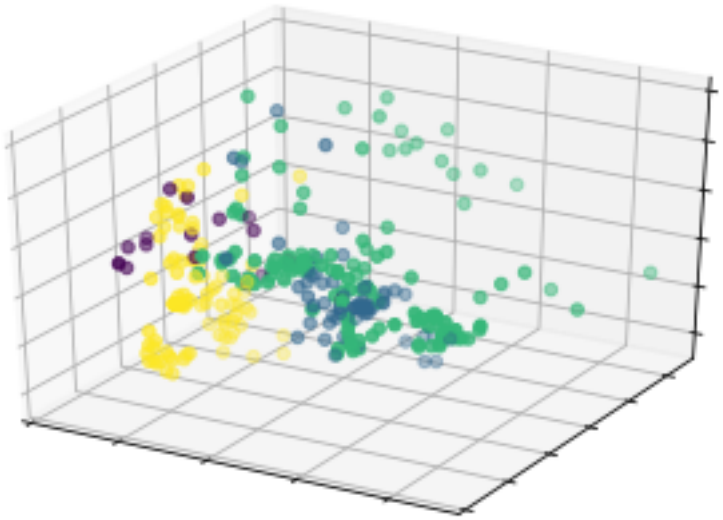
```
           3       0.98      1.00      0.99        57

   micro avg       0.99      0.99      0.99       200
   macro avg       0.99      0.97      0.98       200
weighted avg       0.99      0.99      0.99       200

--- Model 6 ---
trainingData: (300, 1024), trainingLabels: (300,)
testingData: (200, 1024), testingLabels: (200,)
total explained variance: 0.9494058238317254
```

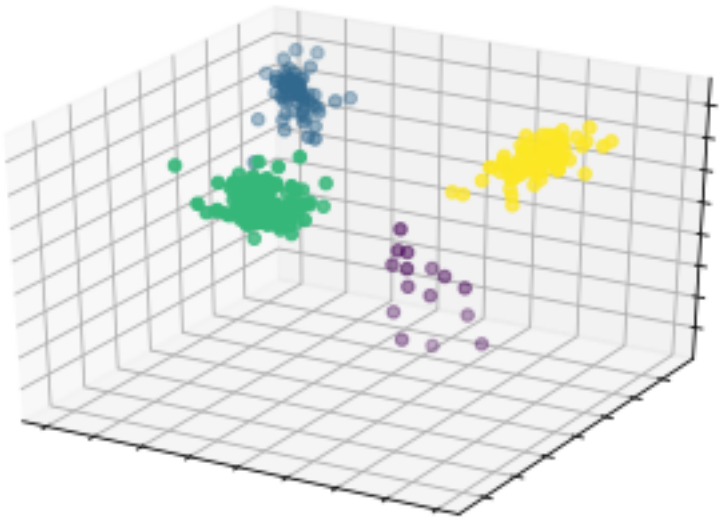



```
pca components: (64, 1024)
```




```
trainingData: (300, 64), trainingLabels: (300,)
testingData: (200, 64), testingLabels: (200,)
```

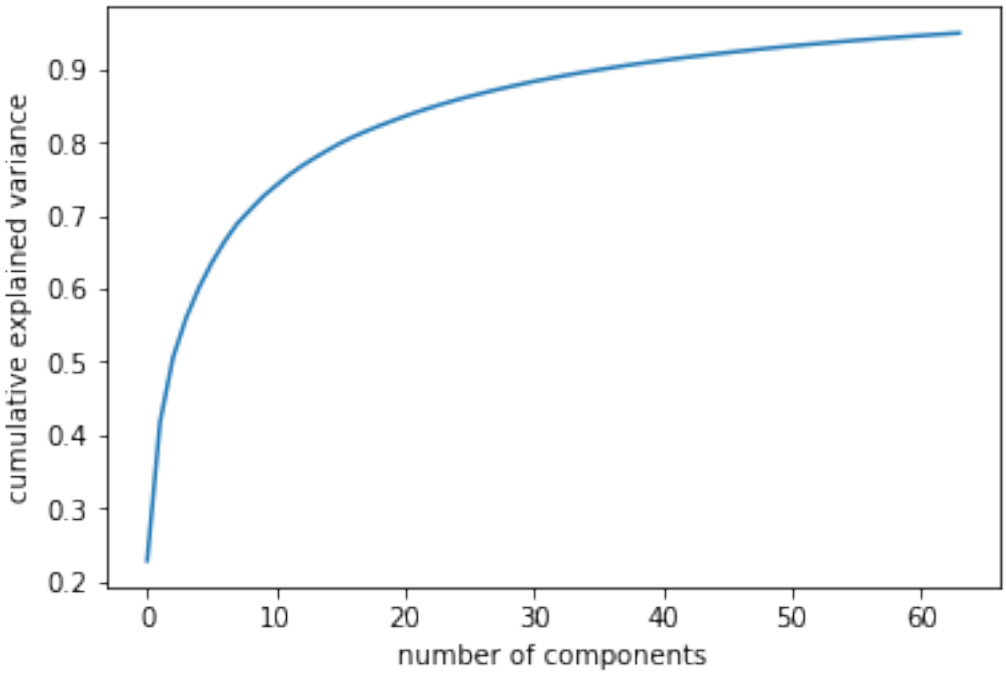




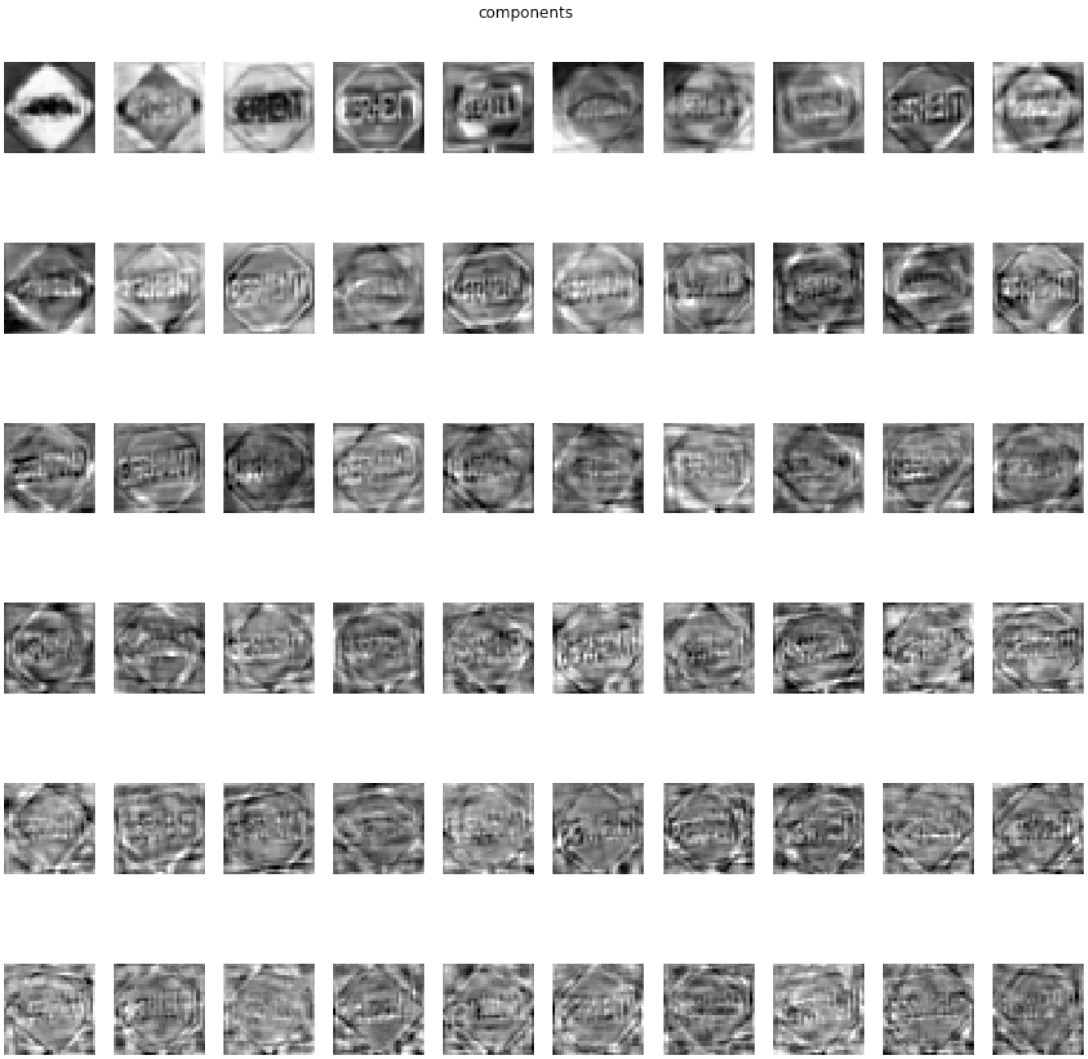
```
trainingData: (300, 3), trainingLabels: (300,)
testingData: (200, 3), testingLabels: (200,)
training accuracy: 1.0
testing accuracy: 0.985
confusion matrix:
 [[10  0  0  1]
```

```
 [ 0 31  0  0]
 [ 0  2 99  0]
 [ 0  0  0 57]]
              precision    recall  f1-score   support

           0       1.00      0.91      0.95        11
           1       0.94      1.00      0.97        31
           2       1.00      0.98      0.99       101
           3       0.98      1.00      0.99        57

   micro avg       0.98      0.98      0.98       200
   macro avg       0.98      0.97      0.98       200
weighted avg       0.99      0.98      0.99       200

--- Model 7 ---
trainingData: (300, 1024), trainingLabels: (300,)
testingData: (200, 1024), testingLabels: (200,)
Iteration 1, loss = 1.13716667
Iteration 2, loss = 0.10960731
Iteration 3, loss = 0.05273263
Iteration 4, loss = 0.00718787
Iteration 5, loss = 0.01229409
Iteration 6, loss = 0.01133413
Iteration 7, loss = 0.00202285
Iteration 8, loss = 0.00109558
Iteration 9, loss = 0.00095829
Iteration 10, loss = 0.00088808
Iteration 11, loss = 0.00084624
Iteration 12, loss = 0.00083289
Iteration 13, loss = 0.00085575
Iteration 14, loss = 0.00087496
Iteration 15, loss = 0.00087158
Iteration 16, loss = 0.00083209
Iteration 17, loss = 0.00080356
Iteration 18, loss = 0.00076530
Iteration 19, loss = 0.00074590
Iteration 20, loss = 0.00073502
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.995
confusion matrix:
```

```
 [[ 10   0   0   1]
 [  0  31   0   0]
 [  0   0 101   0]
 [  0   0   0  57]]
              precision    recall  f1-score   support

           0       1.00      0.91      0.95        11
           1       1.00      1.00      1.00        31
           2       1.00      1.00      1.00       101
           3       0.98      1.00      0.99        57

   micro avg       0.99      0.99      0.99       200
   macro avg       1.00      0.98      0.99       200
weighted avg       1.00      0.99      0.99       200

--- Model 8 ---
trainingData: (300, 1024), trainingLabels: (300,)
testingData: (200, 1024), testingLabels: (200,)
total explained variance: 0.9494106495359718
```




```
pca components: (64, 1024)
```

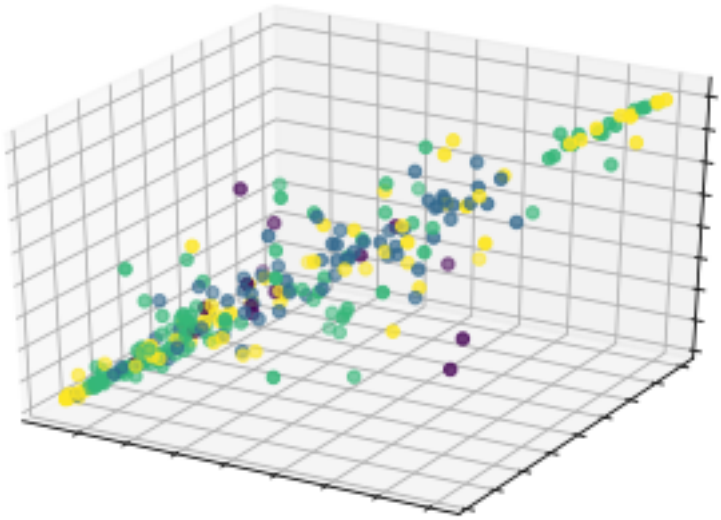


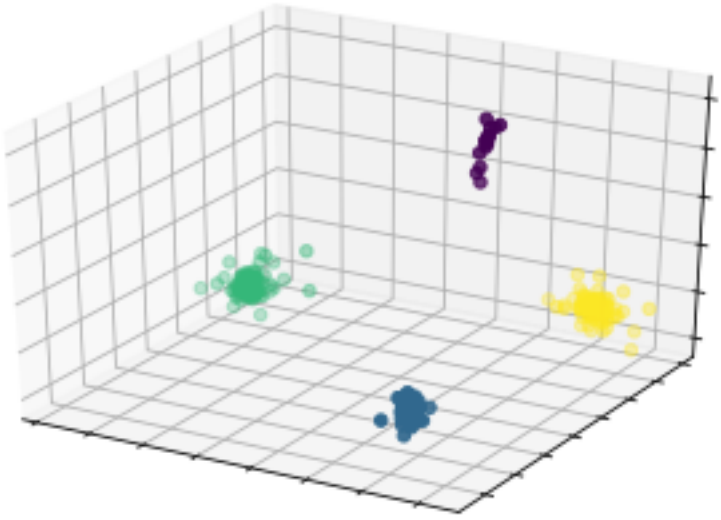

```
trainingData: (300, 64), trainingLabels: (300,)
testingData: (200, 64), testingLabels: (200,)
Iteration 1, loss = 1.60566376
Iteration 2, loss = 1.49075472
Iteration 3, loss = 1.39373341
Iteration 4, loss = 1.30935800
Iteration 5, loss = 1.23531317
Iteration 6, loss = 1.17347504
Iteration 7, loss = 1.11692695
Iteration 8, loss = 1.06494100
Iteration 9, loss = 1.01760514
Iteration 10, loss = 0.97106549
Iteration 11, loss = 0.92709832
Iteration 12, loss = 0.88400635
```

```
Iteration 13, loss = 0.84198877
Iteration 14, loss = 0.80011143
Iteration 15, loss = 0.75905628
Iteration 16, loss = 0.71872644
Iteration 17, loss = 0.67853195
Iteration 18, loss = 0.63979273
Iteration 19, loss = 0.60123722
Iteration 20, loss = 0.56335036
Iteration 21, loss = 0.52709525
Iteration 22, loss = 0.49075217
Iteration 23, loss = 0.45647921
Iteration 24, loss = 0.42270203
Iteration 25, loss = 0.39068038
Iteration 26, loss = 0.36008183
Iteration 27, loss = 0.33097560
Iteration 28, loss = 0.30369758
Iteration 29, loss = 0.27792359
Iteration 30, loss = 0.25397019
Iteration 31, loss = 0.23167234
Iteration 32, loss = 0.21039619
Iteration 33, loss = 0.19136004
Iteration 34, loss = 0.17336650
Iteration 35, loss = 0.15707327
Iteration 36, loss = 0.14229047
Iteration 37, loss = 0.12874953
Iteration 38, loss = 0.11636665
Iteration 39, loss = 0.10539411
Iteration 40, loss = 0.09564226
Iteration 41, loss = 0.08663478
Iteration 42, loss = 0.07878206
Iteration 43, loss = 0.07171433
Iteration 44, loss = 0.06549896
Iteration 45, loss = 0.05969820
Iteration 46, loss = 0.05477251
Iteration 47, loss = 0.05046767
Iteration 48, loss = 0.04633927
Iteration 49, loss = 0.04285809
Iteration 50, loss = 0.03975757
Iteration 51, loss = 0.03689851
Iteration 52, loss = 0.03437601
Iteration 53, loss = 0.03210220
Iteration 54, loss = 0.02999449
Iteration 55, loss = 0.02816004
Iteration 56, loss = 0.02646826
Iteration 57, loss = 0.02492110
Iteration 58, loss = 0.02355993
Iteration 59, loss = 0.02227131
Iteration 60, loss = 0.02108164
```

```
Iteration 61, loss = 0.01999954
Iteration 62, loss = 0.01900513
Iteration 63, loss = 0.01809087
Iteration 64, loss = 0.01725270
Iteration 65, loss = 0.01648161
Iteration 66, loss = 0.01573206
Iteration 67, loss = 0.01506545
Iteration 68, loss = 0.01444250
Iteration 69, loss = 0.01385011
Iteration 70, loss = 0.01329884
Iteration 71, loss = 0.01279254
Iteration 72, loss = 0.01230763
Iteration 73, loss = 0.01184708
Iteration 74, loss = 0.01142720
Iteration 75, loss = 0.01103161
Iteration 76, loss = 0.01063843
Iteration 77, loss = 0.01028017
Iteration 78, loss = 0.00994333
Iteration 79, loss = 0.00961851
Iteration 80, loss = 0.00930965
Iteration 81, loss = 0.00902769
Iteration 82, loss = 0.00874248
Iteration 83, loss = 0.00848153
Iteration 84, loss = 0.00823276
Iteration 85, loss = 0.00799648
Iteration 86, loss = 0.00776857
Iteration 87, loss = 0.00755245
Iteration 88, loss = 0.00734521
Iteration 89, loss = 0.00714907
Iteration 90, loss = 0.00695969
Iteration 91, loss = 0.00677411
Iteration 92, loss = 0.00660197
Iteration 93, loss = 0.00643783
Iteration 94, loss = 0.00627188
Iteration 95, loss = 0.00612008
Iteration 96, loss = 0.00597264
Iteration 97, loss = 0.00582695
Iteration 98, loss = 0.00568501
Iteration 99, loss = 0.00555605
Iteration 100, loss = 0.00542299
Iteration 101, loss = 0.00529955
Iteration 102, loss = 0.00518192
Iteration 103, loss = 0.00506494
Iteration 104, loss = 0.00495513
Iteration 105, loss = 0.00484396
Iteration 106, loss = 0.00474109
Iteration 107, loss = 0.00464147
Iteration 108, loss = 0.00454420
```

```
Iteration 109, loss = 0.00444782
Iteration 110, loss = 0.00435767
Iteration 111, loss = 0.00427081
Iteration 112, loss = 0.00418675
Iteration 113, loss = 0.00410287
Iteration 114, loss = 0.00402362
Iteration 115, loss = 0.00394600
Iteration 116, loss = 0.00387035
Iteration 117, loss = 0.00379966
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(64, 64), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.99
confusion matrix:
 [[ 10   0   0   1]
 [  0  31   0   0]
 [  0   1 100   0]
 [  0   0   0  57]]
              precision    recall  f1-score   support

           0       1.00      0.91      0.95        11
           1       0.97      1.00      0.98        31
           2       1.00      0.99      1.00       101
           3       0.98      1.00      0.99        57

   micro avg       0.99      0.99      0.99       200
   macro avg       0.99      0.97      0.98       200
weighted avg       0.99      0.99      0.99       200

--- Model 9 ---
trainingData: (300, 1024), trainingLabels: (300,)
testingData: (200, 1024), testingLabels: (200,)


C:\Users\GaryNg\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning:
  warnings.warn("Variables are collinear.")
```





```
trainingData: (300, 3), trainingLabels: (300,)
testingData: (200, 3), testingLabels: (200,)
Iteration 1, loss = 1.39807546
Iteration 2, loss = 1.35457928
Iteration 3, loss = 1.31210484
Iteration 4, loss = 1.27113602
```

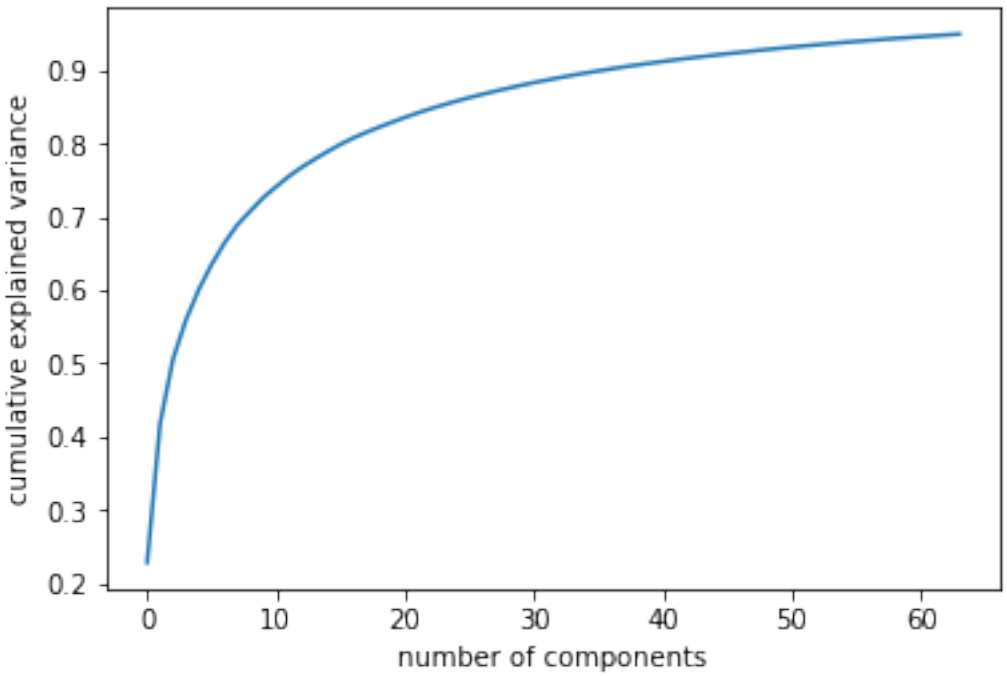
```
Iteration 5, loss = 1.23073532
Iteration 6, loss = 1.19153544
Iteration 7, loss = 1.15270211
Iteration 8, loss = 1.11557849
Iteration 9, loss = 1.07905612
Iteration 10, loss = 1.04329962
Iteration 11, loss = 1.00837235
Iteration 12, loss = 0.97417124
Iteration 13, loss = 0.94080806
Iteration 14, loss = 0.90847124
Iteration 15, loss = 0.87648789
Iteration 16, loss = 0.84547598
Iteration 17, loss = 0.81520169
Iteration 18, loss = 0.78558578
Iteration 19, loss = 0.75659836
Iteration 20, loss = 0.72828706
Iteration 21, loss = 0.70063733
Iteration 22, loss = 0.67394733
Iteration 23, loss = 0.64755523
Iteration 24, loss = 0.62217512
Iteration 25, loss = 0.59736733
Iteration 26, loss = 0.57331306
Iteration 27, loss = 0.55010978
Iteration 28, loss = 0.52746101
Iteration 29, loss = 0.50550507
Iteration 30, loss = 0.48441681
Iteration 31, loss = 0.46393550
Iteration 32, loss = 0.44425737
Iteration 33, loss = 0.42519548
Iteration 34, loss = 0.40689112
Iteration 35, loss = 0.38934709
Iteration 36, loss = 0.37240232
Iteration 37, loss = 0.35628839
Iteration 38, loss = 0.34061733
Iteration 39, loss = 0.32573772
Iteration 40, loss = 0.31150609
Iteration 41, loss = 0.29784489
Iteration 42, loss = 0.28478906
Iteration 43, loss = 0.27234893
Iteration 44, loss = 0.26063386
Iteration 45, loss = 0.24926753
Iteration 46, loss = 0.23844692
Iteration 47, loss = 0.22821002
Iteration 48, loss = 0.21849652
Iteration 49, loss = 0.20920570
Iteration 50, loss = 0.20041671
Iteration 51, loss = 0.19207293
Iteration 52, loss = 0.18411496
```

```
Iteration 53, loss = 0.17649343
Iteration 54, loss = 0.16935217
Iteration 55, loss = 0.16253898
Iteration 56, loss = 0.15600067
Iteration 57, loss = 0.14993633
Iteration 58, loss = 0.14408303
Iteration 59, loss = 0.13850278
Iteration 60, loss = 0.13327382
Iteration 61, loss = 0.12825323
Iteration 62, loss = 0.12347829
Iteration 63, loss = 0.11897411
Iteration 64, loss = 0.11465582
Iteration 65, loss = 0.11055780
Iteration 66, loss = 0.10667346
Iteration 67, loss = 0.10295384
Iteration 68, loss = 0.09941123
Iteration 69, loss = 0.09604252
Iteration 70, loss = 0.09282170
Iteration 71, loss = 0.08973166
Iteration 72, loss = 0.08682088
Iteration 73, loss = 0.08402251
Iteration 74, loss = 0.08136513
Iteration 75, loss = 0.07879966
Iteration 76, loss = 0.07636968
Iteration 77, loss = 0.07406393
Iteration 78, loss = 0.07180598
Iteration 79, loss = 0.06968243
Iteration 80, loss = 0.06765219
Iteration 81, loss = 0.06568874
Iteration 82, loss = 0.06382529
Iteration 83, loss = 0.06202997
Iteration 84, loss = 0.06030300
Iteration 85, loss = 0.05864779
Iteration 86, loss = 0.05704660
Iteration 87, loss = 0.05554939
Iteration 88, loss = 0.05408148
Iteration 89, loss = 0.05265829
Iteration 90, loss = 0.05130775
Iteration 91, loss = 0.04999907
Iteration 92, loss = 0.04874816
Iteration 93, loss = 0.04754042
Iteration 94, loss = 0.04637368
Iteration 95, loss = 0.04525048
Iteration 96, loss = 0.04418154
Iteration 97, loss = 0.04313189
Iteration 98, loss = 0.04212131
Iteration 99, loss = 0.04115248
Iteration 100, loss = 0.04021271
```

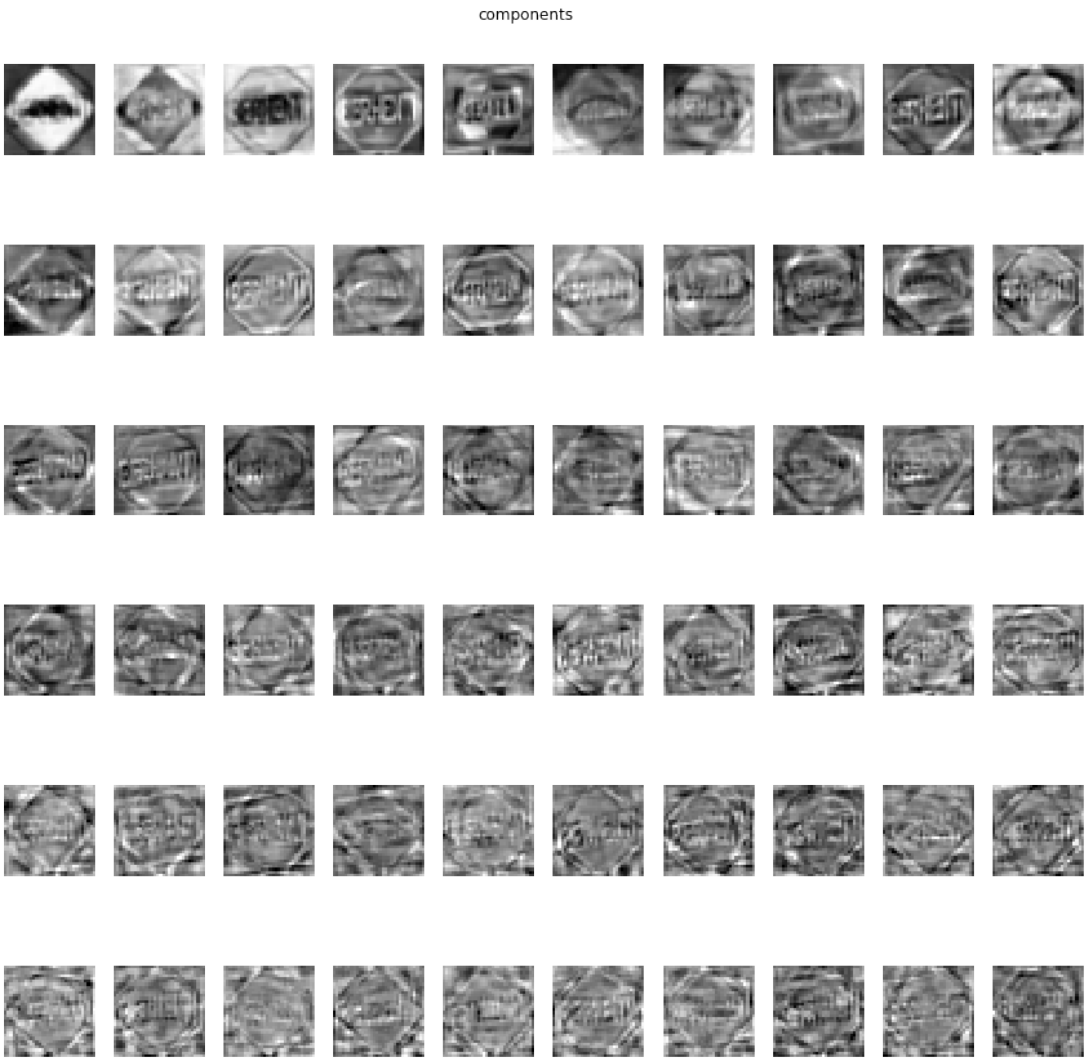
```
Iteration 101, loss = 0.03930541
Iteration 102, loss = 0.03843095
Iteration 103, loss = 0.03757360
Iteration 104, loss = 0.03676299
Iteration 105, loss = 0.03596687
Iteration 106, loss = 0.03519379
Iteration 107, loss = 0.03444526
Iteration 108, loss = 0.03373294
Iteration 109, loss = 0.03302872
Iteration 110, loss = 0.03234695
Iteration 111, loss = 0.03169072
Iteration 112, loss = 0.03104628
Iteration 113, loss = 0.03042899
Iteration 114, loss = 0.02982306
Iteration 115, loss = 0.02923447
Iteration 116, loss = 0.02865752
Iteration 117, loss = 0.02810302
Iteration 118, loss = 0.02754726
Iteration 119, loss = 0.02701860
Iteration 120, loss = 0.02650161
Iteration 121, loss = 0.02598359
Iteration 122, loss = 0.02549007
Iteration 123, loss = 0.02500207
Iteration 124, loss = 0.02453050
Iteration 125, loss = 0.02407026
Iteration 126, loss = 0.02362254
Iteration 127, loss = 0.02318635
Iteration 128, loss = 0.02276412
Iteration 129, loss = 0.02235260
Iteration 130, loss = 0.02195071
Iteration 131, loss = 0.02156284
Iteration 132, loss = 0.02118364
Iteration 133, loss = 0.02081200
Iteration 134, loss = 0.02045566
Iteration 135, loss = 0.02010864
Iteration 136, loss = 0.01976549
Iteration 137, loss = 0.01943189
Iteration 138, loss = 0.01910923
Iteration 139, loss = 0.01879493
Iteration 140, loss = 0.01848948
Iteration 141, loss = 0.01818635
Iteration 142, loss = 0.01789745
Iteration 143, loss = 0.01760969
Iteration 144, loss = 0.01732812
Iteration 145, loss = 0.01706164
Iteration 146, loss = 0.01679479
Iteration 147, loss = 0.01653468
Iteration 148, loss = 0.01627977
```

```
Iteration 149, loss = 0.01603451
Iteration 150, loss = 0.01579175
Iteration 151, loss = 0.01555467
Iteration 152, loss = 0.01532441
Iteration 153, loss = 0.01509556
Iteration 154, loss = 0.01487493
Iteration 155, loss = 0.01465831
Iteration 156, loss = 0.01444463
Iteration 157, loss = 0.01423658
Iteration 158, loss = 0.01403425
Iteration 159, loss = 0.01383425
Iteration 160, loss = 0.01363709
Iteration 161, loss = 0.01344862
Iteration 162, loss = 0.01325998
Iteration 163, loss = 0.01307750
Iteration 164, loss = 0.01289480
Iteration 165, loss = 0.01272015
Iteration 166, loss = 0.01254811
Iteration 167, loss = 0.01237910
Iteration 168, loss = 0.01221265
Iteration 169, loss = 0.01204728
Iteration 170, loss = 0.01188895
Iteration 171, loss = 0.01173393
Iteration 172, loss = 0.01157974
Iteration 173, loss = 0.01142822
Iteration 174, loss = 0.01128186
Iteration 175, loss = 0.01113644
Iteration 176, loss = 0.01099563
Iteration 177, loss = 0.01085604
Iteration 178, loss = 0.01071897
Iteration 179, loss = 0.01058593
Iteration 180, loss = 0.01045334
Iteration 181, loss = 0.01032368
Iteration 182, loss = 0.01019885
Iteration 183, loss = 0.01007297
Iteration 184, loss = 0.00995115
Iteration 185, loss = 0.00983052
Iteration 186, loss = 0.00971107
Iteration 187, loss = 0.00959616
Iteration 188, loss = 0.00948144
Iteration 189, loss = 0.00936946
Iteration 190, loss = 0.00925778
Iteration 191, loss = 0.00915072
Iteration 192, loss = 0.00904232
Iteration 193, loss = 0.00893961
Iteration 194, loss = 0.00883462
Iteration 195, loss = 0.00873359
Iteration 196, loss = 0.00863269
```

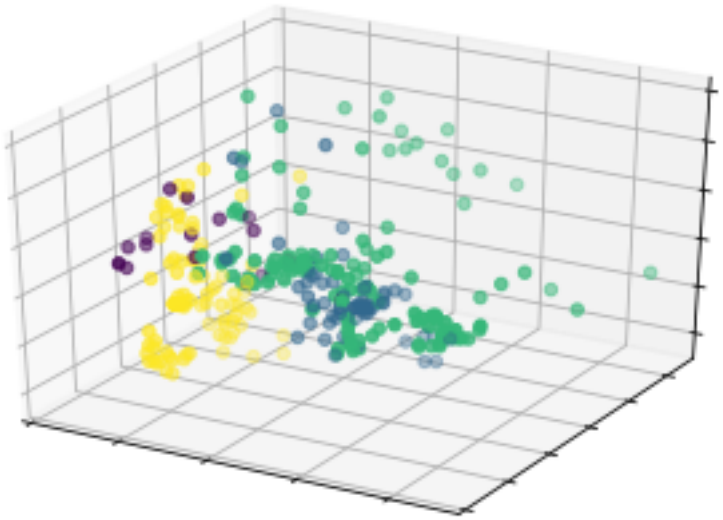
```
Iteration 197, loss = 0.00853566
Iteration 198, loss = 0.00843847
Iteration 199, loss = 0.00834390
Iteration 200, loss = 0.00824972


C:\Users\GaryNg\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:56
  % self.max_iter, ConvergenceWarning)


MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.965
confusion matrix:
 [[10  0  1  0]
 [ 0 31  0  0]
 [ 0  2 97  2]
 [ 1  0  1 55]]
              precision    recall  f1-score   support

           0       0.91      0.91      0.91        11
           1       0.94      1.00      0.97        31
           2       0.98      0.96      0.97       101
           3       0.96      0.96      0.96        57

   micro avg       0.96      0.96      0.96       200
   macro avg       0.95      0.96      0.95       200
weighted avg       0.97      0.96      0.97       200

--- Model 10 ---
trainingData: (300, 1024), trainingLabels: (300,)
testingData: (200, 1024), testingLabels: (200,)
total explained variance: 0.9494212287556807
```

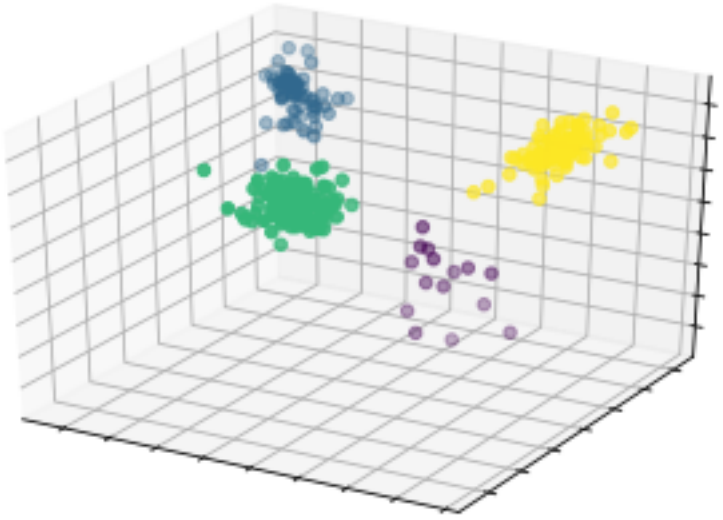



```
pca components: (64, 1024)
```




```
trainingData: (300, 64), trainingLabels: (300,)
testingData: (200, 64), testingLabels: (200,)
```

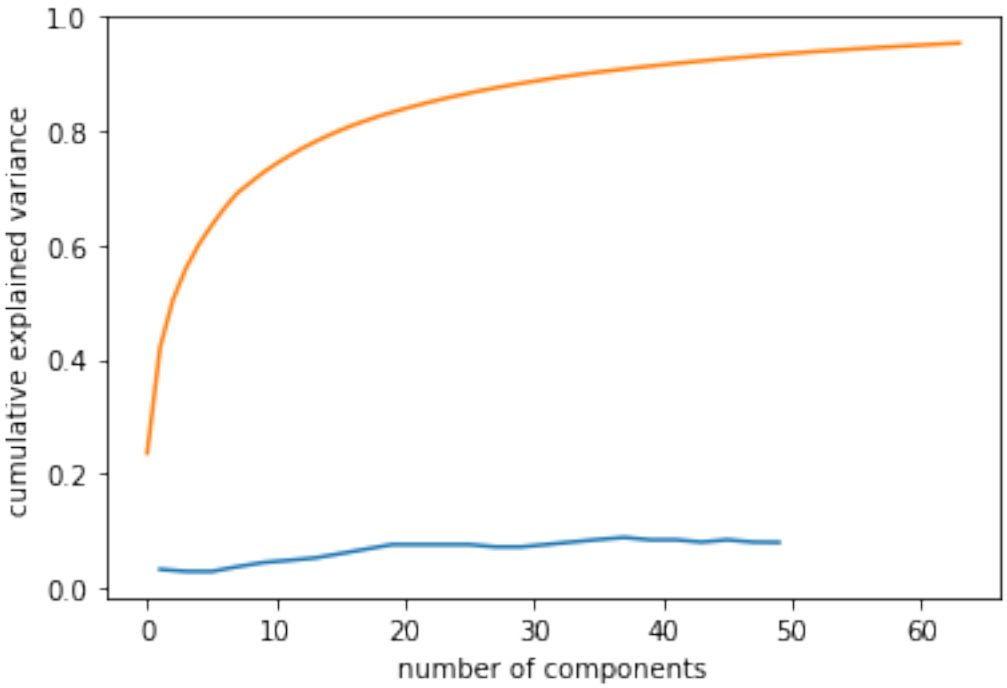




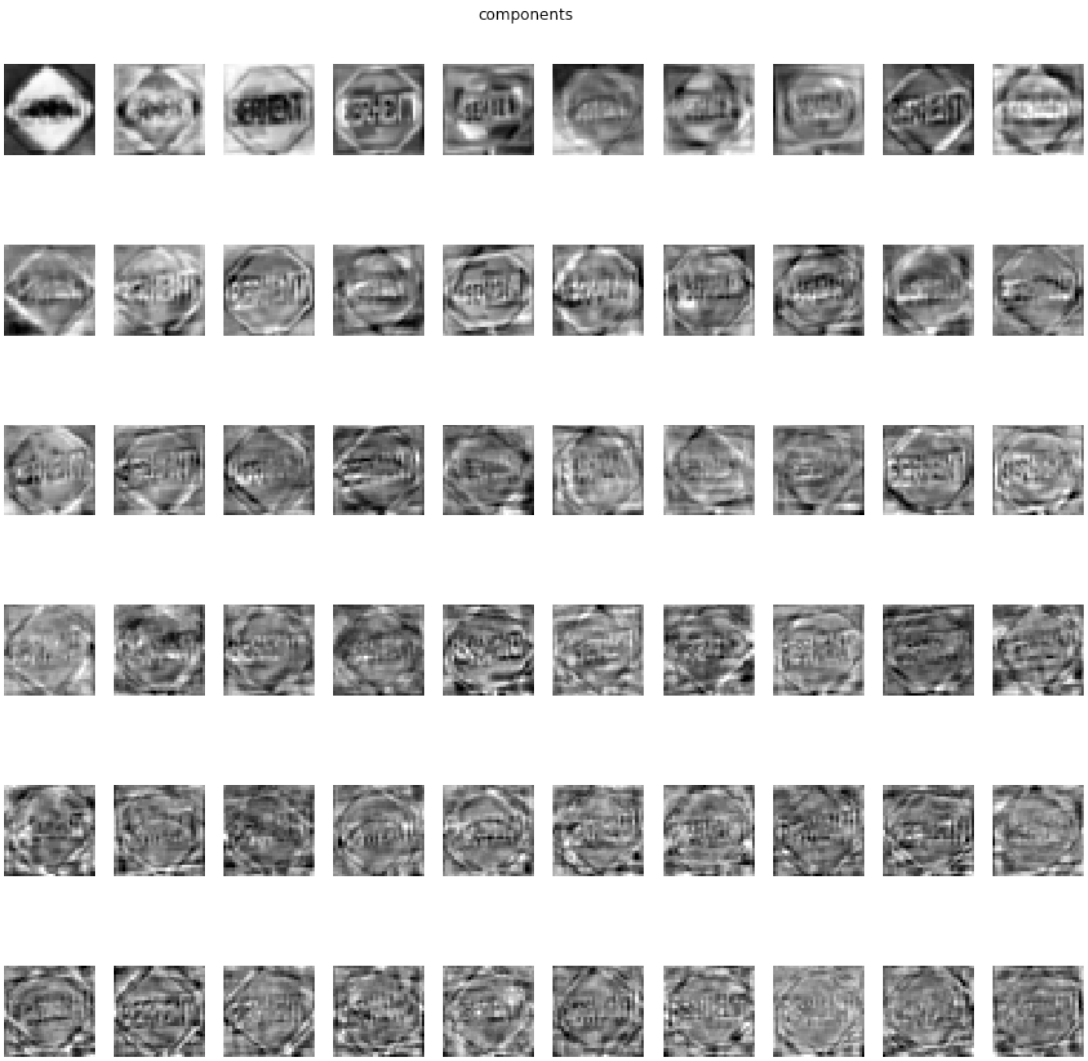
```
trainingData: (300, 3), trainingLabels: (300,)
testingData: (200, 3), testingLabels: (200,)
Iteration 1, loss = 1.26551276
Iteration 2, loss = 0.62153602
Iteration 3, loss = 0.27647742
Iteration 4, loss = 0.11072530
```

```
Iteration 5, loss = 0.04335855
Iteration 6, loss = 0.01800272
Iteration 7, loss = 0.00826062
Iteration 8, loss = 0.00449910
Iteration 9, loss = 0.00267129
Iteration 10, loss = 0.00181871
Iteration 11, loss = 0.00129914
Iteration 12, loss = 0.00103217
Iteration 13, loss = 0.00086534
Iteration 14, loss = 0.00075975
Iteration 15, loss = 0.00068434
Iteration 16, loss = 0.00063493
Iteration 17, loss = 0.00059857
Iteration 18, loss = 0.00056931
Iteration 19, loss = 0.00054769
Iteration 20, loss = 0.00053149
Iteration 21, loss = 0.00051658
Iteration 22, loss = 0.00050645
Iteration 23, loss = 0.00049611
Iteration 24, loss = 0.00048956
Iteration 25, loss = 0.00048290
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.99
confusion matrix:
 [[ 10   0   0   1]
 [  0  31   0   0]
 [  0   1 100   0]
 [  0   0   0  57]]
              precision    recall  f1-score   support

           0       1.00      0.91      0.95        11
           1       0.97      1.00      0.98        31
           2       1.00      0.99      1.00       101
           3       0.98      1.00      0.99        57

   micro avg       0.99      0.99      0.99       200
   macro avg       0.99      0.97      0.98       200
weighted avg       0.99      0.99      0.99       200
```

```
In [47]: runAll(testSize=0.5)

--- Model 1 ---
trainingData: (250, 1024), trainingLabels: (250,)
testingData: (250, 1024), testingLabels: (250,)
optimal k: 5
training accuracy: 0.98
testing accuracy: 0.98
confusion matrix:
 [[ 13   0   0   1]
 [  2  39   0   1]
 [  0   0 117   1]
 [  0   0   0  76]]
              precision    recall  f1-score   support

           0       0.87      0.93      0.90        14
           1       1.00      0.93      0.96        42
           2       1.00      0.99      1.00       118
           3       0.96      1.00      0.98        76

   micro avg       0.98      0.98      0.98       250
   macro avg       0.96      0.96      0.96       250
weighted avg       0.98      0.98      0.98       250

--- Model 2 ---
trainingData: (250, 1024), trainingLabels: (250,)
testingData: (250, 1024), testingLabels: (250,)
total explained variance: 0.9535341962901396
```

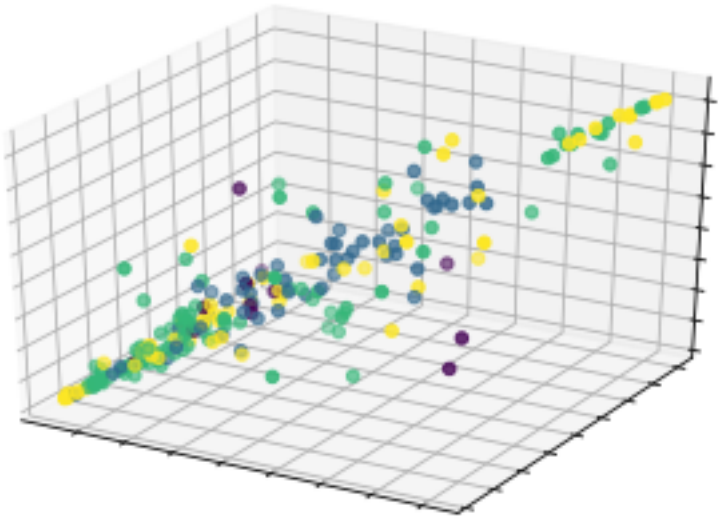


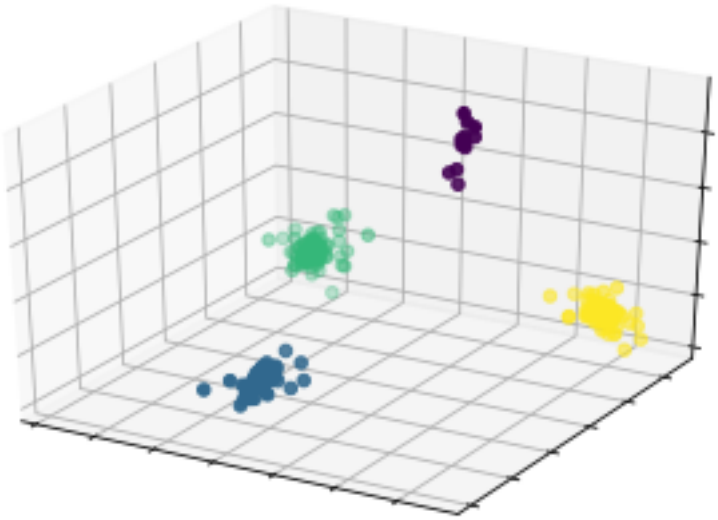

```
pca components: (64, 1024)
```



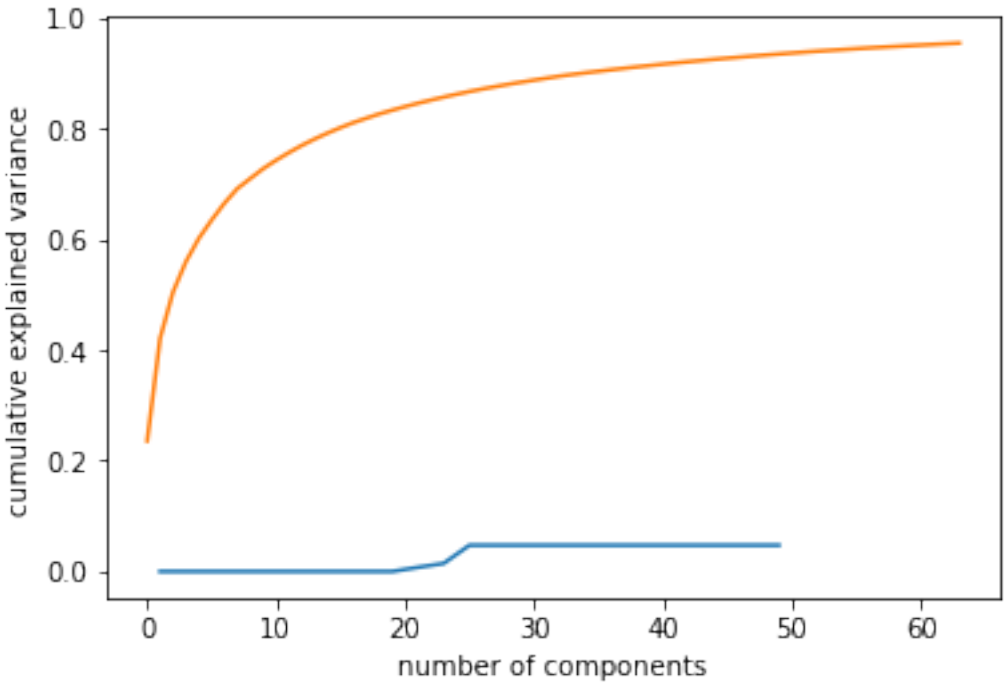

```
trainingData: (250, 64), trainingLabels: (250,)
testingData: (250, 64), testingLabels: (250,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.972
confusion matrix:
 [[ 12   0   2   0]
 [  0  37   4   1]
 [  0   0 118   0]
 [  0   0   0  76]]
              precision    recall  f1-score   support

           0       1.00      0.86      0.92        14
           1       1.00      0.88      0.94        42
```

```
           2       0.95      1.00      0.98       118
           3       0.99      1.00      0.99        76

   micro avg       0.97      0.97      0.97       250
   macro avg       0.98      0.93      0.96       250
weighted avg       0.97      0.97      0.97       250

--- Model 3 ---
trainingData: (250, 1024), trainingLabels: (250,)
testingData: (250, 1024), testingLabels: (250,)


C:\Users\GaryNg\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning:
  warnings.warn("Variables are collinear.")
C:\Users\GaryNg\Anaconda3\lib\site-packages\matplotlib\figure.py:98: MatplotlibDeprecationWarni
Adding an axes using the same arguments as a previous axes currently reuses the earlier instanc
  "Adding an axes using the same arguments as a previous axes "
```

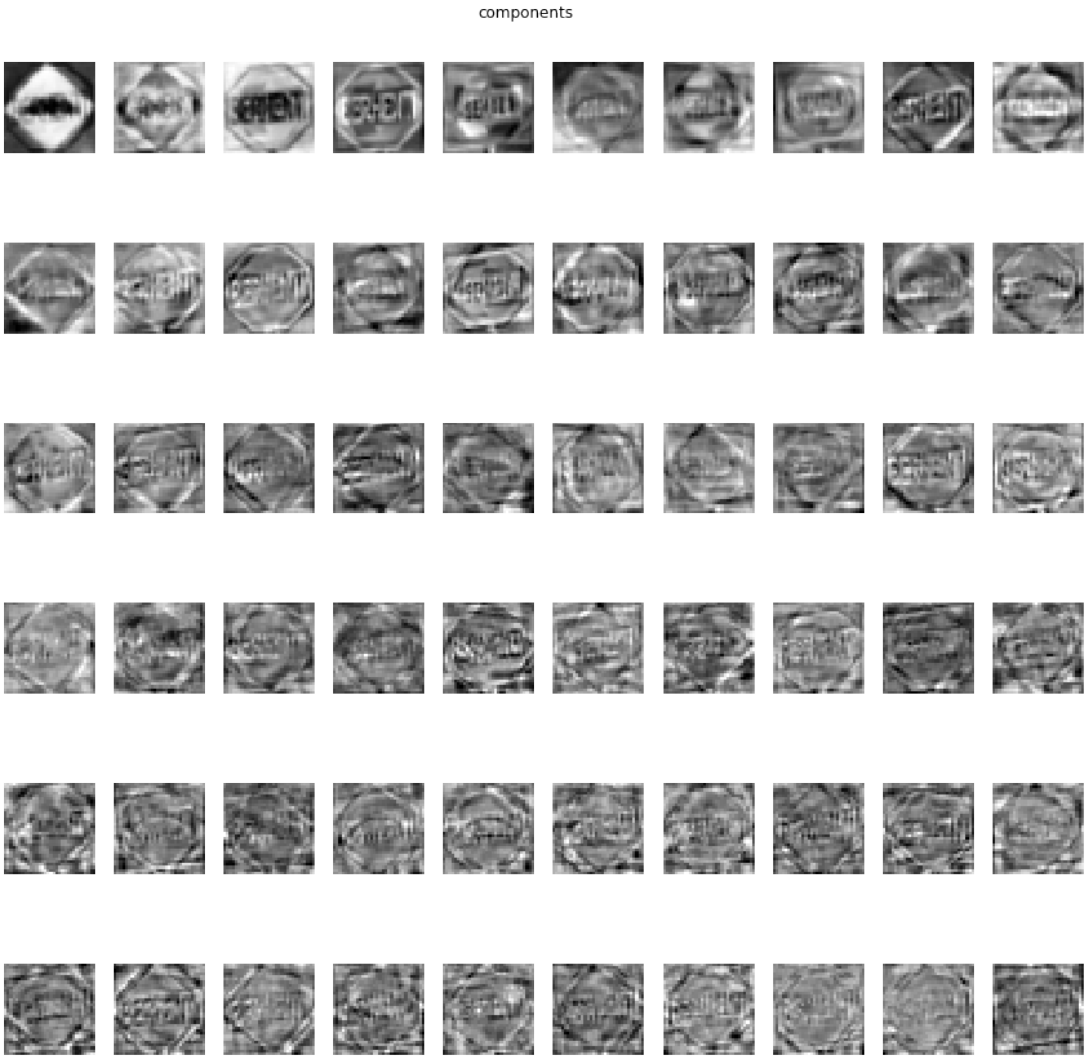




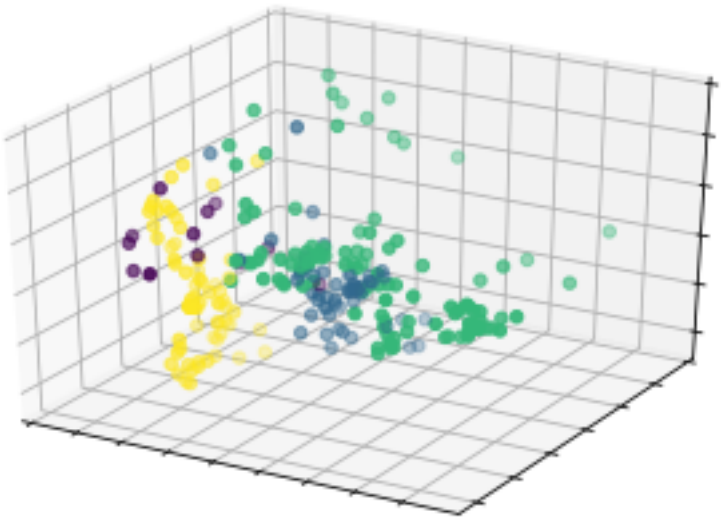
```
trainingData: (250, 3), trainingLabels: (250,)
testingData: (250, 3), testingLabels: (250,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.988
confusion matrix:
 [[ 13   0   1   0]
 [  1  41   0   0]
 [  0   1 117   0]
 [  0   0   0  76]]
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        14
           1       0.98      0.98      0.98        42
           2       0.99      0.99      0.99       118
           3       1.00      1.00      1.00        76

   micro avg       0.99      0.99      0.99       250
   macro avg       0.97      0.97      0.97       250
weighted avg       0.99      0.99      0.99       250

--- Model 4 ---
trainingData: (250, 1024), trainingLabels: (250,)
testingData: (250, 1024), testingLabels: (250,)
total explained variance: 0.9536011681512757
```

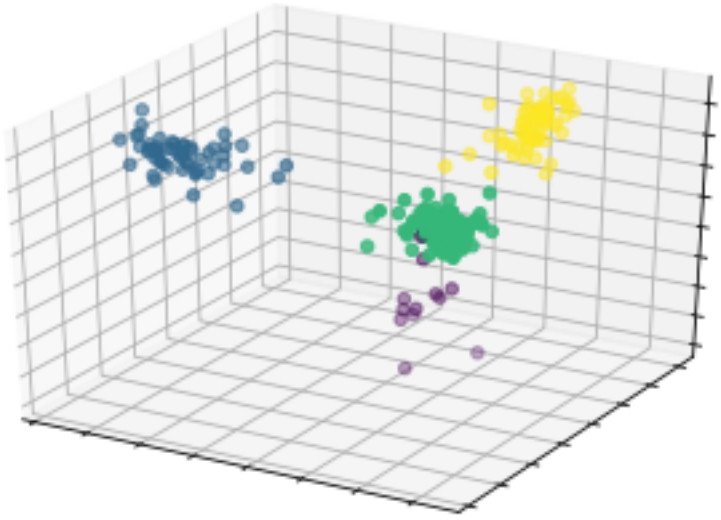



```
pca components: (64, 1024)
```



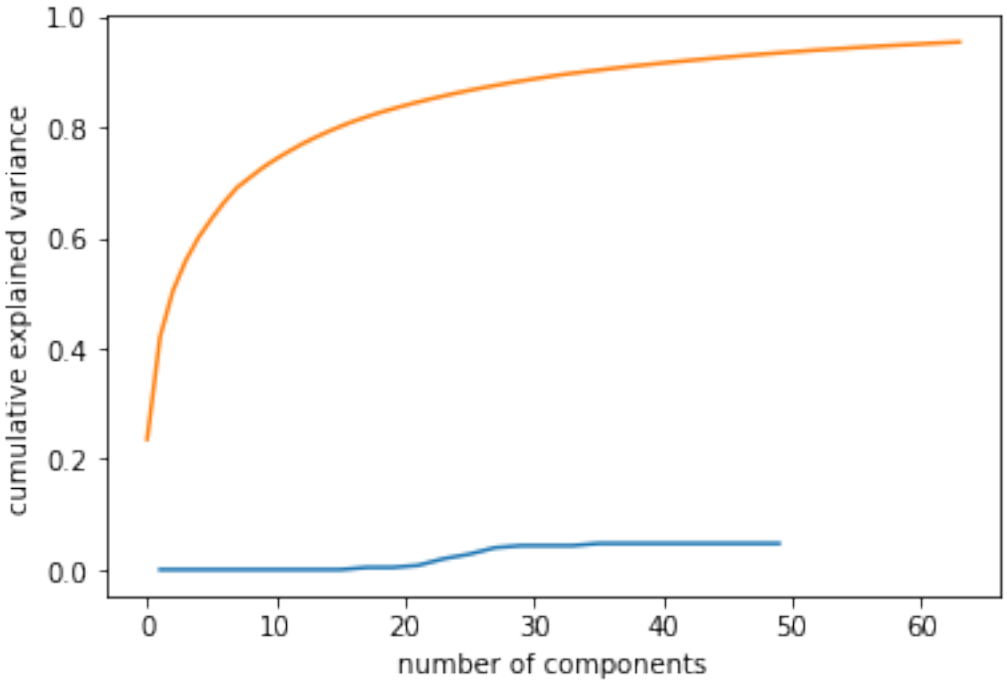

```
trainingData: (250, 64), trainingLabels: (250,)
testingData: (250, 64), testingLabels: (250,)
```



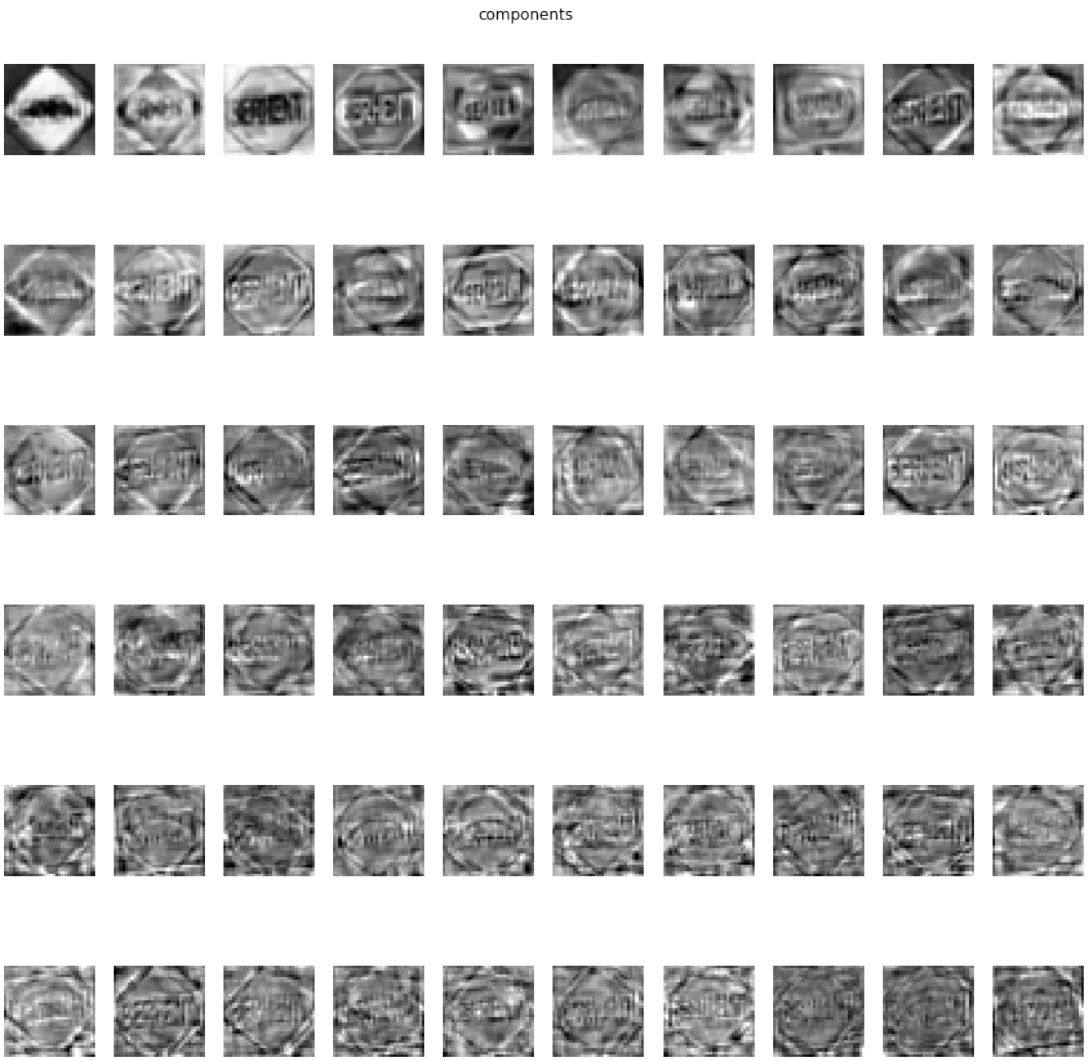


```
trainingData: (250, 3), trainingLabels: (250,)
testingData: (250, 3), testingLabels: (250,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.992
confusion matrix:
```

```
 [[ 13   0   0   1]
 [  0  42   0   0]
 [  0   1 117   0]
 [  0   0   0  76]]
              precision    recall  f1-score   support

           0       1.00      0.93      0.96        14
           1       0.98      1.00      0.99        42
           2       1.00      0.99      1.00       118
           3       0.99      1.00      0.99        76

   micro avg       0.99      0.99      0.99       250
   macro avg       0.99      0.98      0.99       250
weighted avg       0.99      0.99      0.99       250

--- Model 5 ---
trainingData: (250, 1024), trainingLabels: (250,)
testingData: (250, 1024), testingLabels: (250,)
total explained variance: 0.953504329704072
```

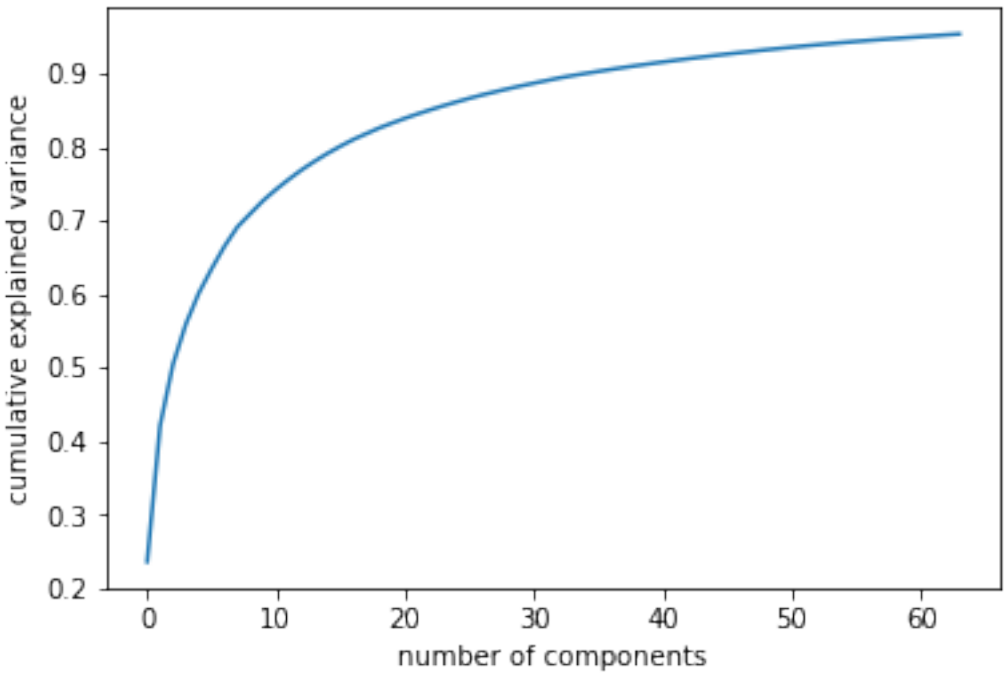



```
pca components: (64, 1024)
```

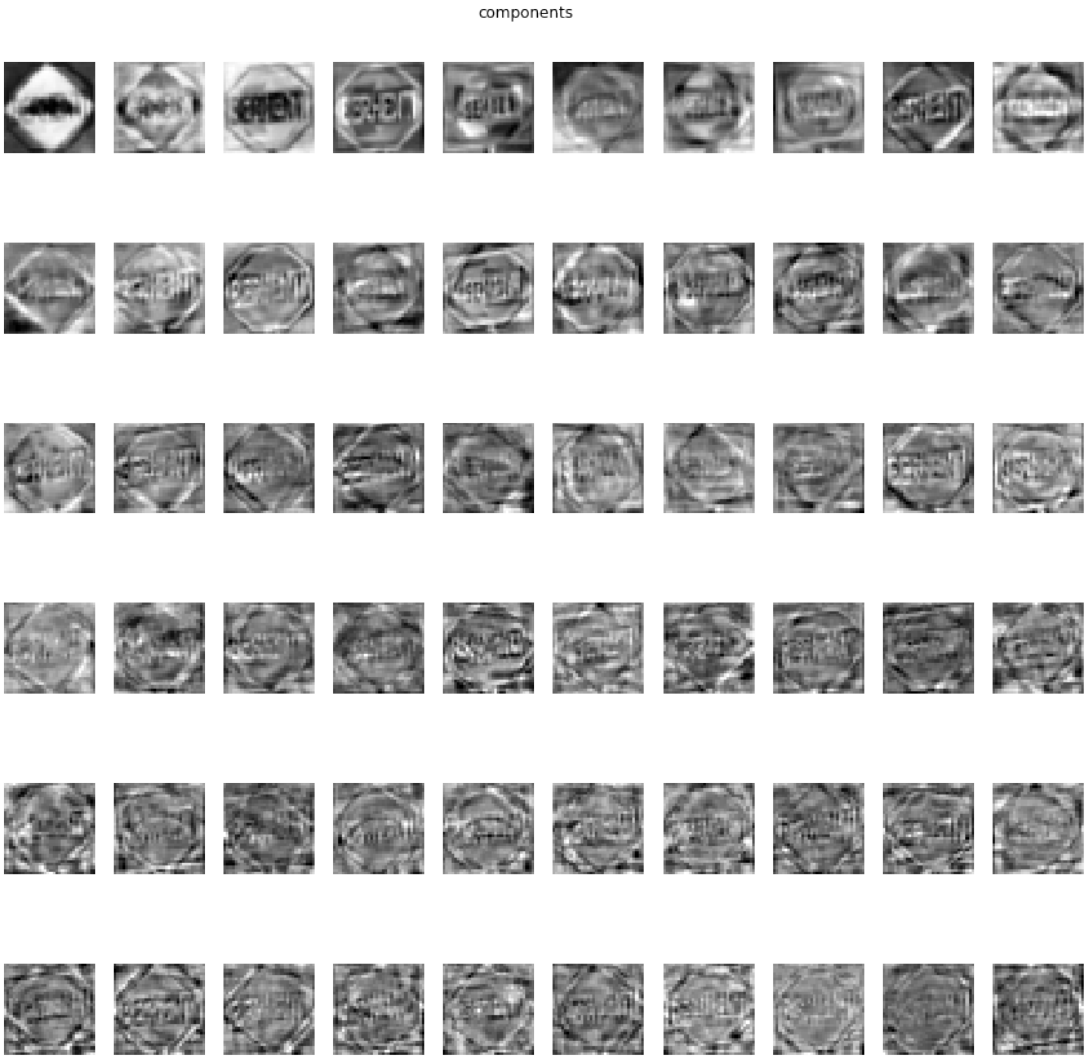



```
trainingData: (250, 64), trainingLabels: (250,)
testingData: (250, 64), testingLabels: (250,)
training accuracy: 1.0
testing accuracy: 0.988
confusion matrix:
 [[ 13   0   0   1]
 [  0  41   0   1]
 [  0   1 117   0]
 [  0   0   0  76]]
              precision    recall  f1-score   support

           0       1.00      0.93      0.96        14
           1       0.98      0.98      0.98        42
           2       1.00      0.99      1.00       118
```

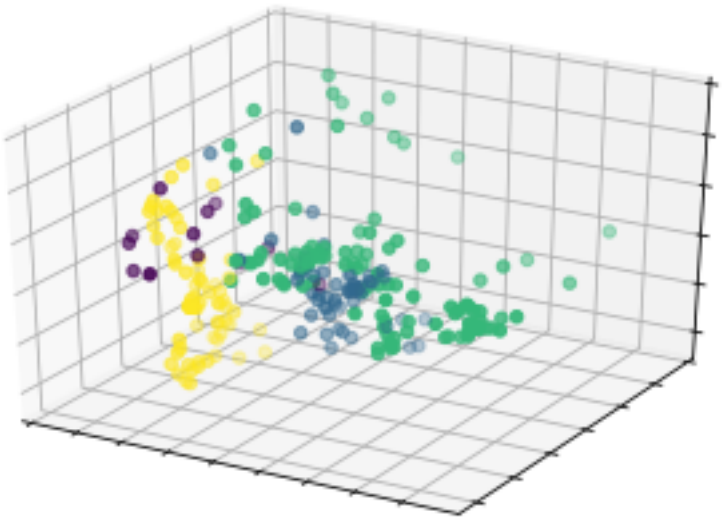
```
           3       0.97      1.00      0.99        76

   micro avg       0.99      0.99      0.99       250
   macro avg       0.99      0.97      0.98       250
weighted avg       0.99      0.99      0.99       250

--- Model 6 ---
trainingData: (250, 1024), trainingLabels: (250,)
testingData: (250, 1024), testingLabels: (250,)
total explained variance: 0.9535996155525811
```

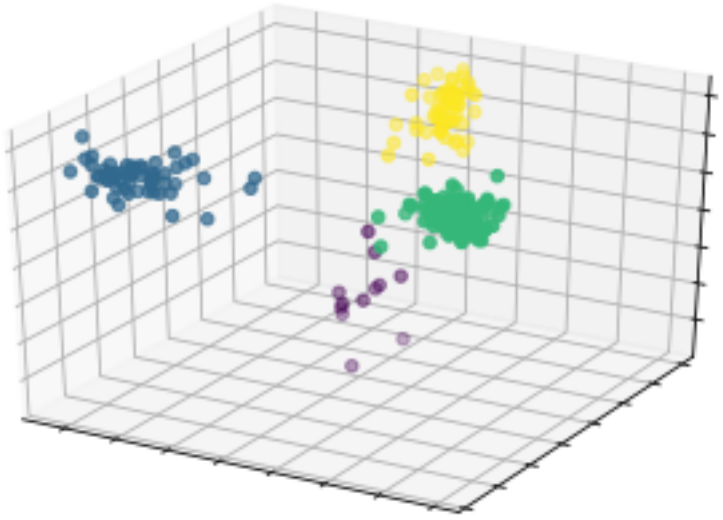



```
pca components: (64, 1024)
```




```
trainingData: (250, 64), trainingLabels: (250,)
testingData: (250, 64), testingLabels: (250,)
```

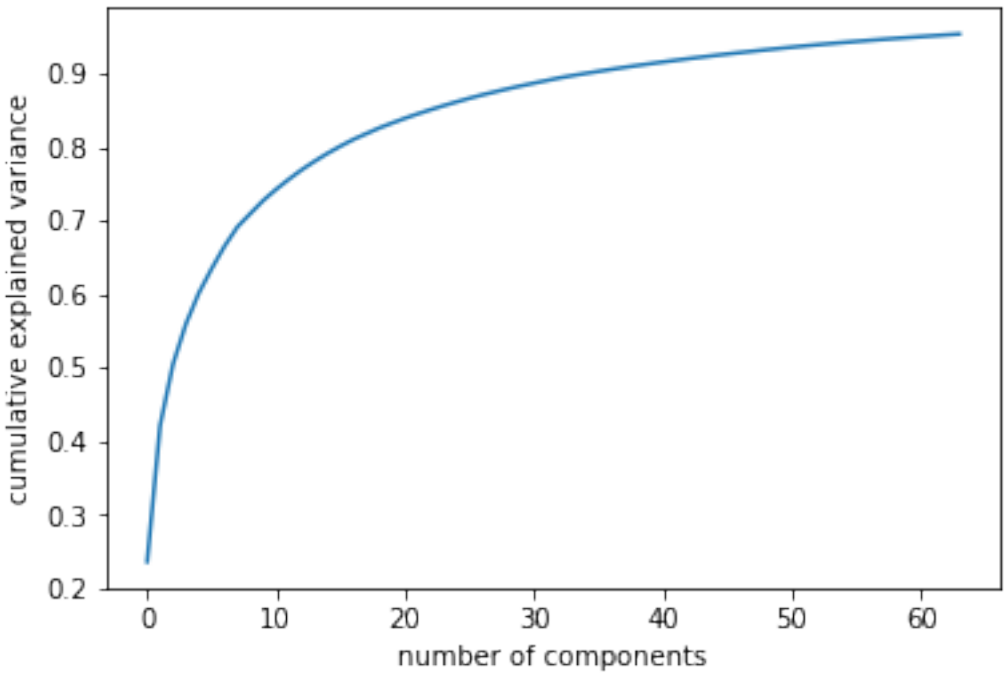




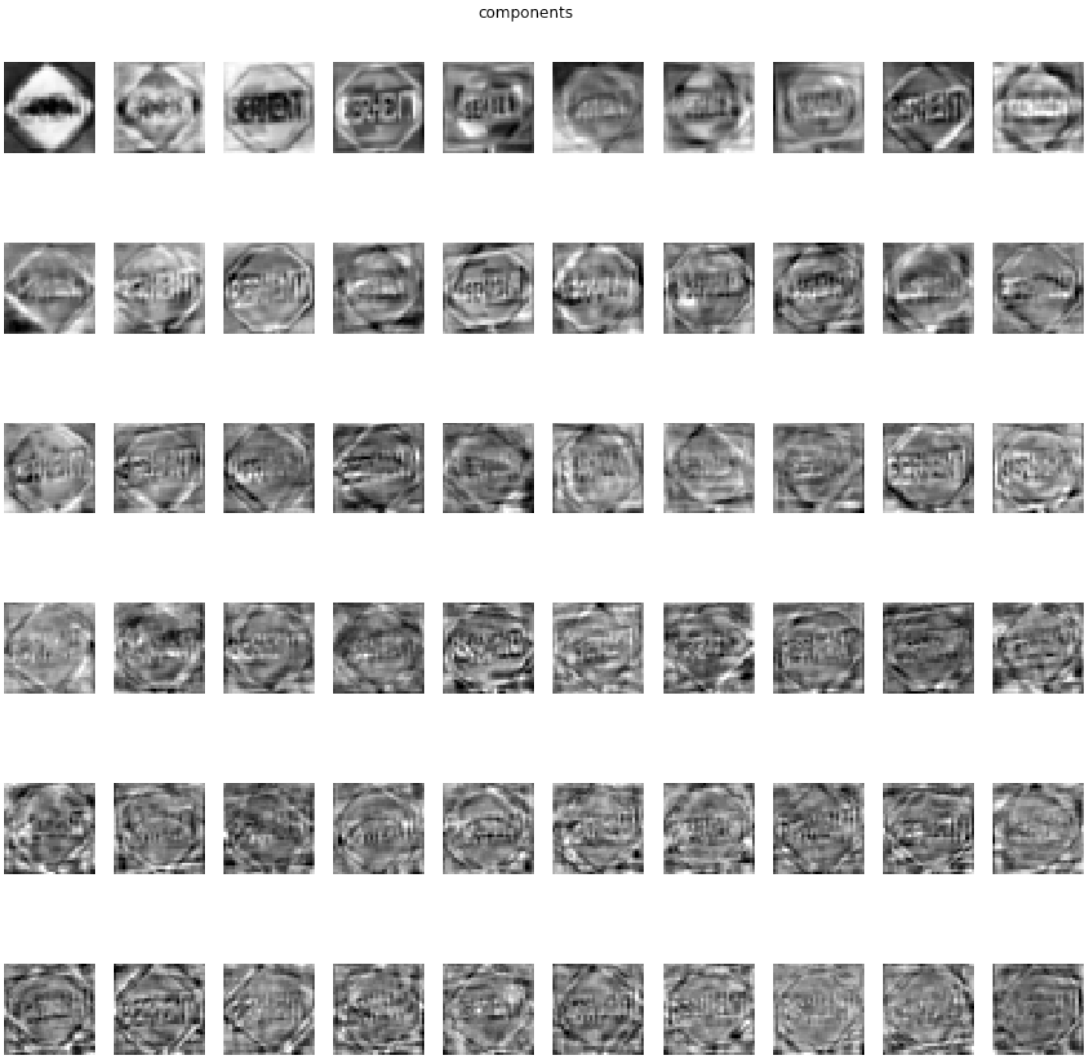
```
trainingData: (250, 3), trainingLabels: (250,)
testingData: (250, 3), testingLabels: (250,)
training accuracy: 1.0
testing accuracy: 0.996
confusion matrix:
 [[ 14   0   0   0]
```

```
 [  0  42   0   0]
 [  0   1 117   0]
 [  0   0   0  76]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       0.98      1.00      0.99        42
           2       1.00      0.99      1.00       118
           3       1.00      1.00      1.00        76

   micro avg       1.00      1.00      1.00       250
   macro avg       0.99      1.00      1.00       250
weighted avg       1.00      1.00      1.00       250

--- Model 7 ---
trainingData: (250, 1024), trainingLabels: (250,)
testingData: (250, 1024), testingLabels: (250,)
Iteration 1, loss = 1.17687057
Iteration 2, loss = 0.05443517
Iteration 3, loss = 0.05426800
Iteration 4, loss = 0.00742041
Iteration 5, loss = 0.09029260
Iteration 6, loss = 0.00448998
Iteration 7, loss = 0.00119625
Iteration 8, loss = 0.01186595
Iteration 9, loss = 0.00933818
Iteration 10, loss = 0.00105311
Iteration 11, loss = 0.00094195
Iteration 12, loss = 0.00093022
Iteration 13, loss = 0.00092288
Iteration 14, loss = 0.00091678
Iteration 15, loss = 0.00091412
Iteration 16, loss = 0.00091488
Iteration 17, loss = 0.00092431
Iteration 18, loss = 0.00092125
Iteration 19, loss = 0.00091716
Iteration 20, loss = 0.00091667
Iteration 21, loss = 0.00091497
Iteration 22, loss = 0.00091190
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
```

```
testing accuracy: 0.988
confusion matrix:
 [[ 13   0   0   1]
 [  0  40   1   1]
 [  0   0 118   0]
 [  0   0   0  76]]
              precision    recall  f1-score   support

           0       1.00      0.93      0.96        14
           1       1.00      0.95      0.98        42
           2       0.99      1.00      1.00       118
           3       0.97      1.00      0.99        76

   micro avg       0.99      0.99      0.99       250
   macro avg       0.99      0.97      0.98       250
weighted avg       0.99      0.99      0.99       250

--- Model 8 ---
trainingData: (250, 1024), trainingLabels: (250,)
testingData: (250, 1024), testingLabels: (250,)
total explained variance: 0.953568910457226
```




```
pca components: (64, 1024)
```



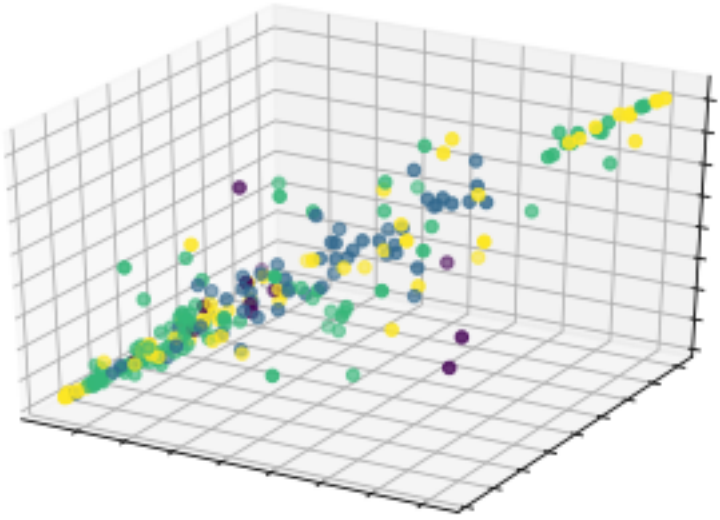

```
trainingData: (250, 64), trainingLabels: (250,)
testingData: (250, 64), testingLabels: (250,)
Iteration 1, loss = 1.62300166
Iteration 2, loss = 1.50113389
Iteration 3, loss = 1.40830556
Iteration 4, loss = 1.32130305
Iteration 5, loss = 1.25165484
Iteration 6, loss = 1.18897671
Iteration 7, loss = 1.13331788
Iteration 8, loss = 1.08291283
Iteration 9, loss = 1.03669488
Iteration 10, loss = 0.99407782
Iteration 11, loss = 0.95304589
Iteration 12, loss = 0.91271376
```

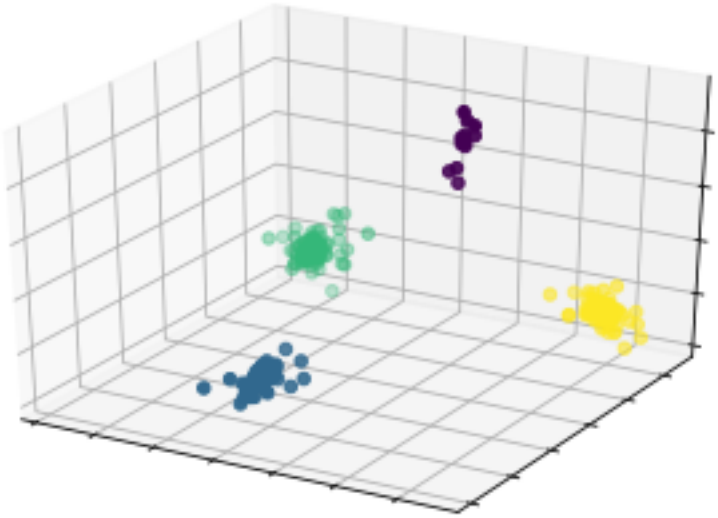
```
Iteration 13, loss = 0.87450859
Iteration 14, loss = 0.83638756
Iteration 15, loss = 0.79972438
Iteration 16, loss = 0.76317359
Iteration 17, loss = 0.72729396
Iteration 18, loss = 0.69212358
Iteration 19, loss = 0.65734496
Iteration 20, loss = 0.62278764
Iteration 21, loss = 0.58884829
Iteration 22, loss = 0.55576067
Iteration 23, loss = 0.52308603
Iteration 24, loss = 0.49062221
Iteration 25, loss = 0.45989218
Iteration 26, loss = 0.42961633
Iteration 27, loss = 0.40059007
Iteration 28, loss = 0.37288681
Iteration 29, loss = 0.34617697
Iteration 30, loss = 0.32125482
Iteration 31, loss = 0.29728462
Iteration 32, loss = 0.27455402
Iteration 33, loss = 0.25337657
Iteration 34, loss = 0.23353856
Iteration 35, loss = 0.21413884
Iteration 36, loss = 0.19670382
Iteration 37, loss = 0.18044716
Iteration 38, loss = 0.16531052
Iteration 39, loss = 0.15133897
Iteration 40, loss = 0.13867752
Iteration 41, loss = 0.12677360
Iteration 42, loss = 0.11612882
Iteration 43, loss = 0.10630089
Iteration 44, loss = 0.09709775
Iteration 45, loss = 0.08888168
Iteration 46, loss = 0.08186288
Iteration 47, loss = 0.07519114
Iteration 48, loss = 0.06936592
Iteration 49, loss = 0.06414815
Iteration 50, loss = 0.05926539
Iteration 51, loss = 0.05503868
Iteration 52, loss = 0.05112226
Iteration 53, loss = 0.04758724
Iteration 54, loss = 0.04431731
Iteration 55, loss = 0.04141420
Iteration 56, loss = 0.03877562
Iteration 57, loss = 0.03631399
Iteration 58, loss = 0.03412598
Iteration 59, loss = 0.03209824
Iteration 60, loss = 0.03025811
```

```
Iteration 61, loss = 0.02860203
Iteration 62, loss = 0.02705748
Iteration 63, loss = 0.02564593
Iteration 64, loss = 0.02432900
Iteration 65, loss = 0.02313231
Iteration 66, loss = 0.02204677
Iteration 67, loss = 0.02099685
Iteration 68, loss = 0.02005934
Iteration 69, loss = 0.01916230
Iteration 70, loss = 0.01832484
Iteration 71, loss = 0.01754113
Iteration 72, loss = 0.01681241
Iteration 73, loss = 0.01614426
Iteration 74, loss = 0.01549705
Iteration 75, loss = 0.01491009
Iteration 76, loss = 0.01434710
Iteration 77, loss = 0.01383216
Iteration 78, loss = 0.01333455
Iteration 79, loss = 0.01285569
Iteration 80, loss = 0.01242049
Iteration 81, loss = 0.01199706
Iteration 82, loss = 0.01159890
Iteration 83, loss = 0.01121528
Iteration 84, loss = 0.01085453
Iteration 85, loss = 0.01051639
Iteration 86, loss = 0.01018469
Iteration 87, loss = 0.00987918
Iteration 88, loss = 0.00958949
Iteration 89, loss = 0.00931151
Iteration 90, loss = 0.00904662
Iteration 91, loss = 0.00879943
Iteration 92, loss = 0.00855524
Iteration 93, loss = 0.00832754
Iteration 94, loss = 0.00810277
Iteration 95, loss = 0.00789406
Iteration 96, loss = 0.00768923
Iteration 97, loss = 0.00749576
Iteration 98, loss = 0.00731211
Iteration 99, loss = 0.00713008
Iteration 100, loss = 0.00696190
Iteration 101, loss = 0.00679356
Iteration 102, loss = 0.00663701
Iteration 103, loss = 0.00648321
Iteration 104, loss = 0.00633874
Iteration 105, loss = 0.00619257
Iteration 106, loss = 0.00606006
Iteration 107, loss = 0.00592396
Iteration 108, loss = 0.00579702
```

```
Iteration 109, loss = 0.00567343
Iteration 110, loss = 0.00555693
Iteration 111, loss = 0.00544175
Iteration 112, loss = 0.00533134
Iteration 113, loss = 0.00522429
Iteration 114, loss = 0.00512257
Iteration 115, loss = 0.00502136
Iteration 116, loss = 0.00492641
Iteration 117, loss = 0.00483170
Iteration 118, loss = 0.00473994
Iteration 119, loss = 0.00465302
Iteration 120, loss = 0.00456730
Iteration 121, loss = 0.00448525
Iteration 122, loss = 0.00440355
Iteration 123, loss = 0.00432591
Iteration 124, loss = 0.00425083
Iteration 125, loss = 0.00417789
Iteration 126, loss = 0.00410544
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(64, 64), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.984
confusion matrix:
 [[ 13   1   0   0]
 [  0  41   0   1]
 [  0   1 117   0]
 [  0   1   0  75]]
              precision    recall  f1-score   support

           0       1.00      0.93      0.96        14
           1       0.93      0.98      0.95        42
           2       1.00      0.99      1.00       118
           3       0.99      0.99      0.99        76

   micro avg       0.98      0.98      0.98       250
   macro avg       0.98      0.97      0.97       250
weighted avg       0.98      0.98      0.98       250

--- Model 9 ---
trainingData: (250, 1024), trainingLabels: (250,)
testingData: (250, 1024), testingLabels: (250,)
```

```
C:\Users\GaryNg\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning:
  warnings.warn("Variables are collinear.")
```





```
trainingData: (250, 3), trainingLabels: (250,)
testingData: (250, 3), testingLabels: (250,)
```

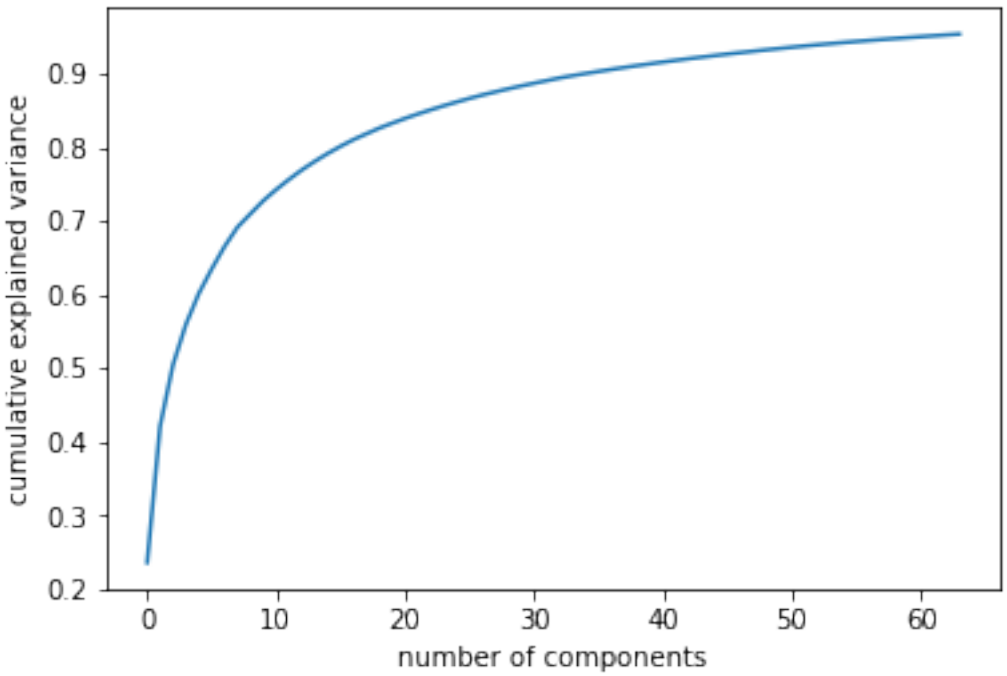
```
Iteration 1, loss = 1.42289366
Iteration 2, loss = 1.37808204
Iteration 3, loss = 1.33569471
Iteration 4, loss = 1.29412682
Iteration 5, loss = 1.25327475
Iteration 6, loss = 1.21328076
Iteration 7, loss = 1.17409536
Iteration 8, loss = 1.13580425
Iteration 9, loss = 1.09850192
Iteration 10, loss = 1.06183443
Iteration 11, loss = 1.02625545
Iteration 12, loss = 0.99141801
Iteration 13, loss = 0.95733897
Iteration 14, loss = 0.92445938
Iteration 15, loss = 0.89225701
Iteration 16, loss = 0.86087947
Iteration 17, loss = 0.83013770
Iteration 18, loss = 0.80001067
Iteration 19, loss = 0.77102553
Iteration 20, loss = 0.74252761
Iteration 21, loss = 0.71467120
Iteration 22, loss = 0.68759048
Iteration 23, loss = 0.66122765
Iteration 24, loss = 0.63544796
Iteration 25, loss = 0.61035538
Iteration 26, loss = 0.58595160
Iteration 27, loss = 0.56220376
Iteration 28, loss = 0.53919653
Iteration 29, loss = 0.51678032
Iteration 30, loss = 0.49517967
Iteration 31, loss = 0.47415035
Iteration 32, loss = 0.45394256
Iteration 33, loss = 0.43446150
Iteration 34, loss = 0.41566451
Iteration 35, loss = 0.39759672
Iteration 36, loss = 0.38026826
Iteration 37, loss = 0.36360068
Iteration 38, loss = 0.34763646
Iteration 39, loss = 0.33239333
Iteration 40, loss = 0.31772084
Iteration 41, loss = 0.30372545
Iteration 42, loss = 0.29034056
Iteration 43, loss = 0.27755211
Iteration 44, loss = 0.26535572
Iteration 45, loss = 0.25370497
Iteration 46, loss = 0.24260310
Iteration 47, loss = 0.23201318
Iteration 48, loss = 0.22194869
```

```
Iteration 49, loss = 0.21241196
Iteration 50, loss = 0.20329731
Iteration 51, loss = 0.19464217
Iteration 52, loss = 0.18646425
Iteration 53, loss = 0.17866183
Iteration 54, loss = 0.17131610
Iteration 55, loss = 0.16428839
Iteration 56, loss = 0.15766153
Iteration 57, loss = 0.15137709
Iteration 58, loss = 0.14541776
Iteration 59, loss = 0.13975794
Iteration 60, loss = 0.13440699
Iteration 61, loss = 0.12933591
Iteration 62, loss = 0.12448062
Iteration 63, loss = 0.11992856
Iteration 64, loss = 0.11554670
Iteration 65, loss = 0.11141240
Iteration 66, loss = 0.10747429
Iteration 67, loss = 0.10372450
Iteration 68, loss = 0.10014147
Iteration 69, loss = 0.09674049
Iteration 70, loss = 0.09349896
Iteration 71, loss = 0.09040461
Iteration 72, loss = 0.08745168
Iteration 73, loss = 0.08463924
Iteration 74, loss = 0.08196176
Iteration 75, loss = 0.07938147
Iteration 76, loss = 0.07694856
Iteration 77, loss = 0.07459646
Iteration 78, loss = 0.07235738
Iteration 79, loss = 0.07021955
Iteration 80, loss = 0.06816402
Iteration 81, loss = 0.06620060
Iteration 82, loss = 0.06432188
Iteration 83, loss = 0.06252756
Iteration 84, loss = 0.06078861
Iteration 85, loss = 0.05913537
Iteration 86, loss = 0.05754367
Iteration 87, loss = 0.05601689
Iteration 88, loss = 0.05454684
Iteration 89, loss = 0.05314539
Iteration 90, loss = 0.05178518
Iteration 91, loss = 0.05048985
Iteration 92, loss = 0.04924323
Iteration 93, loss = 0.04803842
Iteration 94, loss = 0.04687971
Iteration 95, loss = 0.04576835
Iteration 96, loss = 0.04469525
```

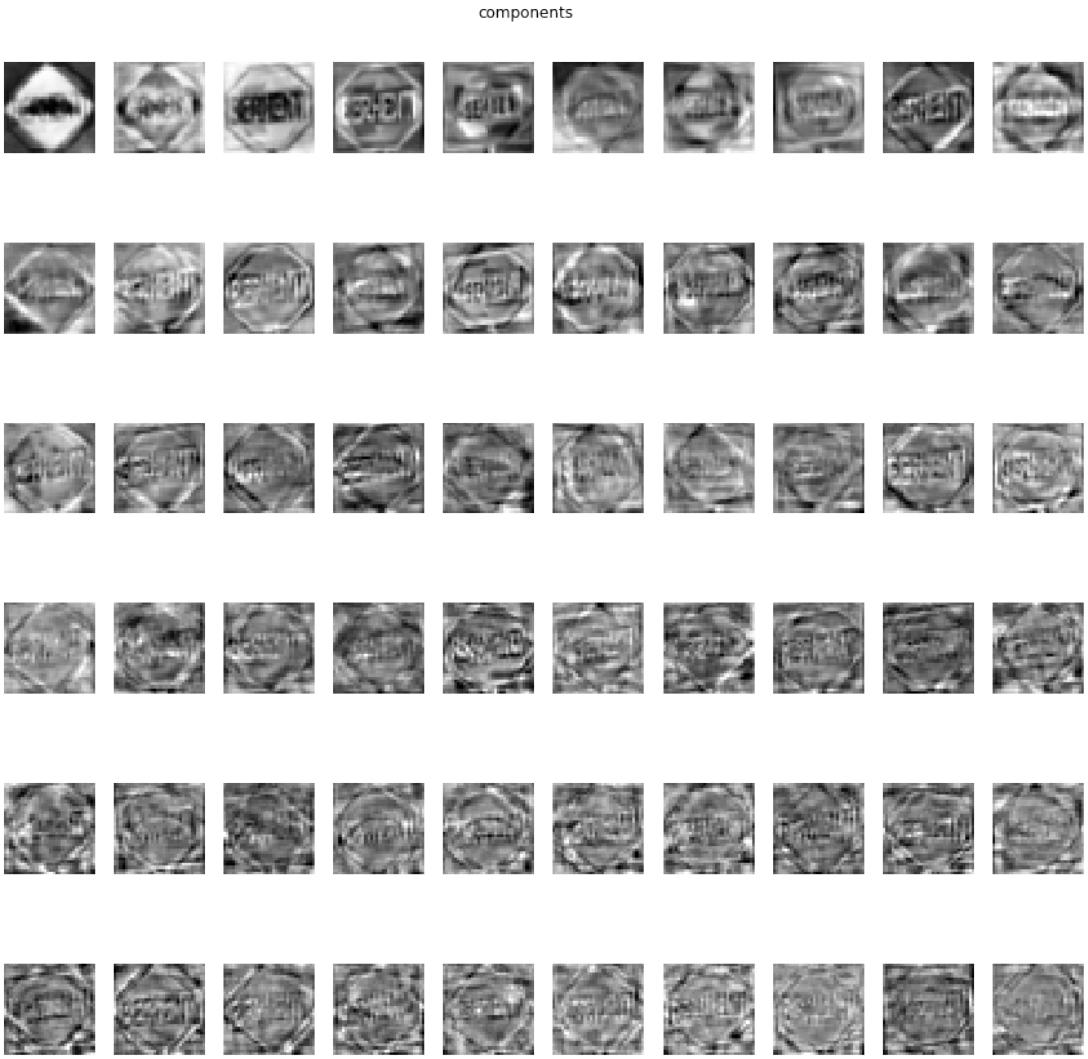
```
Iteration 97, loss = 0.04364900
Iteration 98, loss = 0.04264819
Iteration 99, loss = 0.04167575
Iteration 100, loss = 0.04074039
Iteration 101, loss = 0.03982722
Iteration 102, loss = 0.03895407
Iteration 103, loss = 0.03810635
Iteration 104, loss = 0.03728049
Iteration 105, loss = 0.03649334
Iteration 106, loss = 0.03572409
Iteration 107, loss = 0.03498385
Iteration 108, loss = 0.03426121
Iteration 109, loss = 0.03357038
Iteration 110, loss = 0.03290137
Iteration 111, loss = 0.03224732
Iteration 112, loss = 0.03161448
Iteration 113, loss = 0.03100187
Iteration 114, loss = 0.03040424
Iteration 115, loss = 0.02982716
Iteration 116, loss = 0.02926438
Iteration 117, loss = 0.02871491
Iteration 118, loss = 0.02818012
Iteration 119, loss = 0.02766487
Iteration 120, loss = 0.02716118
Iteration 121, loss = 0.02667189
Iteration 122, loss = 0.02619954
Iteration 123, loss = 0.02573552
Iteration 124, loss = 0.02528675
Iteration 125, loss = 0.02484742
Iteration 126, loss = 0.02442169
Iteration 127, loss = 0.02400562
Iteration 128, loss = 0.02360116
Iteration 129, loss = 0.02320579
Iteration 130, loss = 0.02282055
Iteration 131, loss = 0.02244541
Iteration 132, loss = 0.02207676
Iteration 133, loss = 0.02172153
Iteration 134, loss = 0.02137126
Iteration 135, loss = 0.02103388
Iteration 136, loss = 0.02069521
Iteration 137, loss = 0.02037435
Iteration 138, loss = 0.02005412
Iteration 139, loss = 0.01974775
Iteration 140, loss = 0.01944508
Iteration 141, loss = 0.01915155
Iteration 142, loss = 0.01886128
Iteration 143, loss = 0.01858626
Iteration 144, loss = 0.01831068
```

```
Iteration 145, loss = 0.01804216
Iteration 146, loss = 0.01778356
Iteration 147, loss = 0.01752731
Iteration 148, loss = 0.01727708
Iteration 149, loss = 0.01703075
Iteration 150, loss = 0.01679086
Iteration 151, loss = 0.01655476
Iteration 152, loss = 0.01632182
Iteration 153, loss = 0.01609595
Iteration 154, loss = 0.01587144
Iteration 155, loss = 0.01565730
Iteration 156, loss = 0.01544092
Iteration 157, loss = 0.01523311
Iteration 158, loss = 0.01503036
Iteration 159, loss = 0.01482716
Iteration 160, loss = 0.01463442
Iteration 161, loss = 0.01444075
Iteration 162, loss = 0.01425097
Iteration 163, loss = 0.01406619
Iteration 164, loss = 0.01388294
Iteration 165, loss = 0.01370705
Iteration 166, loss = 0.01352925
Iteration 167, loss = 0.01335710
Iteration 168, loss = 0.01319002
Iteration 169, loss = 0.01302474
Iteration 170, loss = 0.01286354
Iteration 171, loss = 0.01270369
Iteration 172, loss = 0.01255034
Iteration 173, loss = 0.01239815
Iteration 174, loss = 0.01224945
Iteration 175, loss = 0.01210364
Iteration 176, loss = 0.01196057
Iteration 177, loss = 0.01181924
Iteration 178, loss = 0.01168061
Iteration 179, loss = 0.01154376
Iteration 180, loss = 0.01141052
Iteration 181, loss = 0.01127624
Iteration 182, loss = 0.01114614
Iteration 183, loss = 0.01101900
Iteration 184, loss = 0.01089146
Iteration 185, loss = 0.01076805
Iteration 186, loss = 0.01064643
Iteration 187, loss = 0.01052543
Iteration 188, loss = 0.01040803
Iteration 189, loss = 0.01029113
Iteration 190, loss = 0.01017695
Iteration 191, loss = 0.01006389
Iteration 192, loss = 0.00995209
```

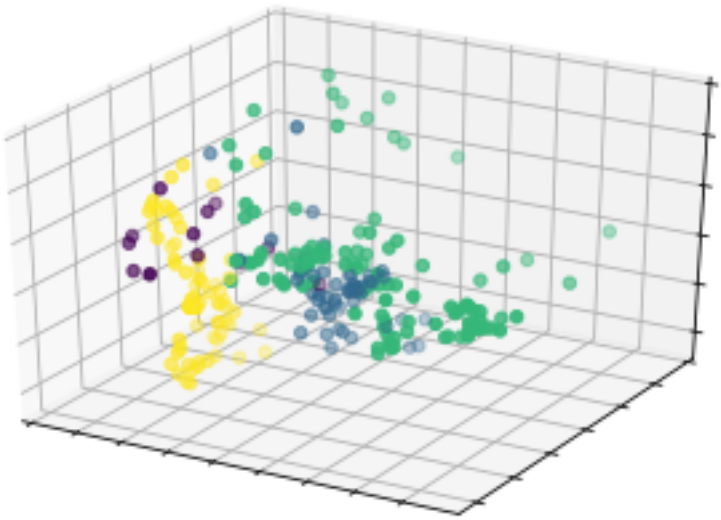
```
Iteration 193, loss = 0.00984188
Iteration 194, loss = 0.00973468
Iteration 195, loss = 0.00962697
Iteration 196, loss = 0.00952148
Iteration 197, loss = 0.00941733
Iteration 198, loss = 0.00931617
Iteration 199, loss = 0.00921347
Iteration 200, loss = 0.00911338


C:\Users\GaryNg\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:562
  % self.max_iter, ConvergenceWarning)


MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.988
confusion matrix:
 [[ 13   0   1   0]
 [  1  41   0   0]
 [  0   1 117   0]
 [  0   0   0  76]]
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        14
           1       0.98      0.98      0.98        42
           2       0.99      0.99      0.99       118
           3       1.00      1.00      1.00        76

   micro avg       0.99      0.99      0.99       250
   macro avg       0.97      0.97      0.97       250
weighted avg       0.99      0.99      0.99       250

--- Model 10 ---
trainingData: (250, 1024), trainingLabels: (250,)
testingData: (250, 1024), testingLabels: (250,)
total explained variance: 0.9535200755454445
```

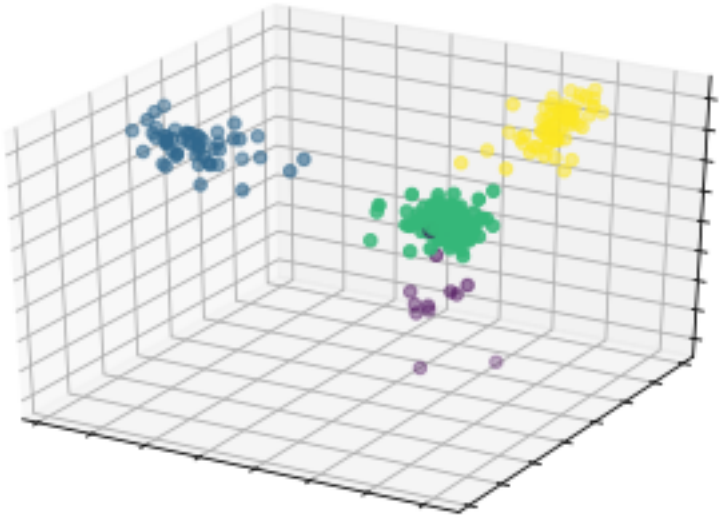



```
pca components: (64, 1024)
```




```
trainingData: (250, 64), trainingLabels: (250,)
testingData: (250, 64), testingLabels: (250,)
```





```
trainingData: (250, 3), trainingLabels: (250,)
testingData: (250, 3), testingLabels: (250,)
Iteration 1, loss = 1.31246071
Iteration 2, loss = 0.64566688
Iteration 3, loss = 0.29128101
Iteration 4, loss = 0.11840961
```

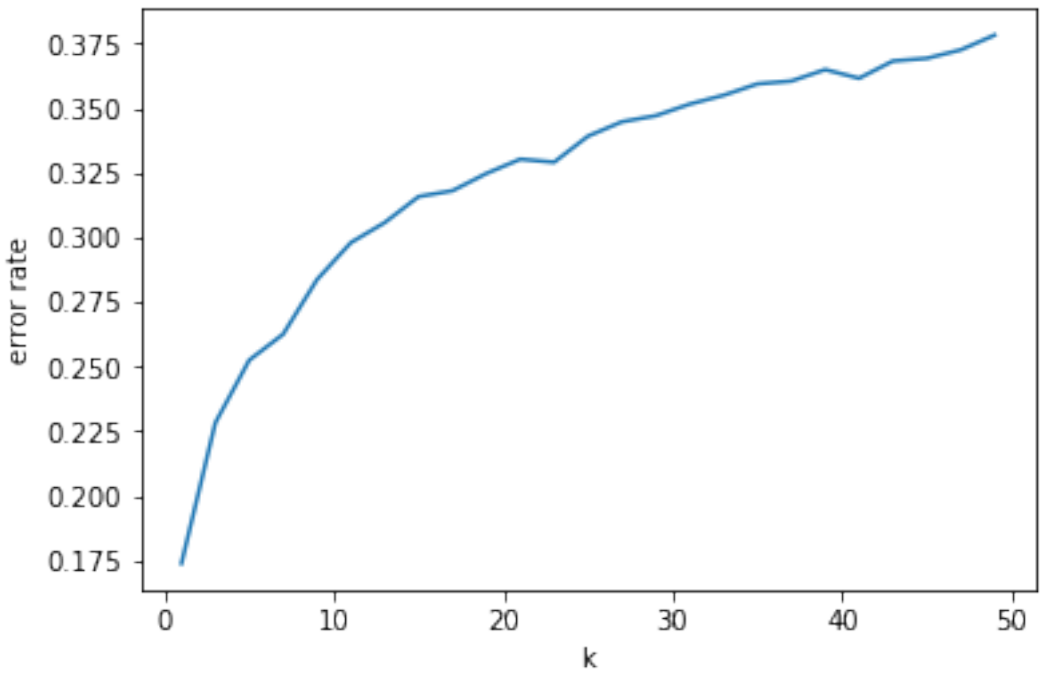
```
Iteration 5, loss = 0.04729502
Iteration 6, loss = 0.02043329
Iteration 7, loss = 0.00981577
Iteration 8, loss = 0.00555157
Iteration 9, loss = 0.00349337
Iteration 10, loss = 0.00239950
Iteration 11, loss = 0.00178487
Iteration 12, loss = 0.00142056
Iteration 13, loss = 0.00118115
Iteration 14, loss = 0.00102018
Iteration 15, loss = 0.00090841
Iteration 16, loss = 0.00082586
Iteration 17, loss = 0.00077207
Iteration 18, loss = 0.00072650
Iteration 19, loss = 0.00069445
Iteration 20, loss = 0.00066795
Iteration 21, loss = 0.00064792
Iteration 22, loss = 0.00062995
Iteration 23, loss = 0.00061679
Iteration 24, loss = 0.00060514
Iteration 25, loss = 0.00059532
Iteration 26, loss = 0.00058531
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.992
confusion matrix:
 [[ 13   0   0   1]
 [  0  42   0   0]
 [  0   1 117   0]
 [  0   0   0  76]]
              precision    recall  f1-score   support

           0       1.00      0.93      0.96        14
           1       0.98      1.00      0.99        42
           2       1.00      0.99      1.00       118
           3       0.99      1.00      0.99        76

   micro avg       0.99      0.99      0.99       250
   macro avg       0.99      0.98      0.99       250
weighted avg       0.99      0.99      0.99       250
```

## 11 Data augmentation

In [48]:
```
def unpickleAugmentedData():
    images = pickle.load(open("./pkl/images_aug.pkl", "rb"))
    labels = pickle.load(open("./pkl/labels_aug.pkl", "rb"))
    return (images, labels)

def getAugmentedData():
    """Return unpicked data seperated into training and testing sets"""
    return train_test_split(*unpickleAugmentedData(), random_state=123)
```

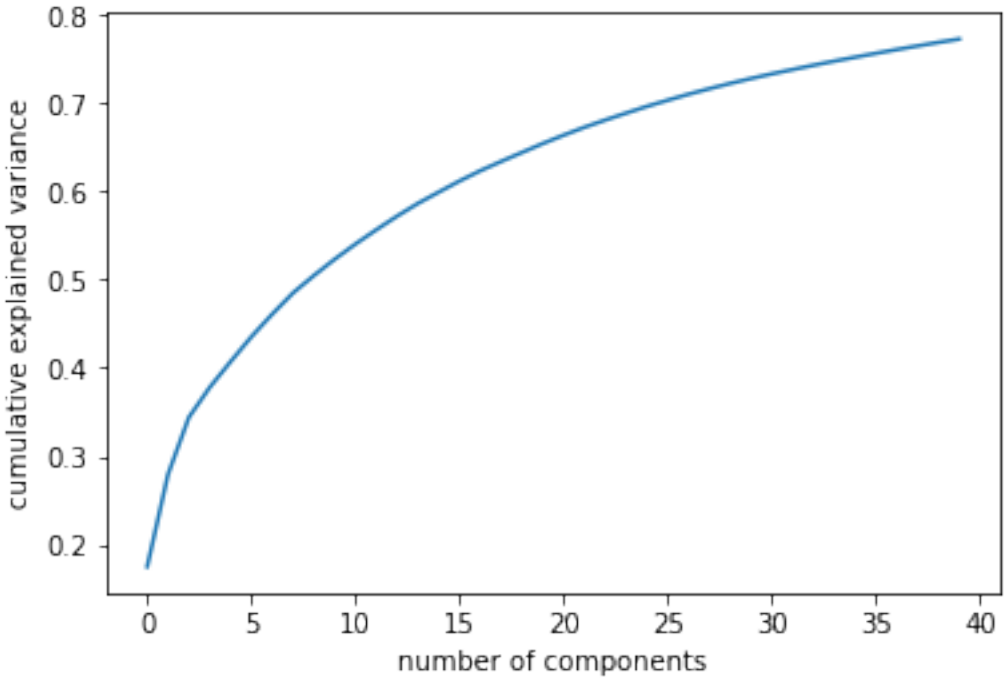
In [63]:
```
images, _ = unpickleAugmentedData()
showImages(np.array(sample(list(images), 50)).reshape(-1, 32, 32), "Augmented images"
```

```
In [64]: print("--- Model 1 ---")
         trainKnn(*getAugmentedData())

--- Model 1 ---
trainingData: (900, 1024), trainingLabels: (900,)
testingData: (300, 1024), testingLabels: (300,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.8
confusion matrix:
 [[83  0  0  1]
 [24 44  0  3]
 [ 6  2 66  3]
 [20  1  0 47]]
              precision    recall  f1-score   support

           0       0.62      0.99      0.76        84
           1       0.94      0.62      0.75        71
           2       1.00      0.86      0.92        77
           3       0.87      0.69      0.77        68

   micro avg       0.80      0.80      0.80       300
   macro avg       0.86      0.79      0.80       300
weighted avg       0.85      0.80      0.80       300
```



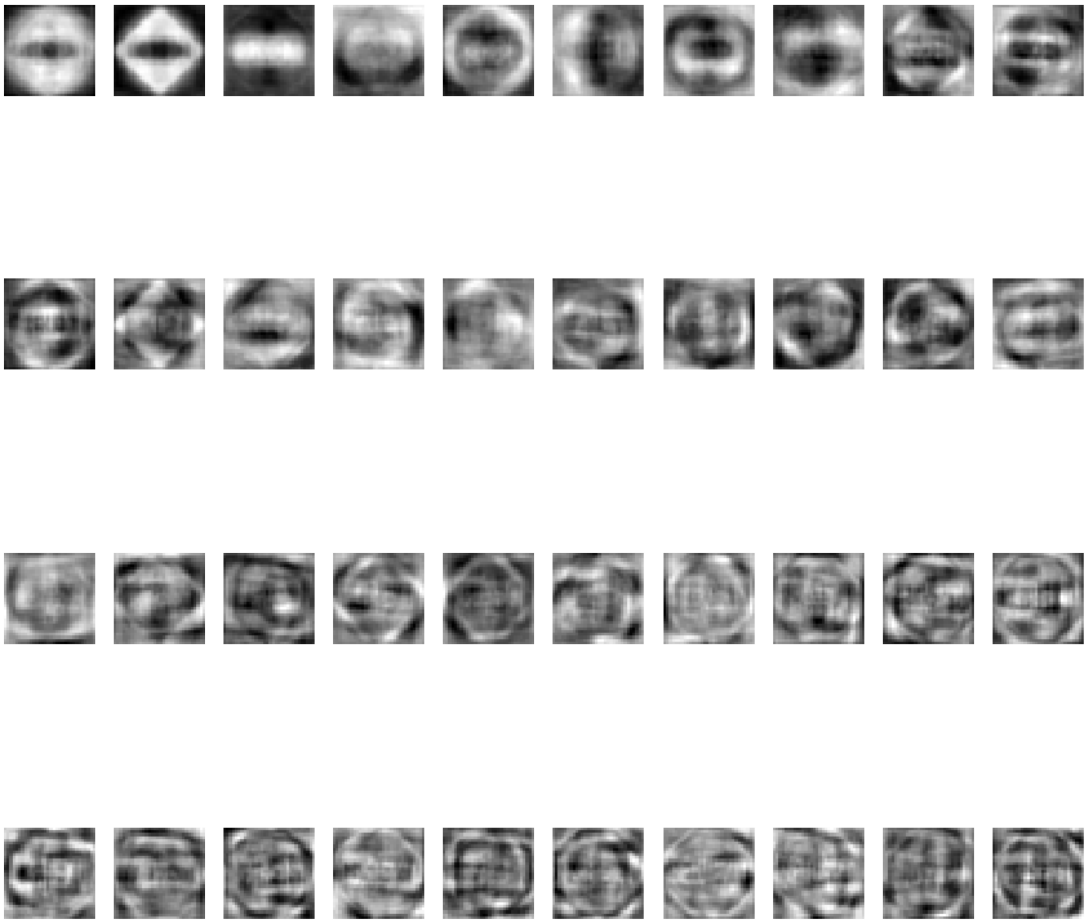
```
In [109]: print("--- Model 2 ---")
          trainKnn(*fitPca(*getAugmentedData(), components=40))
```

```
--- Model 2 ---
trainingData: (900, 1024), trainingLabels: (900,)
testingData: (300, 1024), testingLabels: (300,)
total explained variance: 0.7727455300589892
```



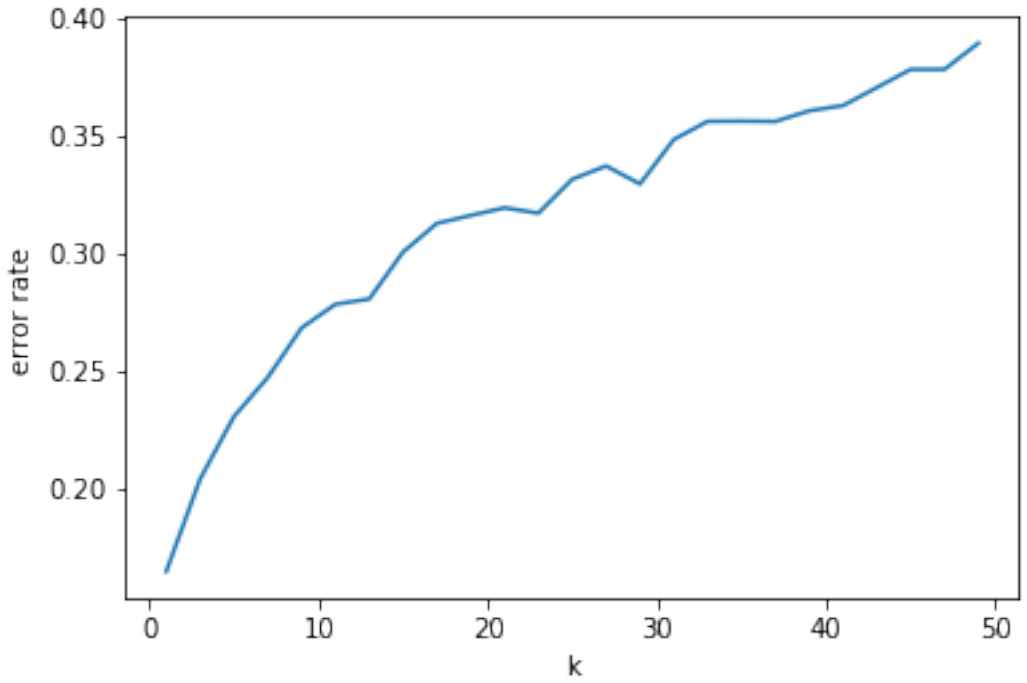

```
pca components: (40, 1024)
```

components

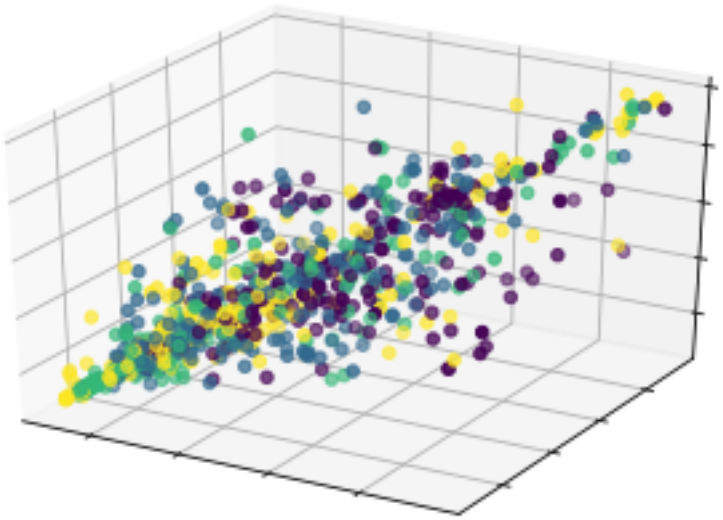


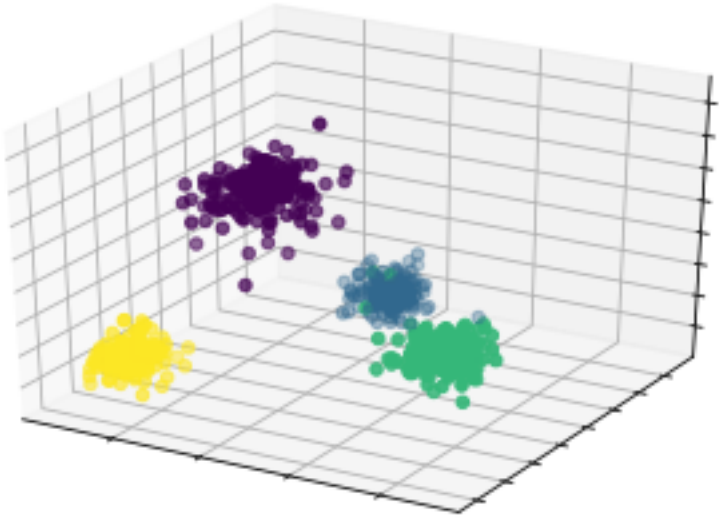
```
trainingData: (900, 40), trainingLabels: (900,)
testingData: (300, 40), testingLabels: (300,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.86
confusion matrix:
 [[78  0  1  5]
 [12 47  4  8]
 [ 3  1 68  5]
 [ 2  0  1 65]]
              precision    recall  f1-score   support

           0       0.82      0.93      0.87        84
           1       0.98      0.66      0.79        71
           2       0.92      0.88      0.90        77
```

```
              3       0.78      0.96      0.86        68

    micro avg       0.86      0.86      0.86       300
    macro avg       0.88      0.86      0.86       300
weighted avg       0.87      0.86      0.86       300
```



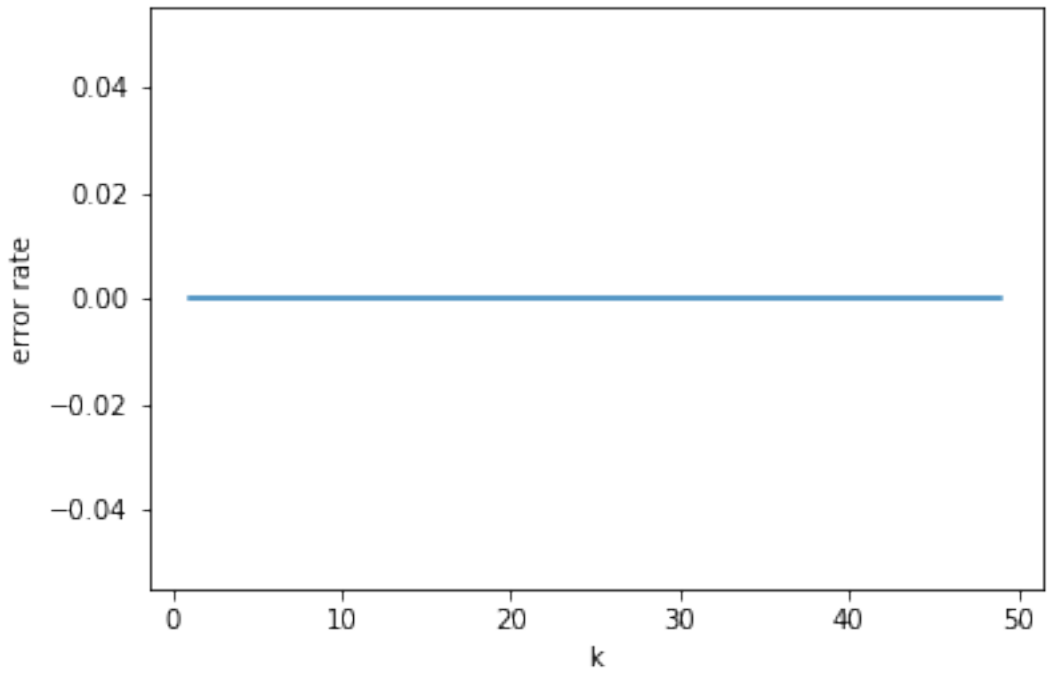

```
In [66]: print("--- Model 3 ---")
         trainKnn(*fitLda(*getAugmentedData()))

--- Model 3 ---
trainingData: (900, 1024), trainingLabels: (900,)
testingData: (300, 1024), testingLabels: (300,)


C:\Users\GaryNg\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning:
  warnings.warn("Variables are collinear.")
```



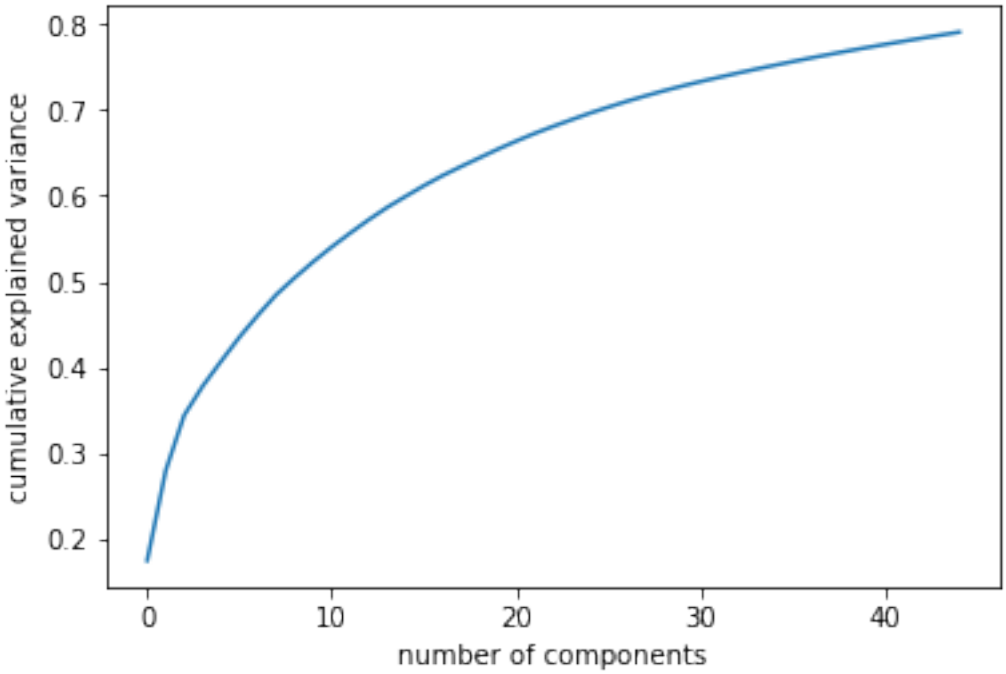


```
trainingData: (900, 3), trainingLabels: (900,)
testingData: (300, 3), testingLabels: (300,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.4766666666666667
confusion matrix:
```

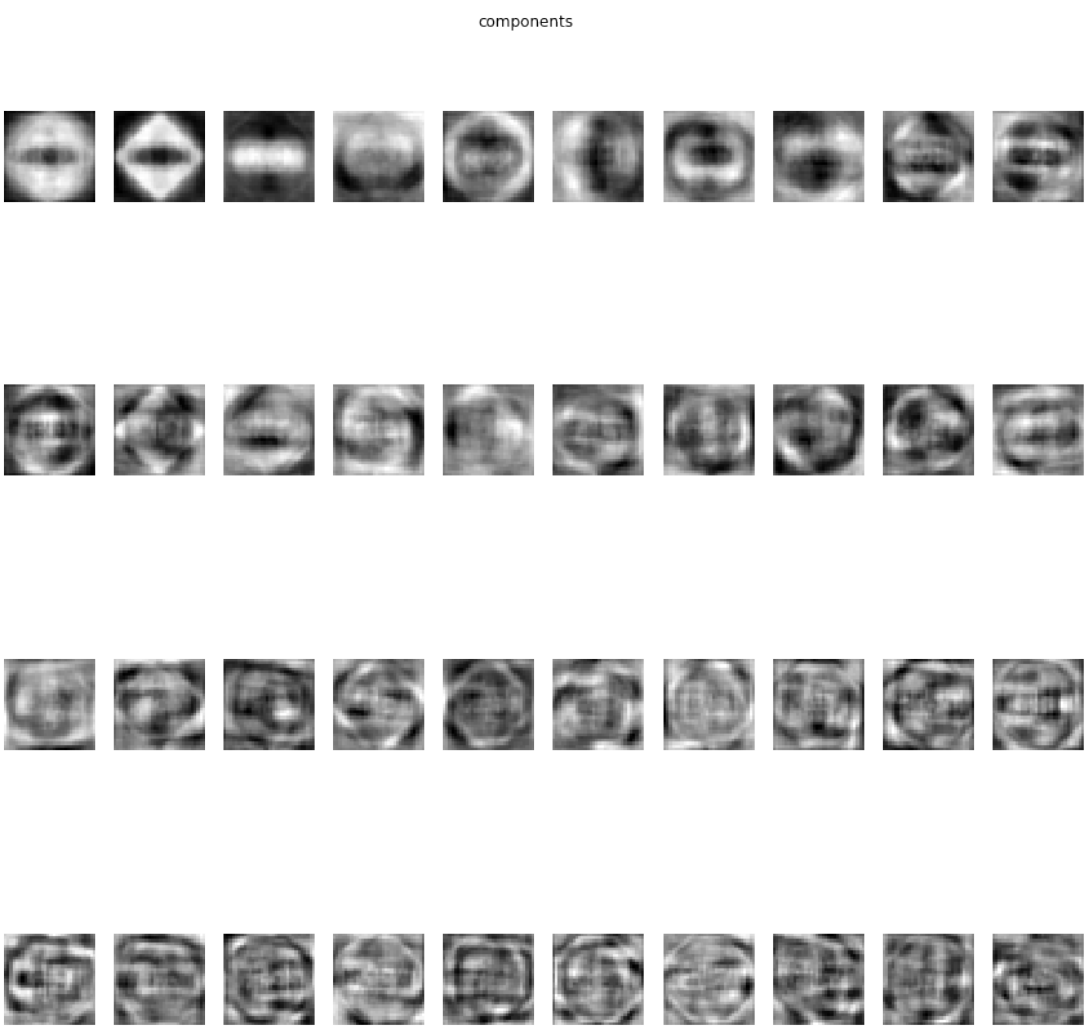
```
 [[55  7  6 16]
 [21 27 14  9]
 [20  9 31 17]
 [19 10  9 30]]
              precision    recall  f1-score   support

           0       0.48      0.65      0.55        84
           1       0.51      0.38      0.44        71
           2       0.52      0.40      0.45        77
           3       0.42      0.44      0.43        68

   micro avg       0.48      0.48      0.48       300
   macro avg       0.48      0.47      0.47       300
weighted avg       0.48      0.48      0.47       300
```



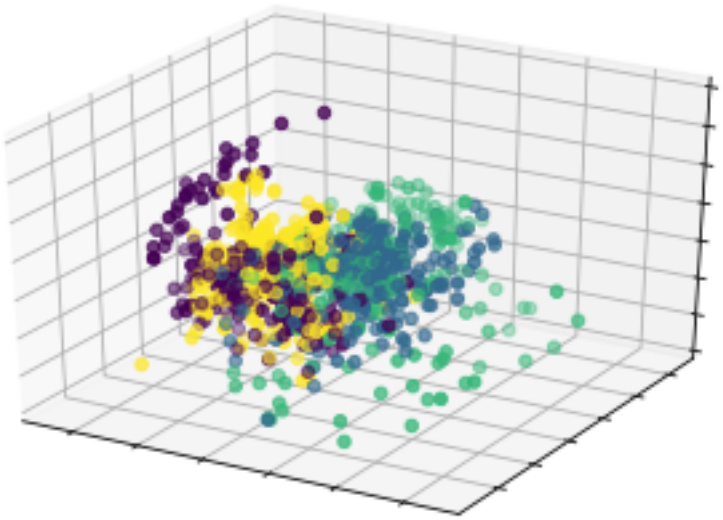

```
In [118]: print("--- Model 4 ---")
          trainKnn(*fitLda(*fitPca(*getAugmentedData(), components=45)))

--- Model 4 ---
trainingData: (900, 1024), trainingLabels: (900,)
testingData: (300, 1024), testingLabels: (300,)
total explained variance: 0.790802143782052
```

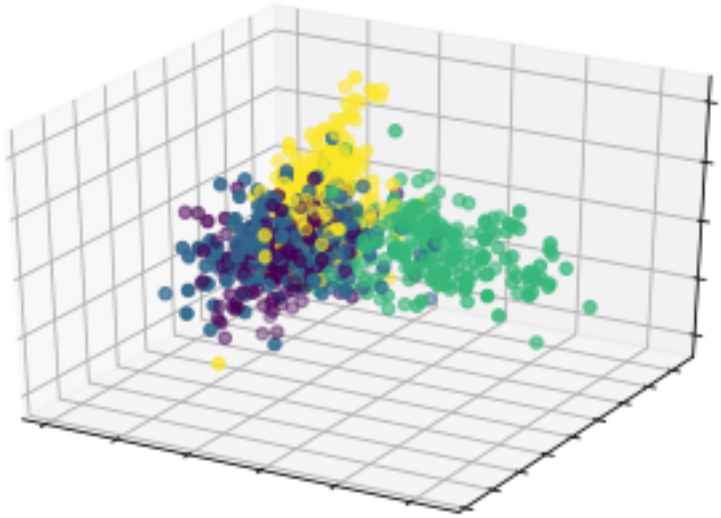



```
pca components: (45, 1024)
```



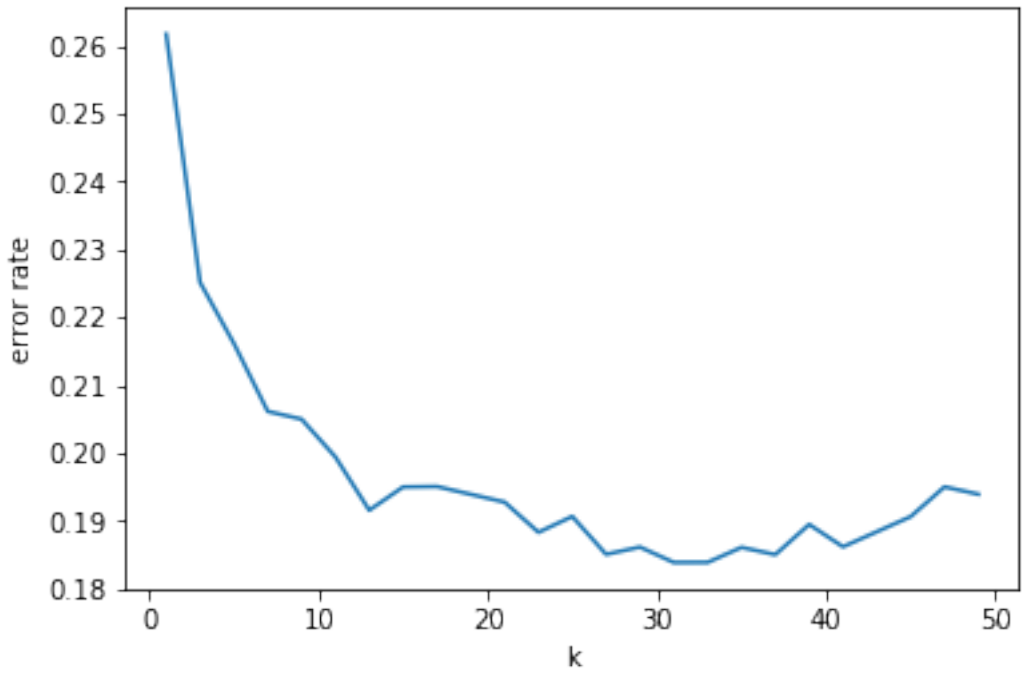

```
trainingData: (900, 45), trainingLabels: (900,)
testingData: (300, 45), testingLabels: (300,)
```



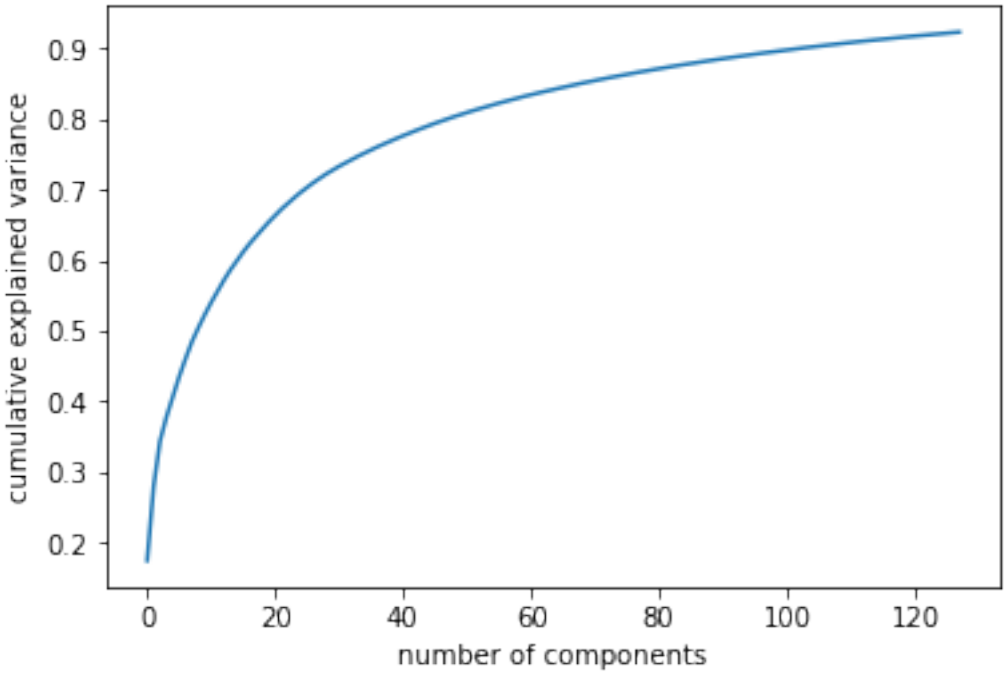


```
trainingData: (900, 3), trainingLabels: (900,)
testingData: (300, 3), testingLabels: (300,)
optimal k: 31
training accuracy: 0.8277777777777777
testing accuracy: 0.7366666666666667
confusion matrix:
```

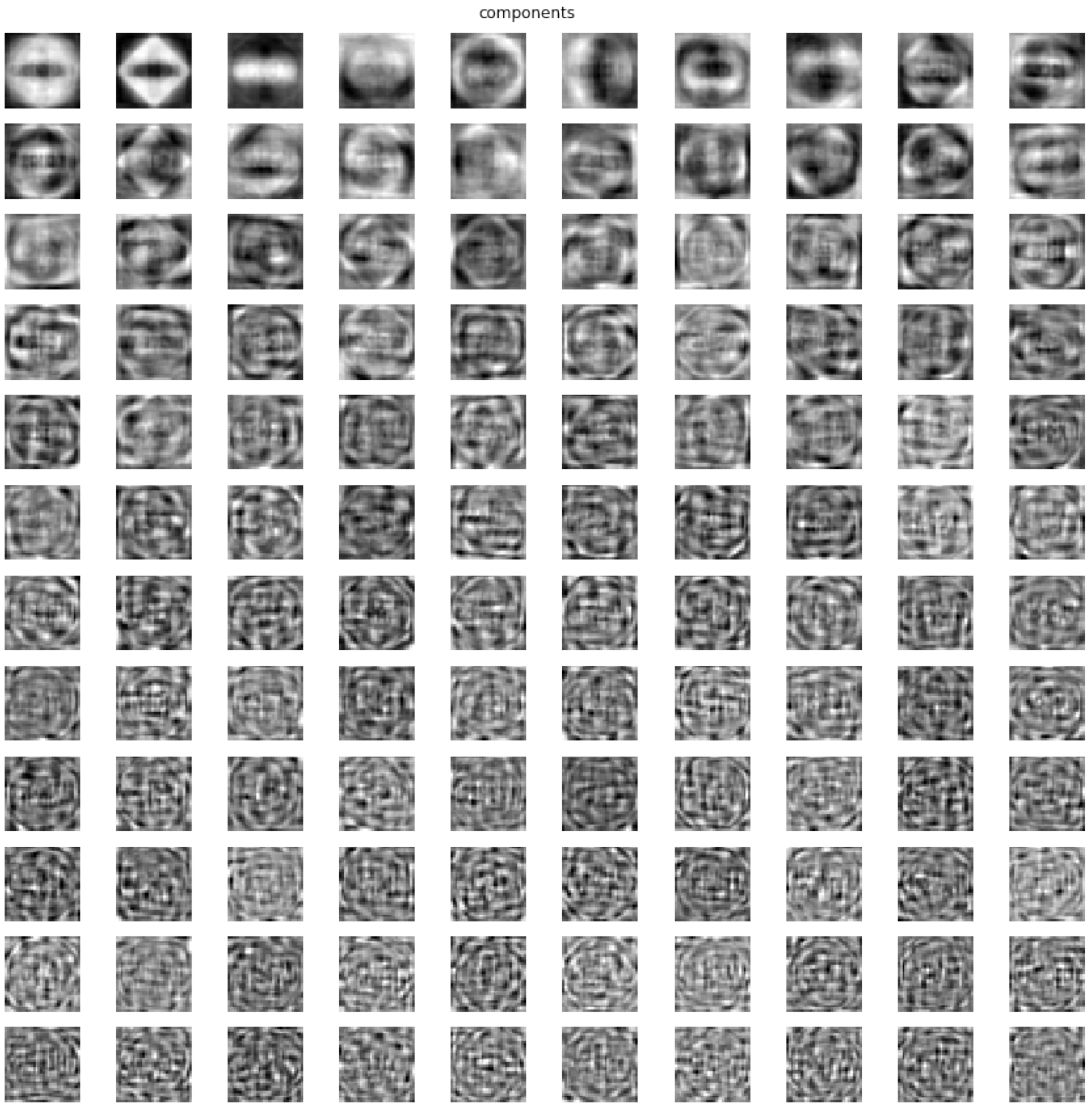
```
 [[60  8  0 16]
 [ 9 50  9  3]
 [ 2  8 64  3]
 [11  7  3 47]]
              precision    recall  f1-score   support

           0       0.73      0.71      0.72        84
           1       0.68      0.70      0.69        71
           2       0.84      0.83      0.84        77
           3       0.68      0.69      0.69        68

   micro avg       0.74      0.74      0.74       300
   macro avg       0.73      0.74      0.74       300
weighted avg       0.74      0.74      0.74       300
```




```
In [68]: print("--- Model 5 ---")
         trainSvm(*fitPca(*getAugmentedData(), components=128))

--- Model 5 ---
trainingData: (900, 1024), trainingLabels: (900,)
testingData: (300, 1024), testingLabels: (300,)
total explained variance: 0.9234252523056466
```




```
pca components: (128, 1024)
```

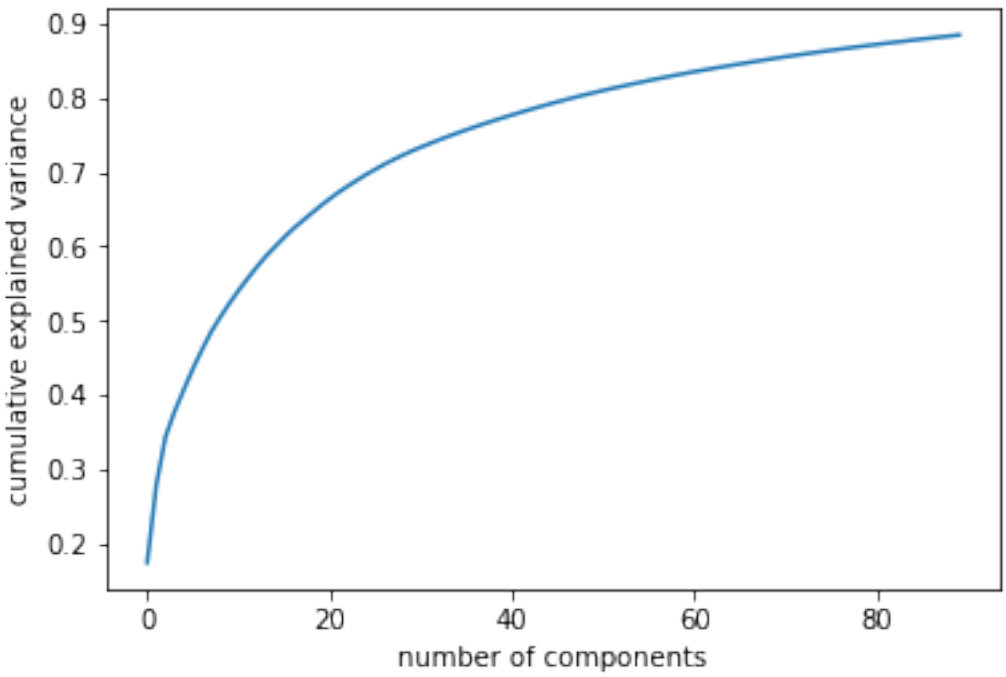



```
trainingData: (900, 128), trainingLabels: (900,)
testingData: (300, 128), testingLabels: (300,)
training accuracy: 0.9822222222222222
testing accuracy: 0.7266666666666667
confusion matrix:
 [[65  6  2 11]
 [12 43 12  4]
 [ 5  9 61  2]
 [10  6  3 49]]
              precision    recall  f1-score   support

           0       0.71      0.77      0.74        84
           1       0.67      0.61      0.64        71
```

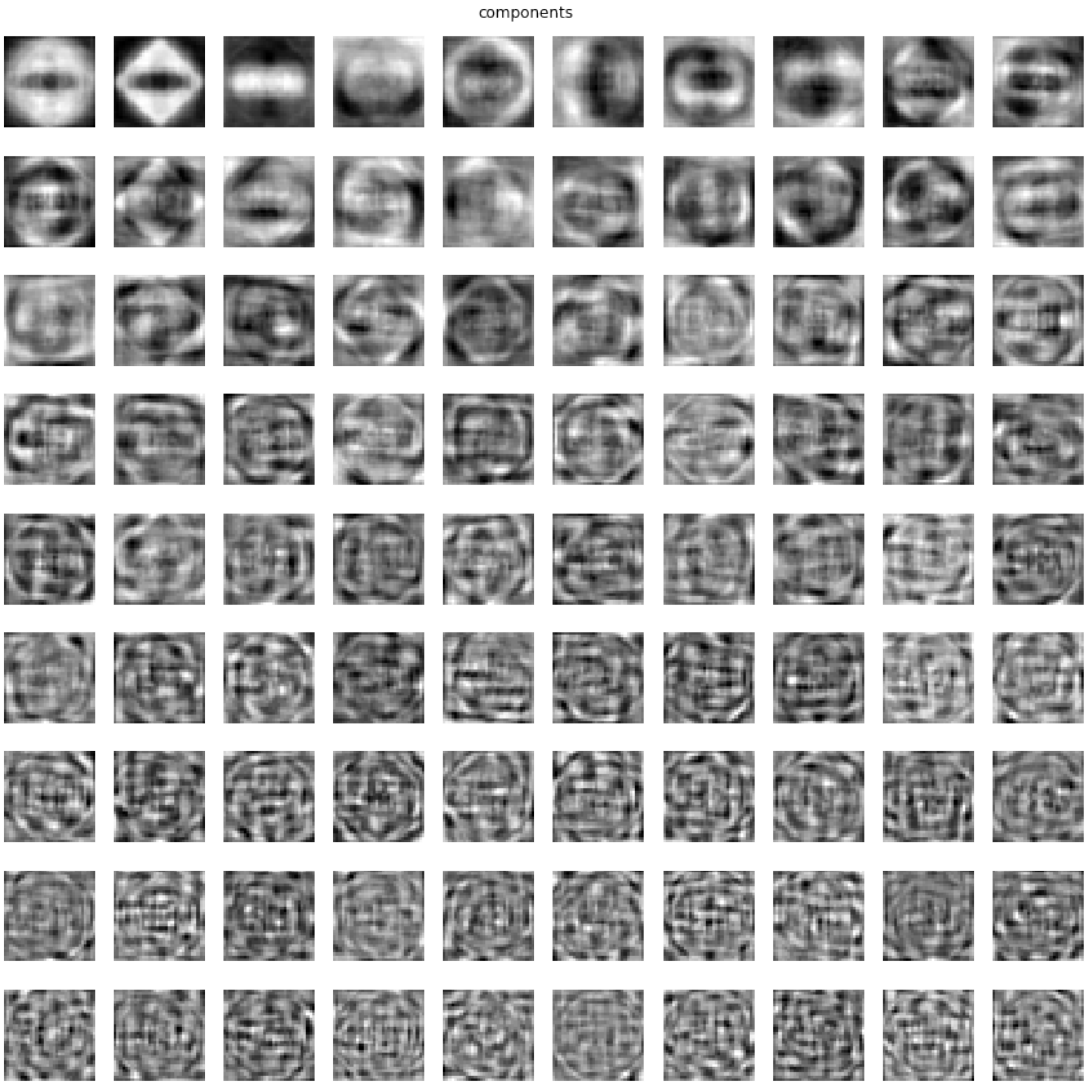
```
           2       0.78      0.79      0.79        77
           3       0.74      0.72      0.73        68

   micro avg       0.73      0.73      0.73       300
   macro avg       0.73      0.72      0.72       300
weighted avg       0.73      0.73      0.73       300
```

```
In [127]: print("--- Model 6 ---")
          trainSvm(*fitLda(*fitPca(*getAugmentedData(), components=90)))
```

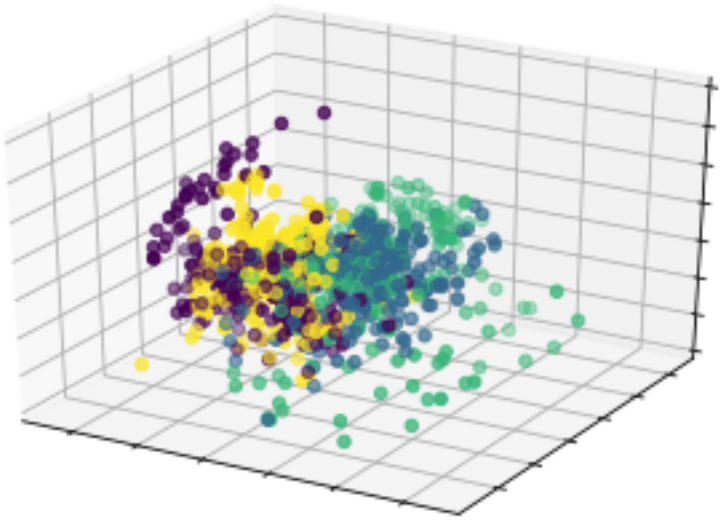
```
--- Model 6 ---
trainingData: (900, 1024), trainingLabels: (900,)
testingData: (300, 1024), testingLabels: (300,)
total explained variance: 0.8838914495470591
```

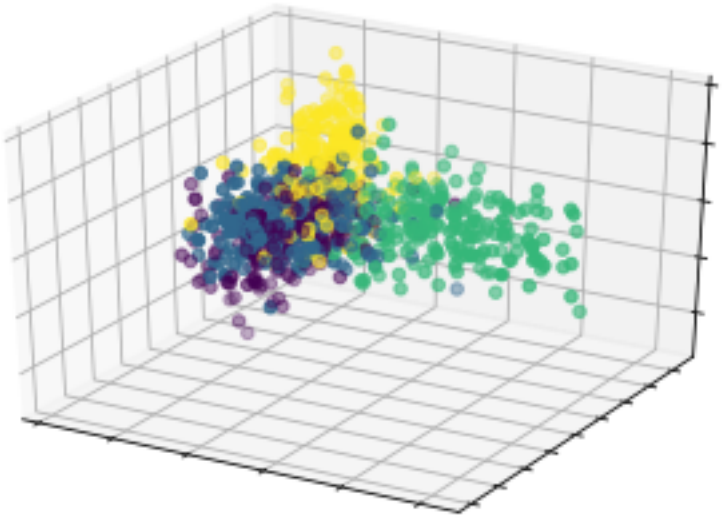



```
pca components: (90, 1024)
```




```
trainingData: (900, 90), trainingLabels: (900,)
testingData: (300, 90), testingLabels: (300,)
```





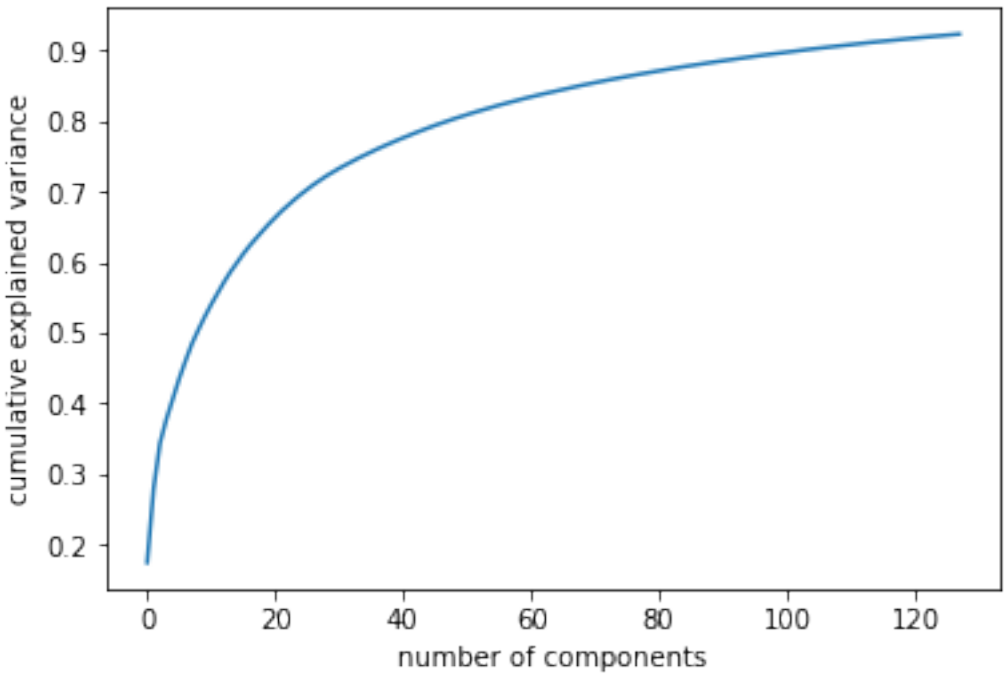
```
trainingData: (900, 3), trainingLabels: (900,)
testingData: (300, 3), testingLabels: (300,)
training accuracy: 0.84
testing accuracy: 0.7266666666666667
confusion matrix:
 [[61  7  1 15]
```

```
 [ 5 50 10  6]
 [ 2  6 63  6]
 [13  6  5 44]]
              precision    recall  f1-score   support

           0       0.75      0.73      0.74        84
           1       0.72      0.70      0.71        71
           2       0.80      0.82      0.81        77
           3       0.62      0.65      0.63        68

   micro avg       0.73      0.73      0.73       300
   macro avg       0.72      0.72      0.72       300
weighted avg       0.73      0.73      0.73       300
```

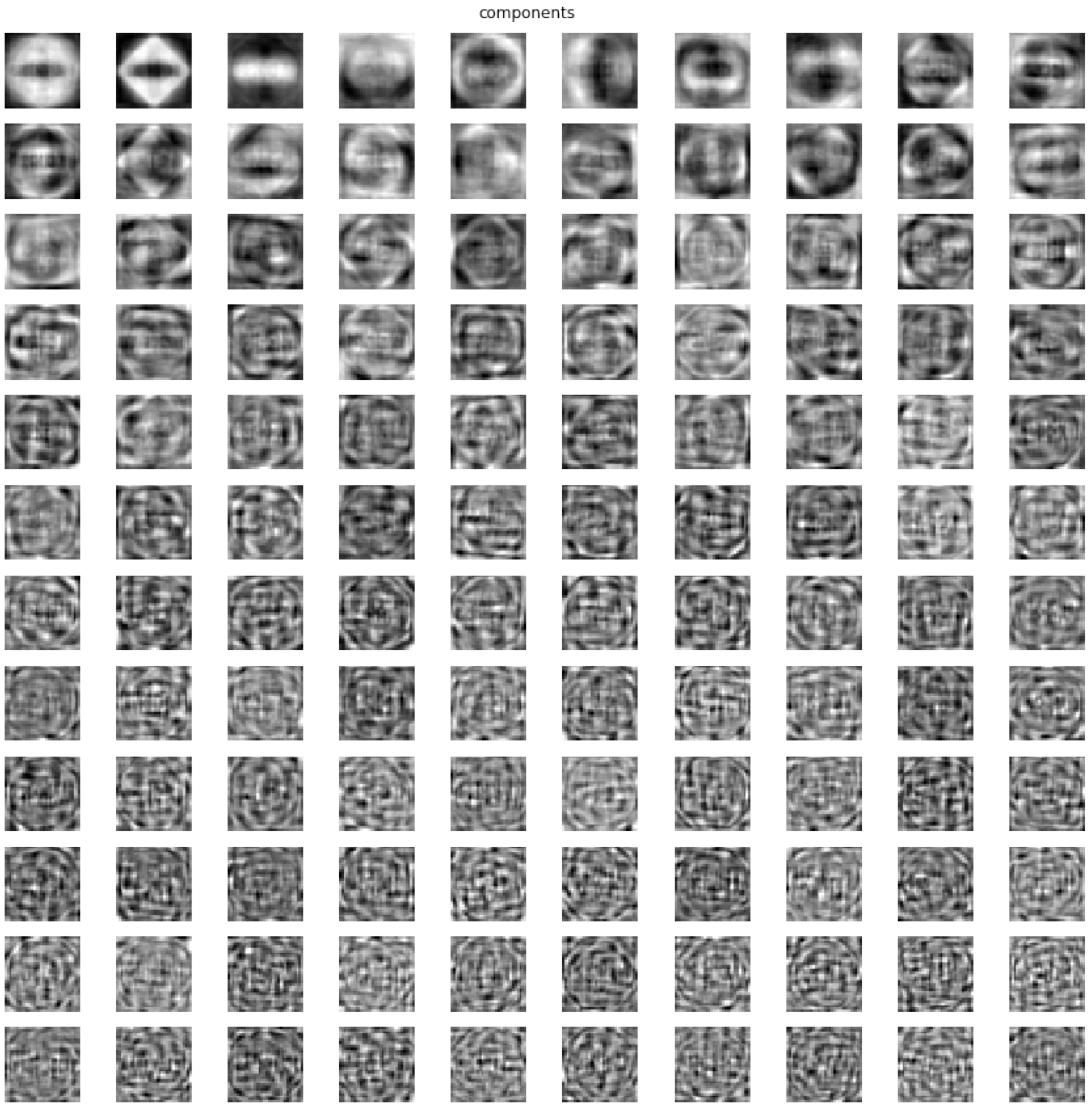
```
In [70]: print("--- Model 7 ---")
         trainMlp(*getAugmentedData())

--- Model 7 ---
trainingData: (900, 1024), trainingLabels: (900,)
testingData: (300, 1024), testingLabels: (300,)
Iteration 1, loss = 1.35154188
Iteration 2, loss = 0.49650054
Iteration 3, loss = 0.29579430
Iteration 4, loss = 0.15202952
Iteration 5, loss = 0.10412936
Iteration 6, loss = 0.05484465
Iteration 7, loss = 0.03364861
Iteration 8, loss = 0.01826231
Iteration 9, loss = 0.01252340
Iteration 10, loss = 0.00803611
Iteration 11, loss = 0.00566265
Iteration 12, loss = 0.00443914
Iteration 13, loss = 0.00352818
Iteration 14, loss = 0.00279497
Iteration 15, loss = 0.00243120
Iteration 16, loss = 0.00218756
Iteration 17, loss = 0.00196311
Iteration 18, loss = 0.00180930
Iteration 19, loss = 0.00169482
Iteration 20, loss = 0.00159892
Iteration 21, loss = 0.00151244
Iteration 22, loss = 0.00144458
Iteration 23, loss = 0.00138048
Iteration 24, loss = 0.00132519
Iteration 25, loss = 0.00127449
Iteration 26, loss = 0.00122991
```

```
Iteration 27, loss = 0.00119227
Iteration 28, loss = 0.00115656
Iteration 29, loss = 0.00112602
Iteration 30, loss = 0.00109343
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.87
confusion matrix:
 [[72  4  3  5]
 [ 2 61  6  2]
 [ 2  3 70  2]
 [ 5  3  2 58]]
              precision    recall  f1-score   support

           0       0.89      0.86      0.87        84
           1       0.86      0.86      0.86        71
           2       0.86      0.91      0.89        77
           3       0.87      0.85      0.86        68

   micro avg       0.87      0.87      0.87       300
   macro avg       0.87      0.87      0.87       300
weighted avg       0.87      0.87      0.87       300
```

```
In [71]: print("--- Model 8 ---")
         trainMlp(*fitPca(*getAugmentedData(), components=128))

--- Model 8 ---
trainingData: (900, 1024), trainingLabels: (900,)
testingData: (300, 1024), testingLabels: (300,)
total explained variance: 0.9235992604060625
```




```
pca components: (128, 1024)
```

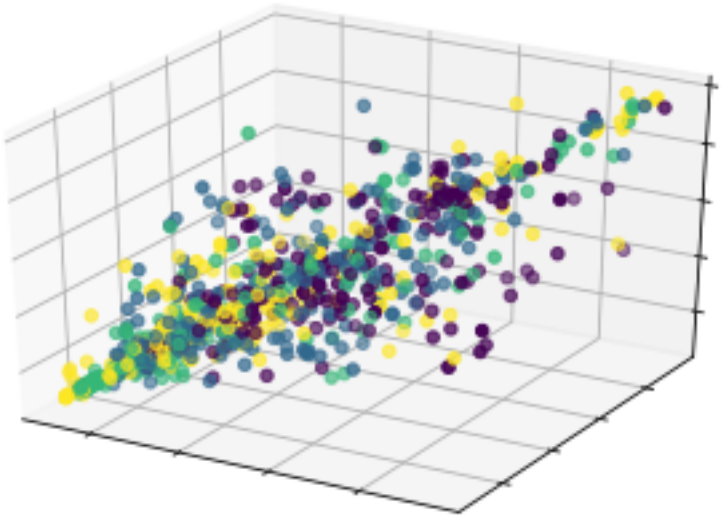


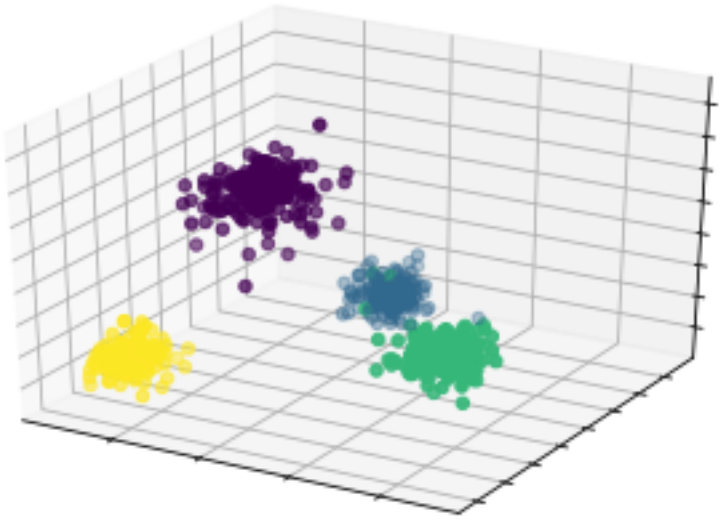

```
trainingData: (900, 128), trainingLabels: (900,)
testingData: (300, 128), testingLabels: (300,)
Iteration 1, loss = 1.26139698
Iteration 2, loss = 0.55945470
Iteration 3, loss = 0.28656853
Iteration 4, loss = 0.14386682
Iteration 5, loss = 0.07427431
Iteration 6, loss = 0.03957934
Iteration 7, loss = 0.02068012
Iteration 8, loss = 0.01125766
Iteration 9, loss = 0.00722166
Iteration 10, loss = 0.00474354
Iteration 11, loss = 0.00354336
```

```
Iteration 12, loss = 0.00285381
Iteration 13, loss = 0.00237755
Iteration 14, loss = 0.00206020
Iteration 15, loss = 0.00185795
Iteration 16, loss = 0.00169633
Iteration 17, loss = 0.00157563
Iteration 18, loss = 0.00147325
Iteration 19, loss = 0.00139308
Iteration 20, loss = 0.00132222
Iteration 21, loss = 0.00125916
Iteration 22, loss = 0.00120502
Iteration 23, loss = 0.00115624
Iteration 24, loss = 0.00111343
Iteration 25, loss = 0.00107216
Iteration 26, loss = 0.00103547
Iteration 27, loss = 0.00100277
Iteration 28, loss = 0.00097089
Iteration 29, loss = 0.00094238
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.8
confusion matrix:
 [[70  5  3  6]
 [ 8 49  9  5]
 [ 2  3 67  5]
 [ 8  3  3 54]]
              precision    recall  f1-score   support

           0       0.80      0.83      0.81        84
           1       0.82      0.69      0.75        71
           2       0.82      0.87      0.84        77
           3       0.77      0.79      0.78        68

   micro avg       0.80      0.80      0.80       300
   macro avg       0.80      0.80      0.80       300
weighted avg       0.80      0.80      0.80       300
```

```
In [141]: print("--- Model 9 ---")
          trainMlp(*fitLda(*getAugmentedData()))
```

```
--- Model 9 ---
trainingData: (900, 1024), trainingLabels: (900,)
testingData: (300, 1024), testingLabels: (300,)


C:\Users\GaryNg\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning:
  warnings.warn("Variables are collinear.")
```

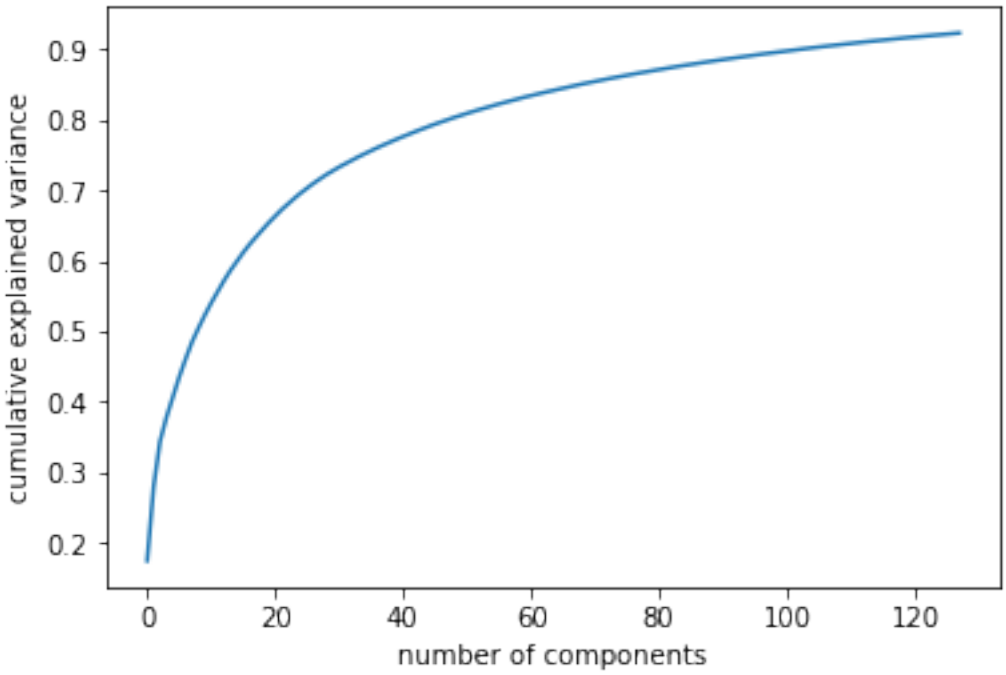




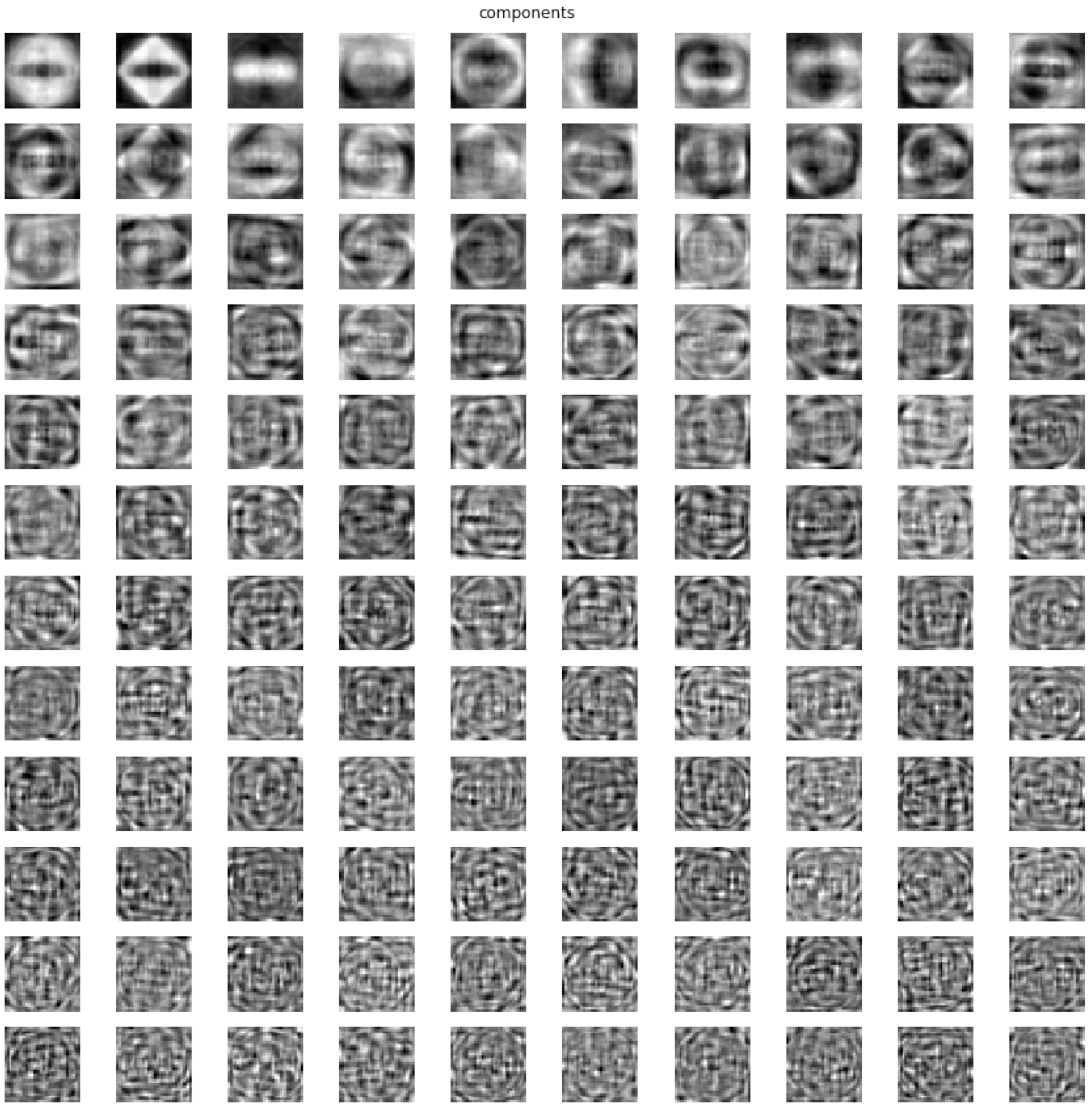
```
trainingData: (900, 3), trainingLabels: (900,)
testingData: (300, 3), testingLabels: (300,)
Iteration 1, loss = 0.78176426
Iteration 2, loss = 0.07771000
Iteration 3, loss = 0.00772085
Iteration 4, loss = 0.00213400
Iteration 5, loss = 0.00118128
Iteration 6, loss = 0.00085550
Iteration 7, loss = 0.00070513
Iteration 8, loss = 0.00060522
Iteration 9, loss = 0.00056849
Iteration 10, loss = 0.00051696
Iteration 11, loss = 0.00050107
Iteration 12, loss = 0.00047147
Iteration 13, loss = 0.00045711
Iteration 14, loss = 0.00044922
Iteration 15, loss = 0.00043888
Iteration 16, loss = 0.00042718
Iteration 17, loss = 0.00042067
Iteration 18, loss = 0.00041513
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.4766666666666667
confusion matrix:
 [[54  8  8 14]
 [20 27 15  9]
 [17  9 34 17]
 [17 11 12 28]]
              precision    recall  f1-score   support

           0       0.50      0.64      0.56        84
           1       0.49      0.38      0.43        71
           2       0.49      0.44      0.47        77
           3       0.41      0.41      0.41        68

   micro avg       0.48      0.48      0.48       300
   macro avg       0.47      0.47      0.47       300
weighted avg       0.48      0.48      0.47       300
```

```
In [143]: print("--- Model 10 ---")
          trainMlp(*fitLda(*fitPca(*getAugmentedData(), components=128)))
```

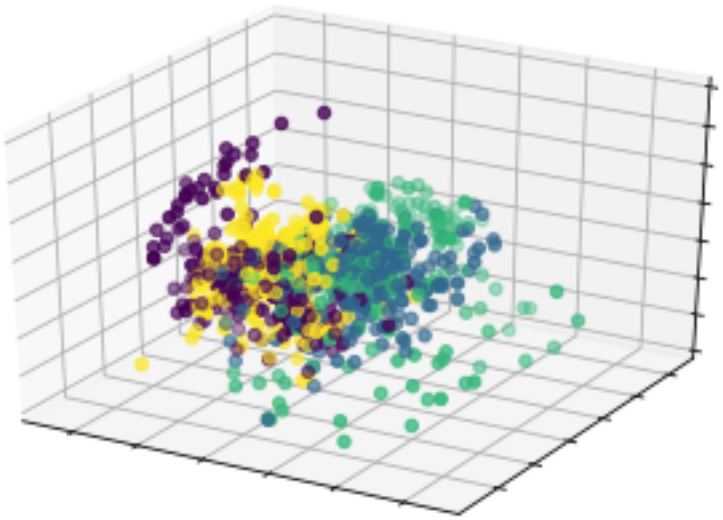
```
--- Model 10 ---
trainingData: (900, 1024), trainingLabels: (900,)
testingData: (300, 1024), testingLabels: (300,)
total explained variance: 0.9234791805381934
```

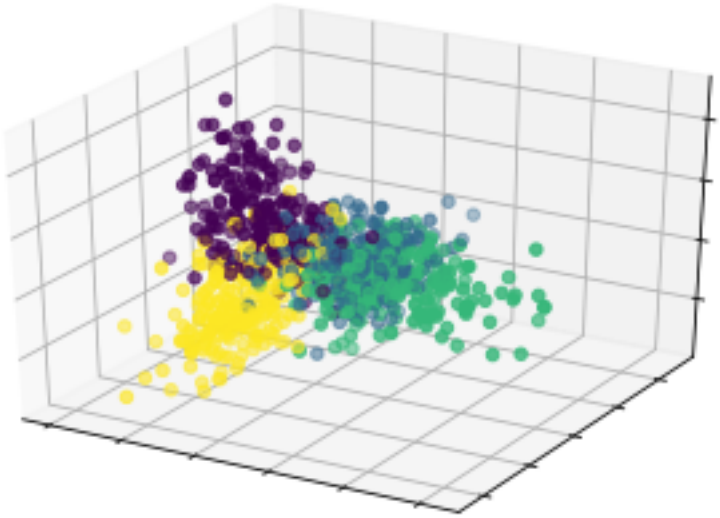



```
pca components: (128, 1024)
```




```
trainingData: (900, 128), trainingLabels: (900,)
testingData: (300, 128), testingLabels: (300,)
```





```
trainingData: (900, 3), trainingLabels: (900,)
testingData: (300, 3), testingLabels: (300,)
Iteration 1, loss = 0.95574992
Iteration 2, loss = 0.45584952
Iteration 3, loss = 0.42247576
Iteration 4, loss = 0.41833278
```

```
Iteration 5, loss = 0.40759649
Iteration 6, loss = 0.39845777
Iteration 7, loss = 0.39087872
Iteration 8, loss = 0.39088774
Iteration 9, loss = 0.38461050
Iteration 10, loss = 0.37997178
Iteration 11, loss = 0.37644285
Iteration 12, loss = 0.37595288
Iteration 13, loss = 0.37237338
Iteration 14, loss = 0.37350367
Iteration 15, loss = 0.37228014
Iteration 16, loss = 0.36902556
Iteration 17, loss = 0.36834605
Iteration 18, loss = 0.36737455
Iteration 19, loss = 0.36586517
Iteration 20, loss = 0.36817489
Iteration 21, loss = 0.36105690
Iteration 22, loss = 0.36294541
Iteration 23, loss = 0.36231090
Iteration 24, loss = 0.36228340
Iteration 25, loss = 0.35980453
Iteration 26, loss = 0.36310538
Iteration 27, loss = 0.35816197
Iteration 28, loss = 0.36075982
Iteration 29, loss = 0.35646944
Iteration 30, loss = 0.35834431
Iteration 31, loss = 0.35553181
Iteration 32, loss = 0.35365784
Iteration 33, loss = 0.35276967
Iteration 34, loss = 0.34861417
Iteration 35, loss = 0.35007707
Iteration 36, loss = 0.34758275
Iteration 37, loss = 0.34540911
Iteration 38, loss = 0.34409695
Iteration 39, loss = 0.34787671
Iteration 40, loss = 0.34295680
Iteration 41, loss = 0.34363968
Iteration 42, loss = 0.34664926
Iteration 43, loss = 0.34101252
Iteration 44, loss = 0.33852706
Iteration 45, loss = 0.33912465
Iteration 46, loss = 0.33582049
Iteration 47, loss = 0.33758530
Iteration 48, loss = 0.33671031
Iteration 49, loss = 0.33815909
Iteration 50, loss = 0.33780188
Iteration 51, loss = 0.33146258
Iteration 52, loss = 0.33209407
```

```
Iteration 53, loss = 0.33407682
Iteration 54, loss = 0.33116972
Iteration 55, loss = 0.33061369
Iteration 56, loss = 0.33040979
Iteration 57, loss = 0.32895087
Iteration 58, loss = 0.32902791
Iteration 59, loss = 0.32543022
Iteration 60, loss = 0.32728978
Iteration 61, loss = 0.32422190
Iteration 62, loss = 0.32360453
Iteration 63, loss = 0.32238916
Iteration 64, loss = 0.32277891
Iteration 65, loss = 0.32290921
Iteration 66, loss = 0.32122655
Iteration 67, loss = 0.32056813
Iteration 68, loss = 0.32056449
Iteration 69, loss = 0.31906633
Iteration 70, loss = 0.31790411
Iteration 71, loss = 0.31441585
Iteration 72, loss = 0.31684911
Iteration 73, loss = 0.31498432
Iteration 74, loss = 0.30900472
Iteration 75, loss = 0.31432446
Iteration 76, loss = 0.31056439
Iteration 77, loss = 0.30805943
Iteration 78, loss = 0.30742271
Iteration 79, loss = 0.30693733
Iteration 80, loss = 0.30549452
Iteration 81, loss = 0.30378364
Iteration 82, loss = 0.30639097
Iteration 83, loss = 0.30422040
Iteration 84, loss = 0.30407961
Iteration 85, loss = 0.30261573
Iteration 86, loss = 0.30643371
Iteration 87, loss = 0.30239099
Iteration 88, loss = 0.30356498
Iteration 89, loss = 0.30035482
Iteration 90, loss = 0.29867554
Iteration 91, loss = 0.29091451
Iteration 92, loss = 0.29413738
Iteration 93, loss = 0.29016626
Iteration 94, loss = 0.29298879
Iteration 95, loss = 0.28833958
Iteration 96, loss = 0.28922477
Iteration 97, loss = 0.28843797
Iteration 98, loss = 0.28415135
Iteration 99, loss = 0.28295767
Iteration 100, loss = 0.28636073
```

```
Iteration 101, loss = 0.28529655
Iteration 102, loss = 0.28484386
Iteration 103, loss = 0.28778513
Iteration 104, loss = 0.28639778
Iteration 105, loss = 0.28944276
Iteration 106, loss = 0.27964485
Iteration 107, loss = 0.27654187
Iteration 108, loss = 0.27507383
Iteration 109, loss = 0.27712496
Iteration 110, loss = 0.28107921
Iteration 111, loss = 0.27466398
Iteration 112, loss = 0.27693831
Iteration 113, loss = 0.27410315
Iteration 114, loss = 0.27677804
Iteration 115, loss = 0.27486019
Iteration 116, loss = 0.27042526
Iteration 117, loss = 0.26500951
Iteration 118, loss = 0.26804243
Iteration 119, loss = 0.26869540
Iteration 120, loss = 0.26700561
Iteration 121, loss = 0.27737928
Iteration 122, loss = 0.28036904
Iteration 123, loss = 0.26367532
Iteration 124, loss = 0.25645073
Iteration 125, loss = 0.25751271
Iteration 126, loss = 0.25643686
Iteration 127, loss = 0.26203836
Iteration 128, loss = 0.25729368
Iteration 129, loss = 0.25465518
Iteration 130, loss = 0.25463176
Iteration 131, loss = 0.24980065
Iteration 132, loss = 0.25790683
Iteration 133, loss = 0.25451887
Iteration 134, loss = 0.25439563
Iteration 135, loss = 0.26167920
Iteration 136, loss = 0.24501375
Iteration 137, loss = 0.24647365
Iteration 138, loss = 0.24919614
Iteration 139, loss = 0.24866300
Iteration 140, loss = 0.24857594
Iteration 141, loss = 0.24308240
Iteration 142, loss = 0.24906784
Iteration 143, loss = 0.25349703
Iteration 144, loss = 0.23904336
Iteration 145, loss = 0.23566923
Iteration 146, loss = 0.23849964
Iteration 147, loss = 0.23841532
Iteration 148, loss = 0.23528898
```

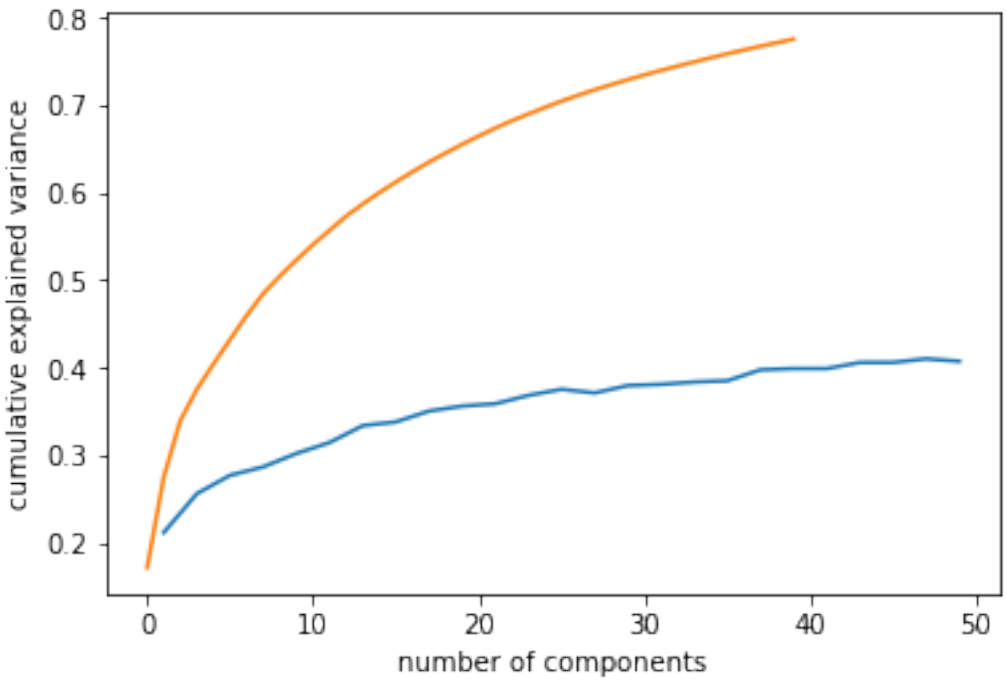
```
Iteration 149, loss = 0.23795650
Iteration 150, loss = 0.23193605
Iteration 151, loss = 0.23976770
Iteration 152, loss = 0.23539441
Iteration 153, loss = 0.23226365
Iteration 154, loss = 0.23532630
Iteration 155, loss = 0.23583875
Iteration 156, loss = 0.23165771
Iteration 157, loss = 0.23379542
Iteration 158, loss = 0.24114075
Iteration 159, loss = 0.23259340
Iteration 160, loss = 0.22985503
Iteration 161, loss = 0.22464825
Iteration 162, loss = 0.22919556
Iteration 163, loss = 0.22282466
Iteration 164, loss = 0.22195836
Iteration 165, loss = 0.21851188
Iteration 166, loss = 0.22287239
Iteration 167, loss = 0.21680071
Iteration 168, loss = 0.21423155
Iteration 169, loss = 0.21328883
Iteration 170, loss = 0.22615580
Iteration 171, loss = 0.23235854
Iteration 172, loss = 0.22439076
Iteration 173, loss = 0.21681674
Iteration 174, loss = 0.22421320
Iteration 175, loss = 0.23365969
Iteration 176, loss = 0.22693083
Iteration 177, loss = 0.21846587
Iteration 178, loss = 0.22221135
Iteration 179, loss = 0.20050237
Iteration 180, loss = 0.21029779
Iteration 181, loss = 0.20398629
Iteration 182, loss = 0.20283749
Iteration 183, loss = 0.20201007
Iteration 184, loss = 0.20351055
Iteration 185, loss = 0.19781600
Iteration 186, loss = 0.19598051
Iteration 187, loss = 0.20304512
Iteration 188, loss = 0.19678987
Iteration 189, loss = 0.19284968
Iteration 190, loss = 0.19776338
Iteration 191, loss = 0.19870361
Iteration 192, loss = 0.19390987
Iteration 193, loss = 0.19224593
Iteration 194, loss = 0.18817813
Iteration 195, loss = 0.18850765
Iteration 196, loss = 0.18721186
```

```
Iteration 197, loss = 0.18403646
Iteration 198, loss = 0.18857716
Iteration 199, loss = 0.18852866
Iteration 200, loss = 0.18604884


C:\Users\GaryNg\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:562
  % self.max_iter, ConvergenceWarning)


MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 0.9211111111111111
testing accuracy: 0.7333333333333333
confusion matrix:
 [[61  6  2 15]
 [ 7 47 12  5]
 [ 1 10 59  7]
 [ 6  5  4 53]]
              precision    recall  f1-score   support

           0       0.81      0.73      0.77        84
           1       0.69      0.66      0.68        71
           2       0.77      0.77      0.77        77
           3       0.66      0.78      0.72        68

   micro avg       0.73      0.73      0.73       300
   macro avg       0.73      0.73      0.73       300
weighted avg       0.74      0.73      0.73       300



In [146]: def runAll(testSize = 0.25):
              data = train_test_split(*unpickleAugmentedData(), random_state=123, test_size=tes

              print("--- Model 1 ---")
              trainKnn(*data)
              print("--- Model 2 ---")
              trainKnn(*fitPca(*data, components=40))
              print("--- Model 3 ---")
              trainKnn(*fitLda(*data))
              print("--- Model 4 ---")
              trainKnn(*fitLda(*fitPca(*data, components=45)))
```

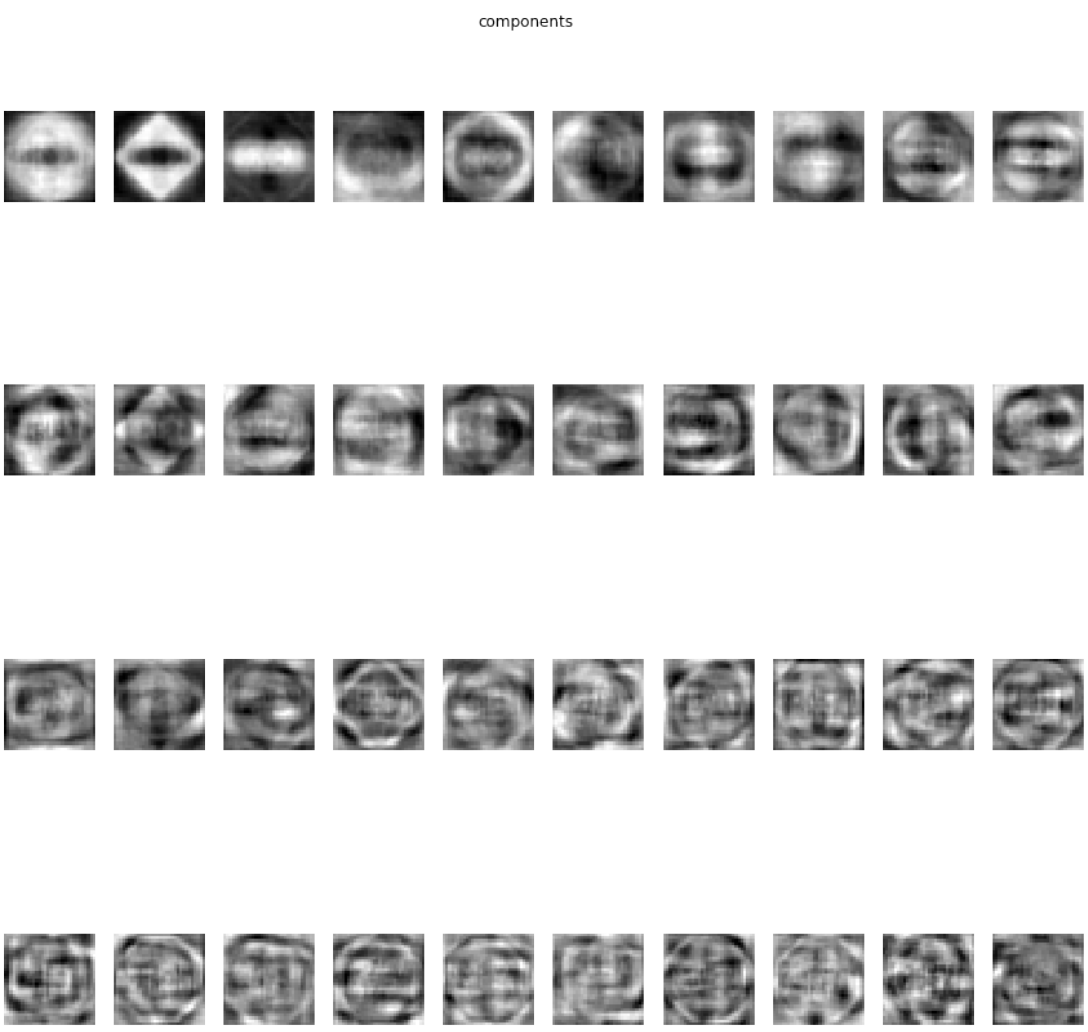
```
    print("--- Model 5 ---")
    trainSvm(*fitPca(*data, components=128))
    print("--- Model 6 ---")
    trainSvm(*fitLda(*fitPca(*data, components=90)))
    print("--- Model 7 ---")
    trainMlp(*data)
    print("--- Model 8 ---")
    trainMlp(*fitPca(*data, components=128))
    print("--- Model 9 ---")
    trainMlp(*fitLda(*data))
    print("--- Model 10 ---")
    trainMlp(*fitLda(*fitPca(*data, components=128)))
```

```
In [147]: runAll(testSize=0.4)
```

```
--- Model 1 ---
trainingData: (720, 1024), trainingLabels: (720,)
testingData: (480, 1024), testingLabels: (480,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.8041666666666667
confusion matrix:
 [[125   1   2   3]
 [ 31  79   0   3]
 [ 11   3 101   5]
 [ 33   0   2  81]]
              precision    recall  f1-score   support

           0       0.62      0.95      0.76       131
           1       0.95      0.70      0.81       113
           2       0.96      0.84      0.90       120
           3       0.88      0.70      0.78       116

   micro avg       0.80      0.80      0.80       480
   macro avg       0.85      0.80      0.81       480
weighted avg       0.85      0.80      0.81       480

--- Model 2 ---
trainingData: (720, 1024), trainingLabels: (720,)
testingData: (480, 1024), testingLabels: (480,)
total explained variance: 0.7760278631785977
```

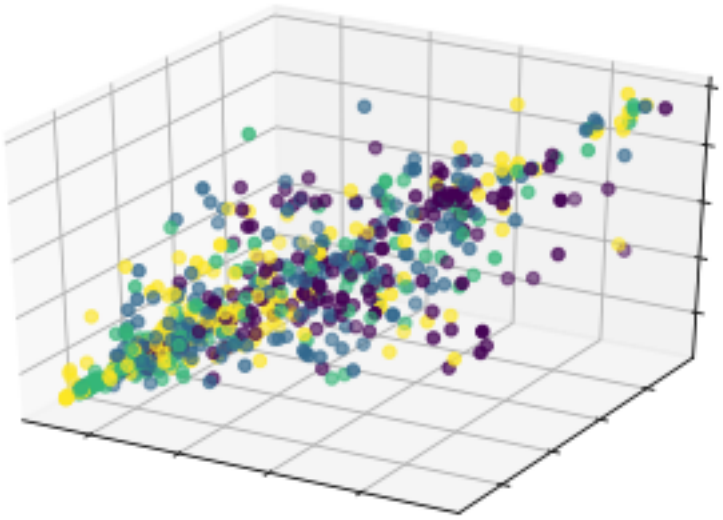


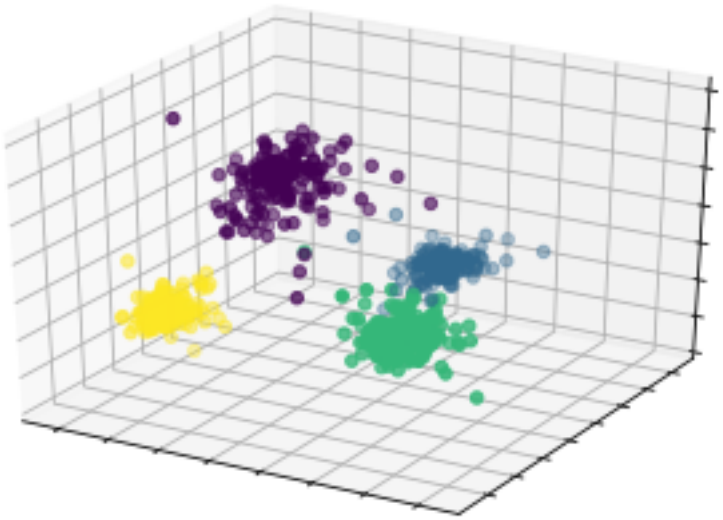

```
pca components: (40, 1024)
```



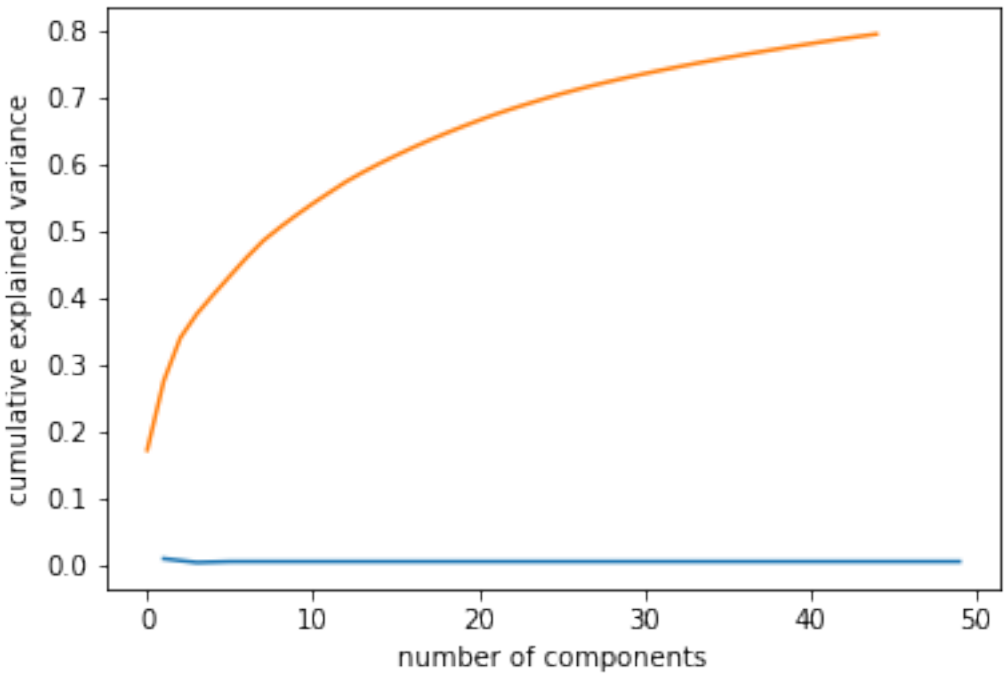

```
trainingData: (720, 40), trainingLabels: (720,)
testingData: (480, 40), testingLabels: (480,)
optimal k: 1
training accuracy: 1.0
testing accuracy: 0.8395833333333333
confusion matrix:
 [[116   3   2  10]
 [ 22  79   6   6]
 [  9   4 101   6]
 [  6   0   3 107]]
              precision    recall  f1-score   support

           0       0.76      0.89      0.82       131
           1       0.92      0.70      0.79       113
           2       0.90      0.84      0.87       120
```

```
           3       0.83      0.92      0.87       116

   micro avg       0.84      0.84      0.84       480
   macro avg       0.85      0.84      0.84       480
weighted avg       0.85      0.84      0.84       480

--- Model 3 ---
trainingData: (720, 1024), trainingLabels: (720,)
testingData: (480, 1024), testingLabels: (480,)


C:\Users\GaryNg\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning:
  warnings.warn("Variables are collinear.")
C:\Users\GaryNg\Anaconda3\lib\site-packages\matplotlib\figure.py:98: MatplotlibDeprecationWarni
Adding an axes using the same arguments as a previous axes currently reuses the earlier instanc
  "Adding an axes using the same arguments as a previous axes "
```

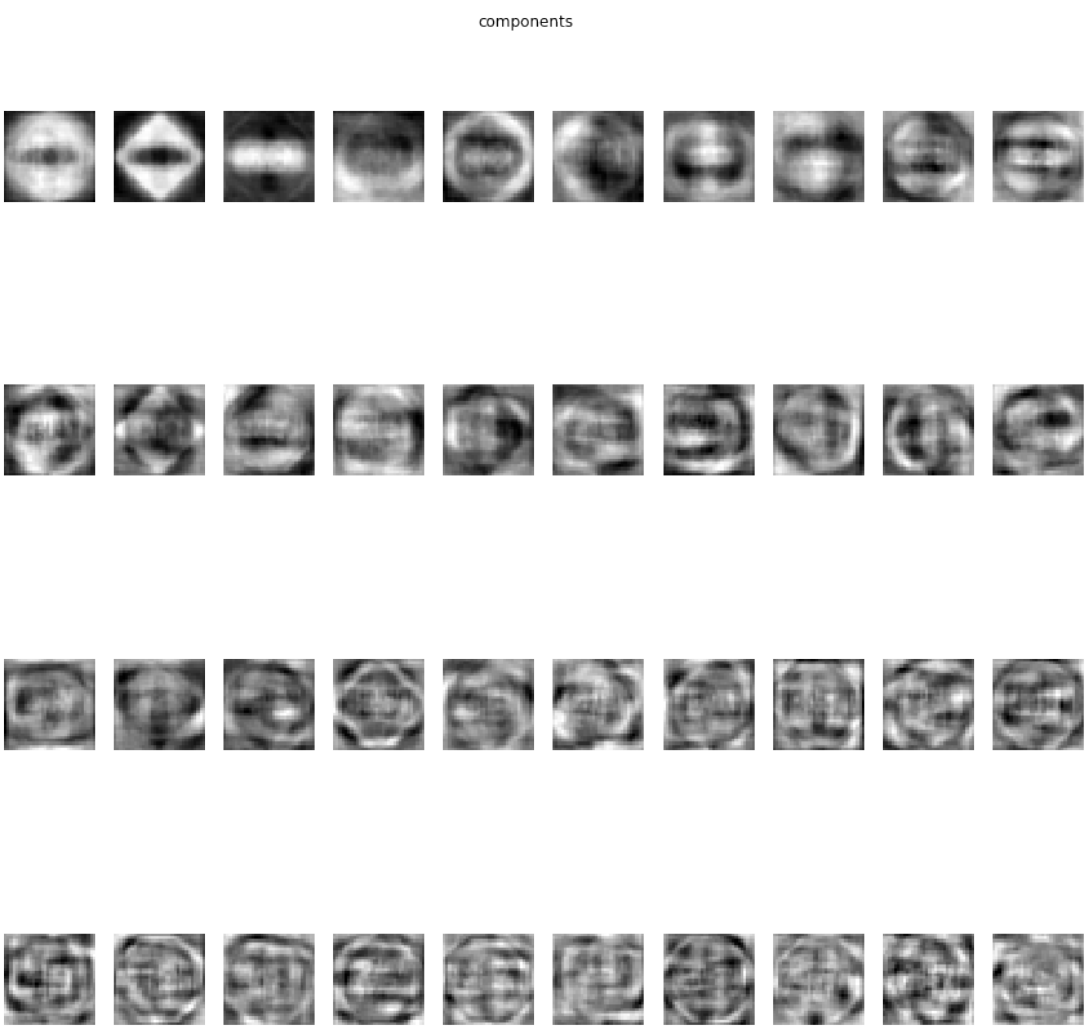




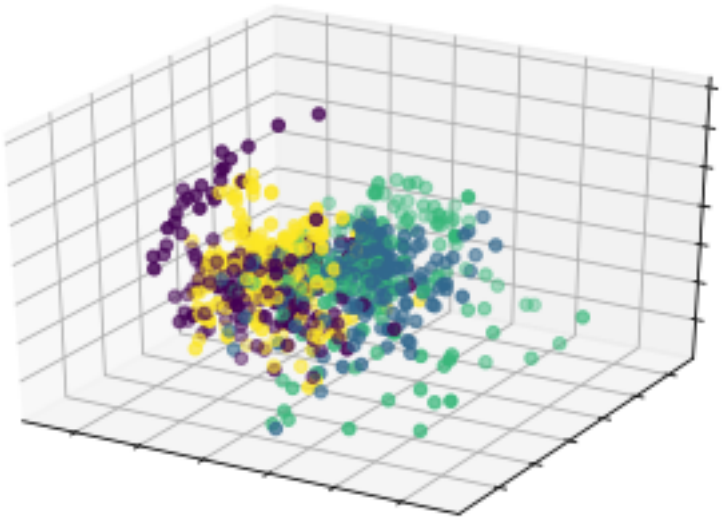
```
trainingData: (720, 3), trainingLabels: (720,)
testingData: (480, 3), testingLabels: (480,)
optimal k: 3
training accuracy: 0.9972222222222222
testing accuracy: 0.5354166666666667
confusion matrix:
 [[92 20 11  8]
 [29 47 21 16]
 [20 12 78 10]
 [36 16 24 40]]
              precision    recall  f1-score   support

           0       0.52      0.70      0.60       131
           1       0.49      0.42      0.45       113
           2       0.58      0.65      0.61       120
           3       0.54      0.34      0.42       116

   micro avg       0.54      0.54      0.54       480
   macro avg       0.53      0.53      0.52       480
weighted avg       0.53      0.54      0.52       480

--- Model 4 ---
trainingData: (720, 1024), trainingLabels: (720,)
testingData: (480, 1024), testingLabels: (480,)
total explained variance: 0.7943797720610103
```

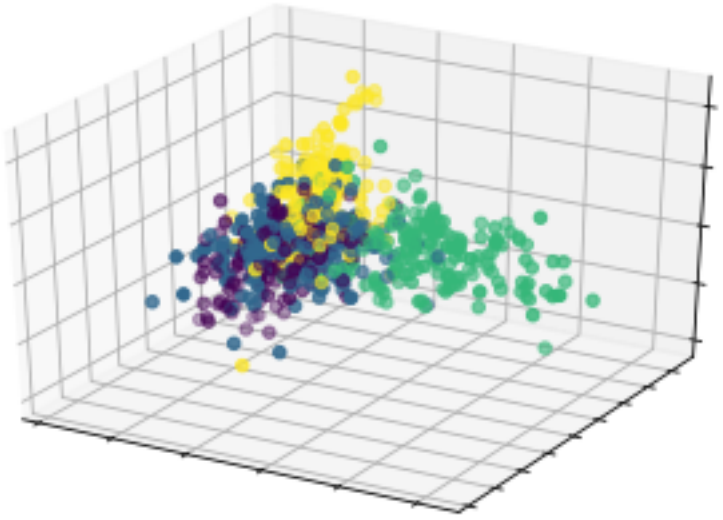



```
pca components: (45, 1024)
```



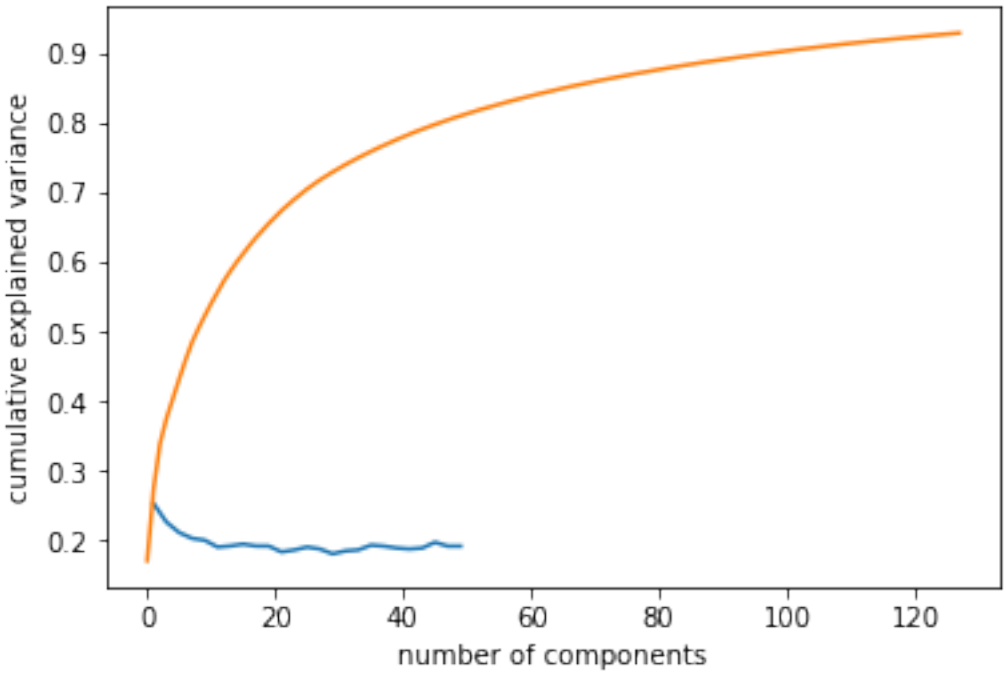

```
trainingData: (720, 45), trainingLabels: (720,)
testingData: (480, 45), testingLabels: (480,)
```



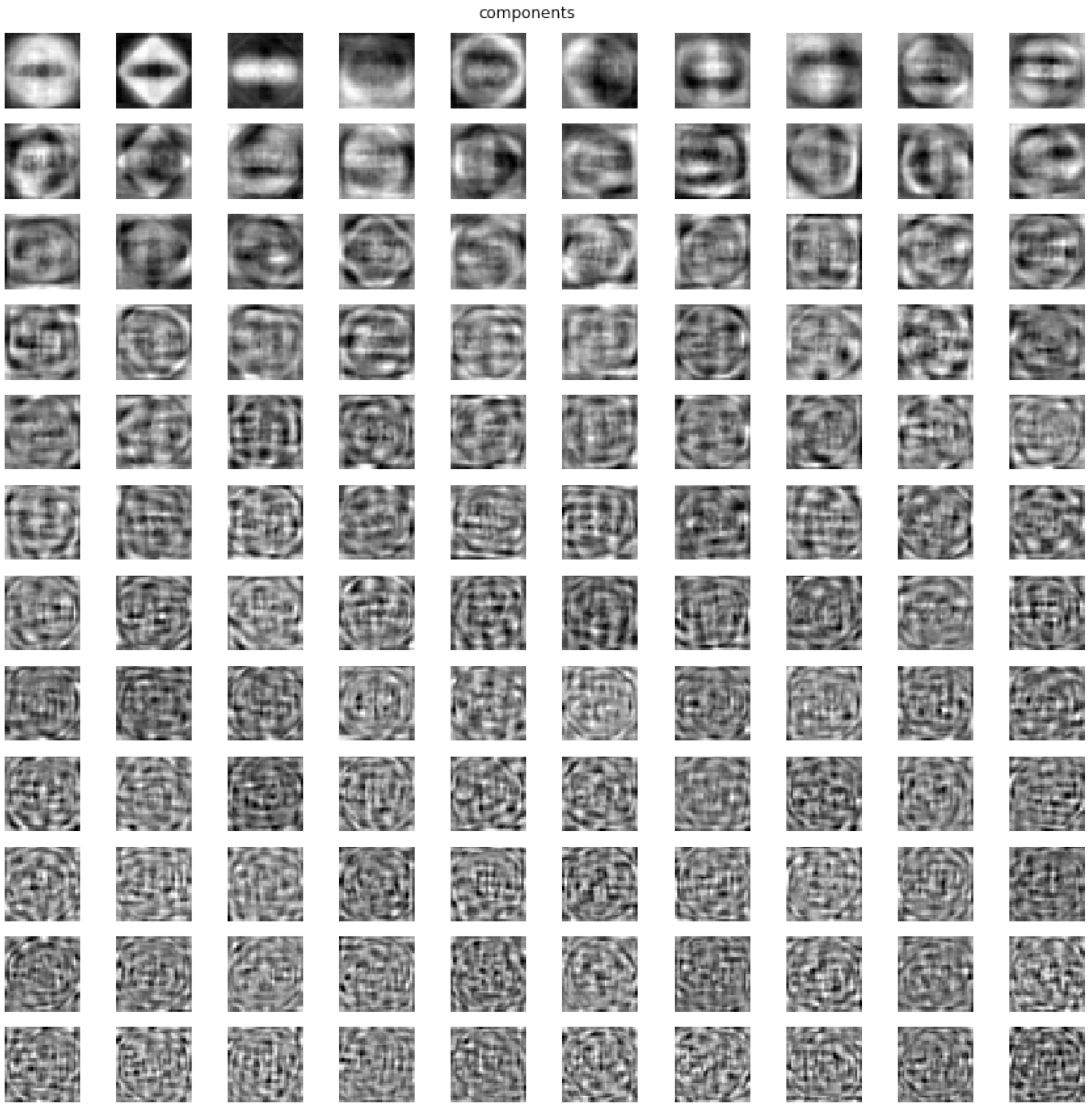


```
trainingData: (720, 3), trainingLabels: (720,)
testingData: (480, 3), testingLabels: (480,)
optimal k: 29
training accuracy: 0.825
testing accuracy: 0.7625
confusion matrix:
```

```
 [[ 95  13   2  21]
 [ 14  88   8   3]
 [  2  14 100   4]
 [ 17  13   3  83]]
              precision    recall  f1-score   support

           0       0.74      0.73      0.73       131
           1       0.69      0.78      0.73       113
           2       0.88      0.83      0.86       120
           3       0.75      0.72      0.73       116

   micro avg       0.76      0.76      0.76       480
   macro avg       0.77      0.76      0.76       480
weighted avg       0.77      0.76      0.76       480

--- Model 5 ---
trainingData: (720, 1024), trainingLabels: (720,)
testingData: (480, 1024), testingLabels: (480,)
total explained variance: 0.9290803675188927
```

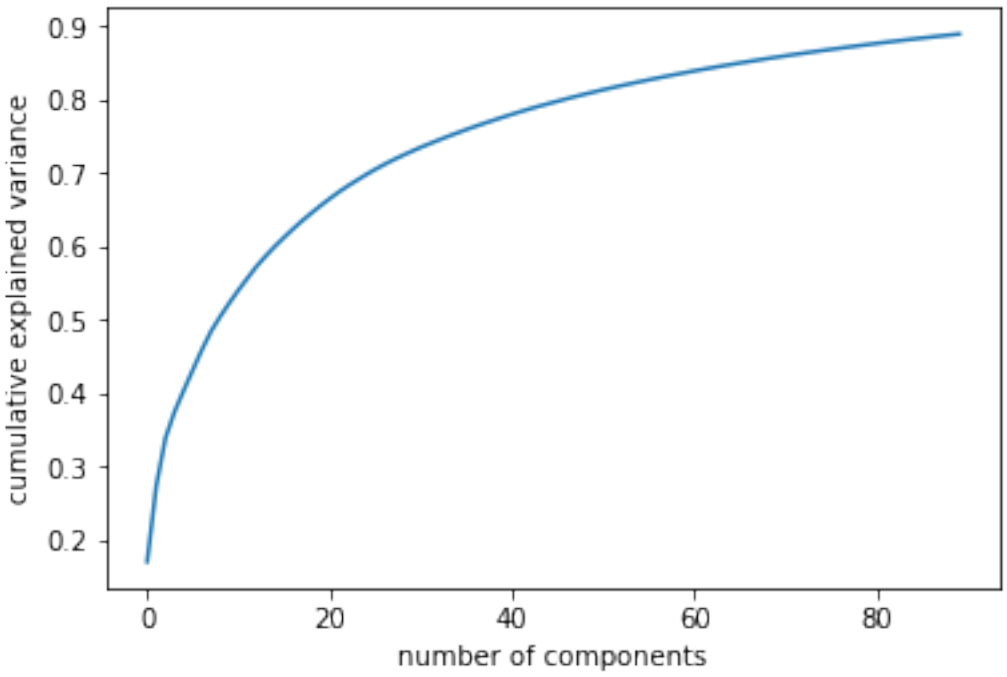



```
pca components: (128, 1024)
```

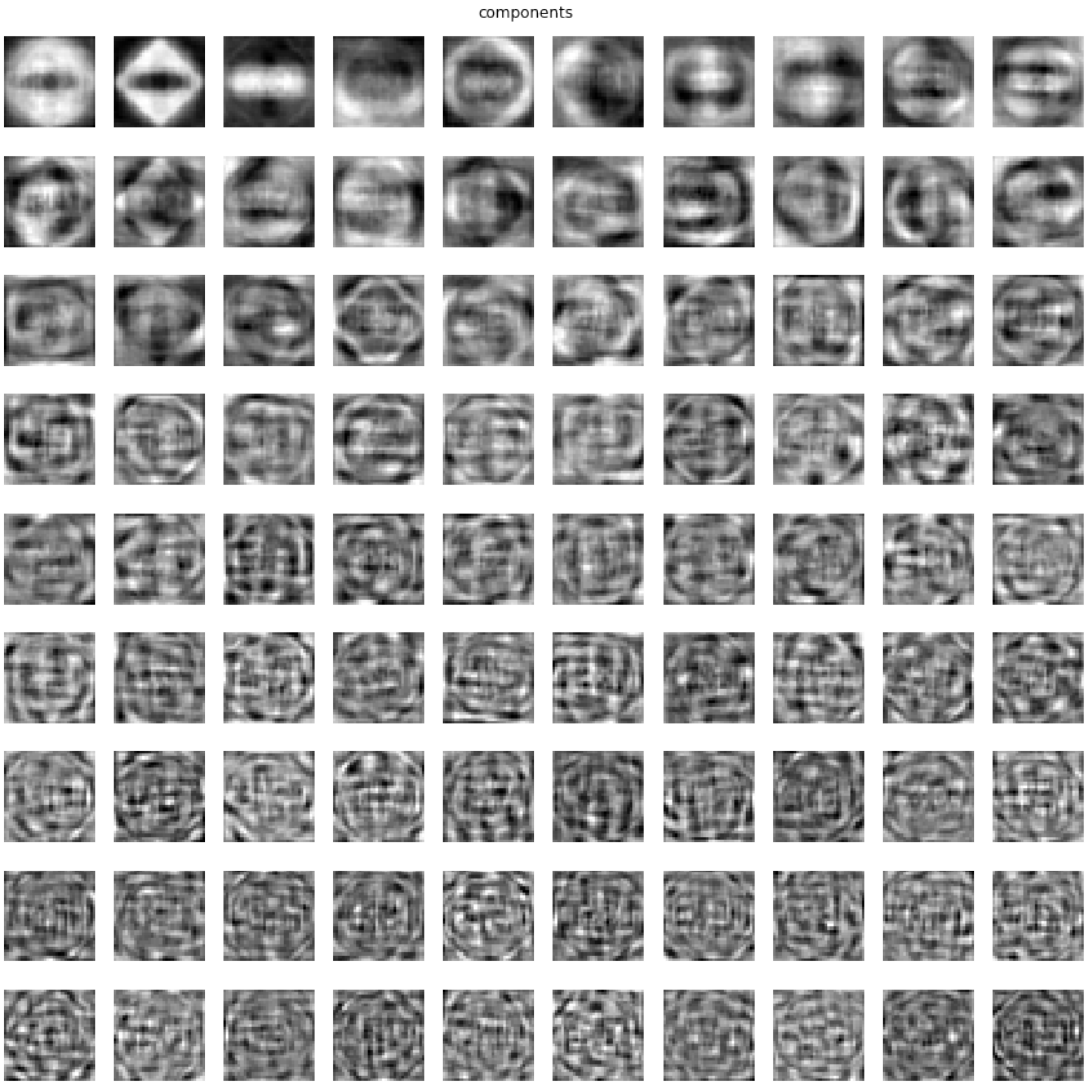



```
trainingData: (720, 128), trainingLabels: (720,)
testingData: (480, 128), testingLabels: (480,)
training accuracy: 0.9958333333333333
testing accuracy: 0.70625
confusion matrix:
 [[99  9  3 20]
 [22 74 13  4]
 [ 6 15 93  6]
 [24 14  5 73]]
              precision    recall  f1-score   support

           0       0.66      0.76      0.70       131
           1       0.66      0.65      0.66       113
```

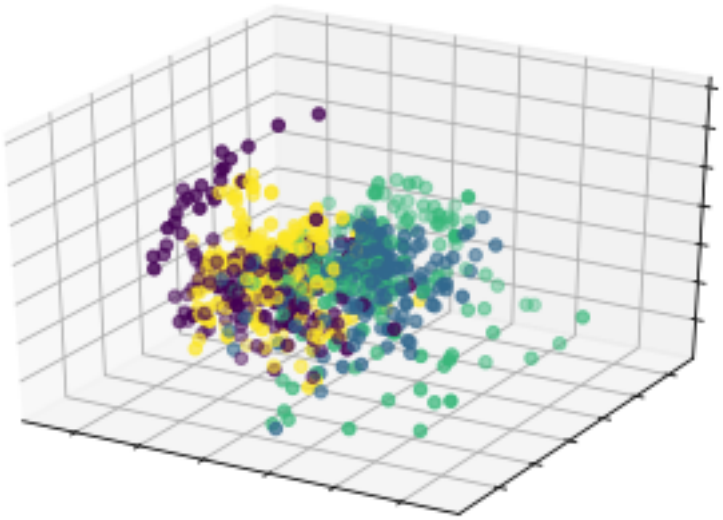
```
           2       0.82      0.78      0.79       120
           3       0.71      0.63      0.67       116

   micro avg       0.71      0.71      0.71       480
   macro avg       0.71      0.70      0.71       480
weighted avg       0.71      0.71      0.71       480

--- Model 6 ---
trainingData: (720, 1024), trainingLabels: (720,)
testingData: (480, 1024), testingLabels: (480,)
total explained variance: 0.8890576383549031
```

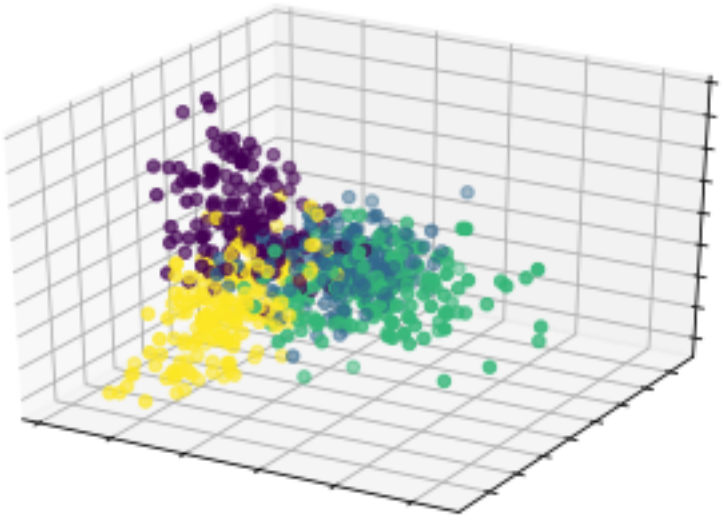



```
pca components: (90, 1024)
```




```
trainingData: (720, 90), trainingLabels: (720,)
testingData: (480, 90), testingLabels: (480,)
```

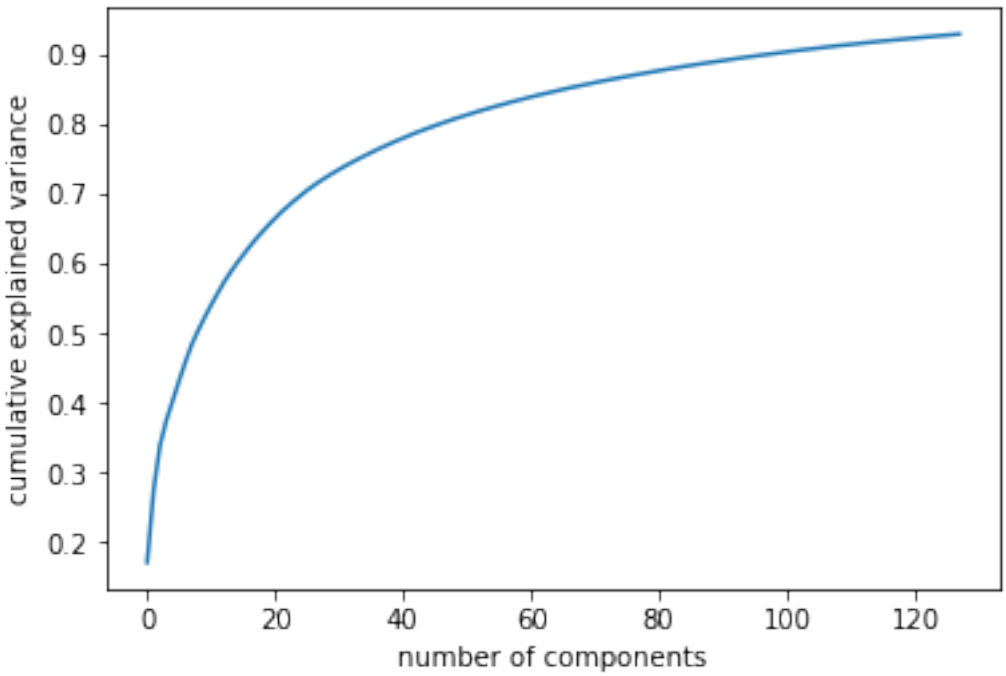




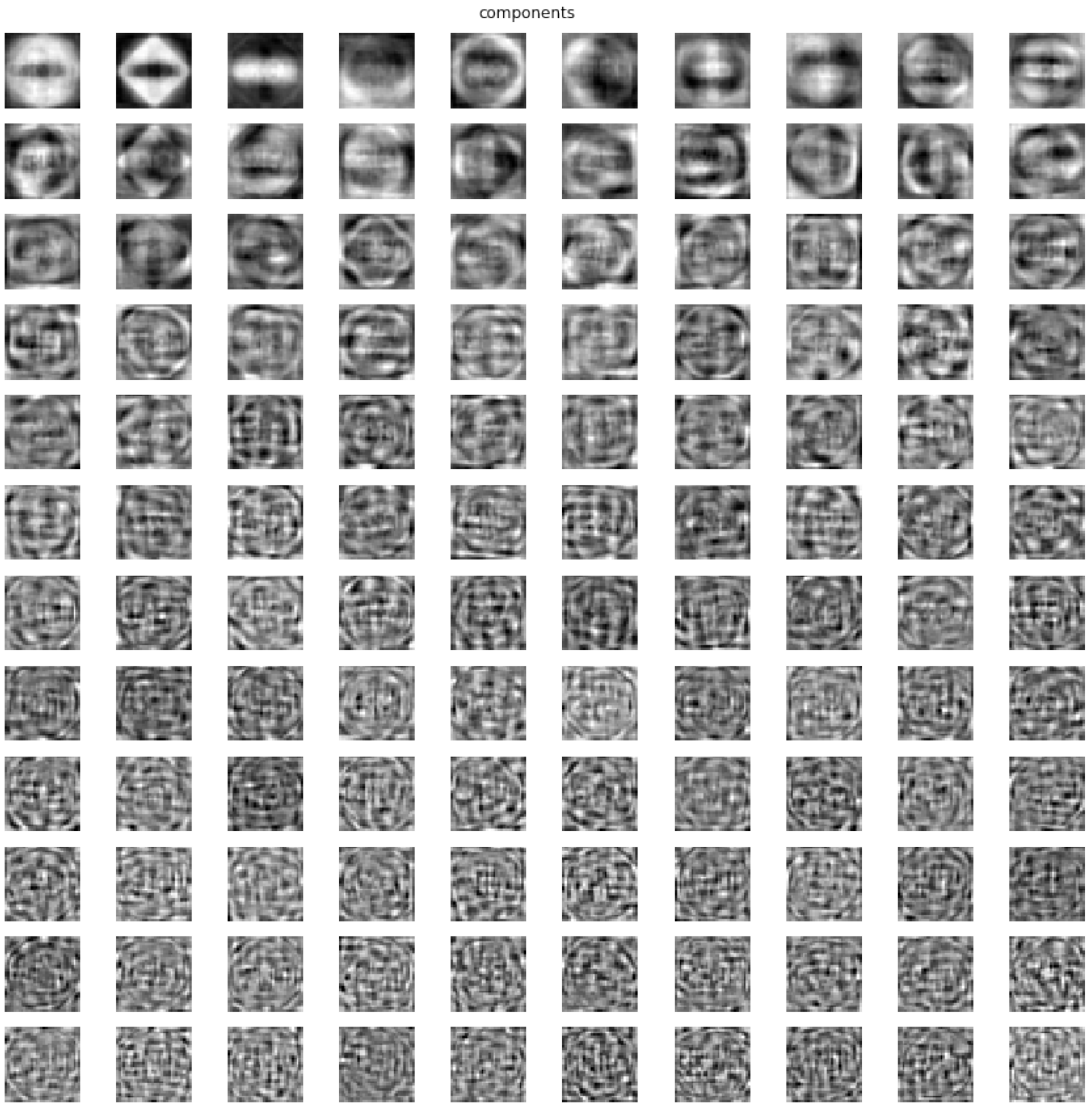
```
trainingData: (720, 3), trainingLabels: (720,)
testingData: (480, 3), testingLabels: (480,)
training accuracy: 0.8416666666666667
testing accuracy: 0.7520833333333333
confusion matrix:
 [[ 91  10   5  25]
```

```
 [  8  84  15   6]
 [  5   8 103   4]
 [ 14  13   6  83]]
              precision    recall  f1-score   support

           0       0.77      0.69      0.73       131
           1       0.73      0.74      0.74       113
           2       0.80      0.86      0.83       120
           3       0.70      0.72      0.71       116

   micro avg       0.75      0.75      0.75       480
   macro avg       0.75      0.75      0.75       480
weighted avg       0.75      0.75      0.75       480

--- Model 7 ---
trainingData: (720, 1024), trainingLabels: (720,)
testingData: (480, 1024), testingLabels: (480,)
Iteration 1, loss = 1.47602339
Iteration 2, loss = 0.58621561
Iteration 3, loss = 0.26229212
Iteration 4, loss = 0.20554925
Iteration 5, loss = 0.10561282
Iteration 6, loss = 0.05289469
Iteration 7, loss = 0.03997310
Iteration 8, loss = 0.02163930
Iteration 9, loss = 0.01229867
Iteration 10, loss = 0.00867576
Iteration 11, loss = 0.00620063
Iteration 12, loss = 0.00442928
Iteration 13, loss = 0.00348352
Iteration 14, loss = 0.00293659
Iteration 15, loss = 0.00243920
Iteration 16, loss = 0.00211654
Iteration 17, loss = 0.00191200
Iteration 18, loss = 0.00174852
Iteration 19, loss = 0.00163139
Iteration 20, loss = 0.00154254
Iteration 21, loss = 0.00146105
Iteration 22, loss = 0.00139626
Iteration 23, loss = 0.00133949
Iteration 24, loss = 0.00129016
Iteration 25, loss = 0.00124857
Iteration 26, loss = 0.00121335
Iteration 27, loss = 0.00117798
Iteration 28, loss = 0.00114825
Iteration 29, loss = 0.00112051
Iteration 30, loss = 0.00109442
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.8479166666666667
confusion matrix:
 [[107  10   3  11]
 [  5  96   7   5]
 [  3   5 108   4]
 [ 10   8   2  96]]
              precision    recall  f1-score   support

           0       0.86      0.82      0.84       131
           1       0.81      0.85      0.83       113
           2       0.90      0.90      0.90       120
           3       0.83      0.83      0.83       116

   micro avg       0.85      0.85      0.85       480
   macro avg       0.85      0.85      0.85       480
weighted avg       0.85      0.85      0.85       480

--- Model 8 ---
trainingData: (720, 1024), trainingLabels: (720,)
testingData: (480, 1024), testingLabels: (480,)
total explained variance: 0.9290525437346294
```



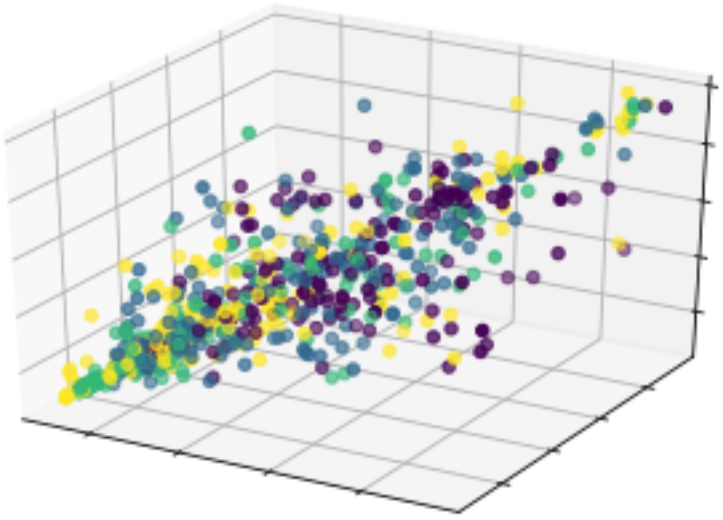

```
pca components: (128, 1024)
```

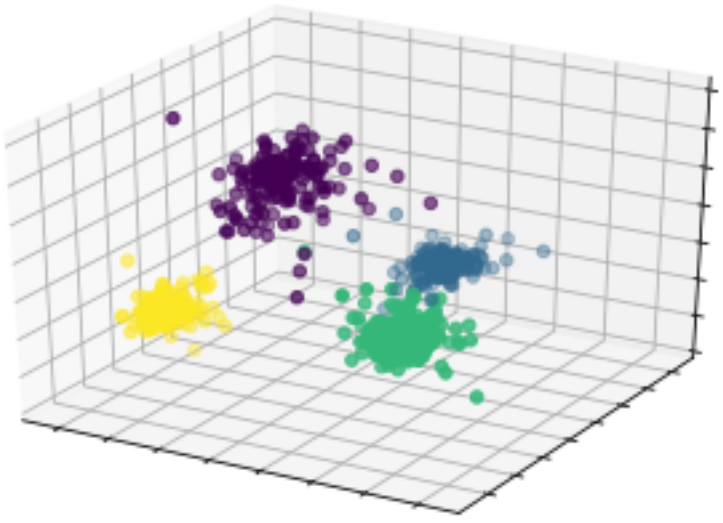



```
trainingData: (720, 128), trainingLabels: (720,)
testingData: (480, 128), testingLabels: (480,)
Iteration 1, loss = 1.54625542
Iteration 2, loss = 0.89981013
Iteration 3, loss = 0.58054572
Iteration 4, loss = 0.33056398
Iteration 5, loss = 0.19760581
Iteration 6, loss = 0.12300960
Iteration 7, loss = 0.06351247
Iteration 8, loss = 0.03919718
Iteration 9, loss = 0.02191894
Iteration 10, loss = 0.01297830
Iteration 11, loss = 0.00947660
```

```
Iteration 12, loss = 0.00688095
Iteration 13, loss = 0.00501165
Iteration 14, loss = 0.00390247
Iteration 15, loss = 0.00328297
Iteration 16, loss = 0.00284220
Iteration 17, loss = 0.00252495
Iteration 18, loss = 0.00227825
Iteration 19, loss = 0.00207558
Iteration 20, loss = 0.00191991
Iteration 21, loss = 0.00179986
Iteration 22, loss = 0.00169454
Iteration 23, loss = 0.00160660
Iteration 24, loss = 0.00153273
Iteration 25, loss = 0.00146630
Iteration 26, loss = 0.00140562
Iteration 27, loss = 0.00135137
Iteration 28, loss = 0.00130308
Iteration 29, loss = 0.00125775
Iteration 30, loss = 0.00121800
Iteration 31, loss = 0.00117954
Iteration 32, loss = 0.00114320
Iteration 33, loss = 0.00110931
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.775
confusion matrix:
 [[105   8   6  12]
 [ 13  82  14   4]
 [  3   7 104   6]
 [ 21   8   6  81]]
              precision    recall  f1-score   support

           0       0.74      0.80      0.77       131
           1       0.78      0.73      0.75       113
           2       0.80      0.87      0.83       120
           3       0.79      0.70      0.74       116

   micro avg       0.78      0.78      0.78       480
   macro avg       0.78      0.77      0.77       480
weighted avg       0.78      0.78      0.77       480
```

```
--- Model 9 ---
trainingData: (720, 1024), trainingLabels: (720,)
testingData: (480, 1024), testingLabels: (480,)


C:\Users\GaryNg\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning:
  warnings.warn("Variables are collinear.")
```

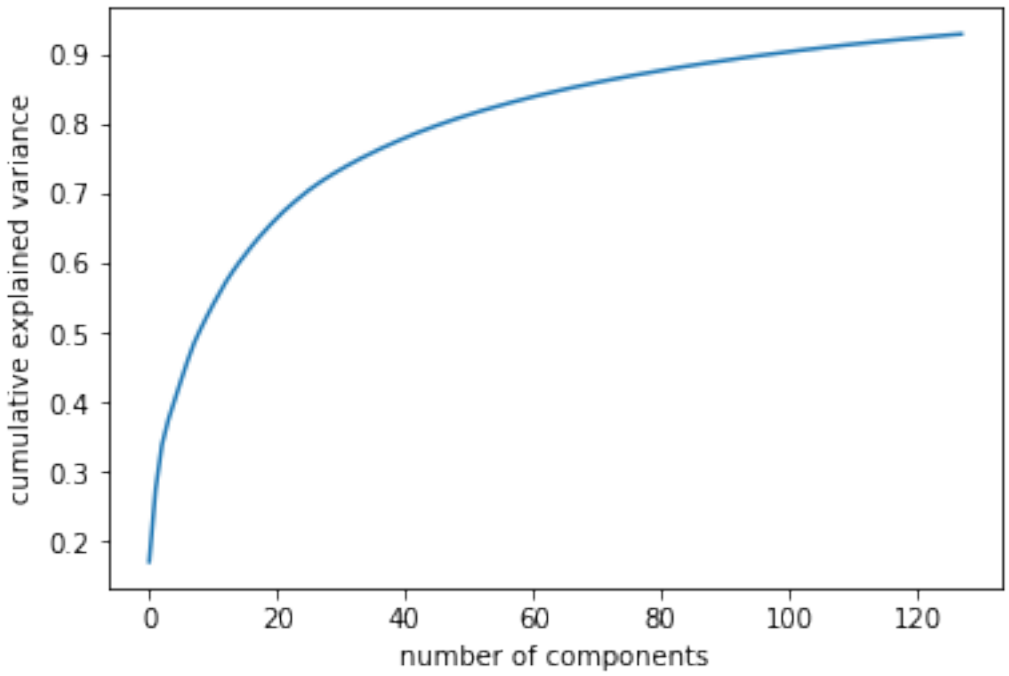




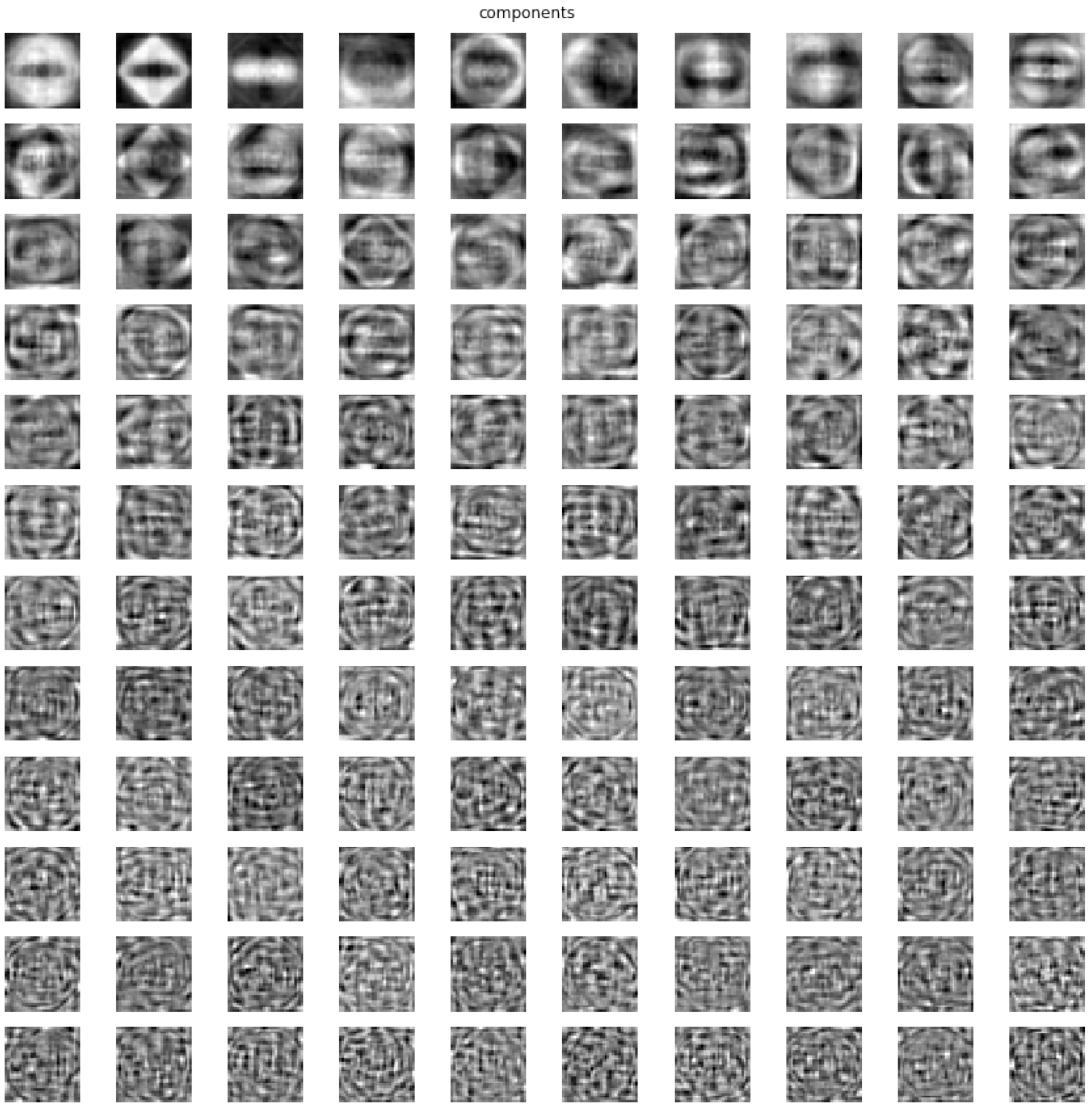
```
trainingData: (720, 3), trainingLabels: (720,)
testingData: (480, 3), testingLabels: (480,)
Iteration 1, loss = 0.90917457
Iteration 2, loss = 0.16777627
Iteration 3, loss = 0.03337685
Iteration 4, loss = 0.01637683
Iteration 5, loss = 0.01361828
Iteration 6, loss = 0.01278697
Iteration 7, loss = 0.01249617
Iteration 8, loss = 0.01196479
Iteration 9, loss = 0.01130730
Iteration 10, loss = 0.01043138
Iteration 11, loss = 0.01002519
Iteration 12, loss = 0.00953023
Iteration 13, loss = 0.00894275
Iteration 14, loss = 0.00853150
Iteration 15, loss = 0.00811581
Iteration 16, loss = 0.00759440
Iteration 17, loss = 0.00733777
Iteration 18, loss = 0.00703457
Iteration 19, loss = 0.00675293
Iteration 20, loss = 0.00650508
Iteration 21, loss = 0.00633842
Iteration 22, loss = 0.00604022
Iteration 23, loss = 0.00582080
Iteration 24, loss = 0.00578480
Iteration 25, loss = 0.00556815
Iteration 26, loss = 0.00535473
Iteration 27, loss = 0.00520991
Iteration 28, loss = 0.00494877
Iteration 29, loss = 0.00495714
Iteration 30, loss = 0.00467602
Iteration 31, loss = 0.00461013
Iteration 32, loss = 0.00441782
Iteration 33, loss = 0.00439038
Iteration 34, loss = 0.00425128
Iteration 35, loss = 0.00404601
Iteration 36, loss = 0.00407108
Iteration 37, loss = 0.00402154
Iteration 38, loss = 0.00390147
Iteration 39, loss = 0.00374863
Iteration 40, loss = 0.00352183
Iteration 41, loss = 0.00347745
Iteration 42, loss = 0.00346833
Iteration 43, loss = 0.00362472
Iteration 44, loss = 0.00337963
Iteration 45, loss = 0.00330316
Iteration 46, loss = 0.00313612
```

```
Iteration 47, loss = 0.00326525
Iteration 48, loss = 0.00313615
Iteration 49, loss = 0.00308562
Iteration 50, loss = 0.00299211
Iteration 51, loss = 0.00302721
Iteration 52, loss = 0.00285450
Iteration 53, loss = 0.00315025
Iteration 54, loss = 0.00282295
Iteration 55, loss = 0.00271659
Iteration 56, loss = 0.00280591
Iteration 57, loss = 0.00295658
Iteration 58, loss = 0.00301325
Iteration 59, loss = 0.00274861
Iteration 60, loss = 0.00271049
Iteration 61, loss = 0.00262437
Iteration 62, loss = 0.00280718
Iteration 63, loss = 0.00277765
Iteration 64, loss = 0.00259984
Iteration 65, loss = 0.00259151
Iteration 66, loss = 0.00245723
Iteration 67, loss = 0.00268928
Iteration 68, loss = 0.00245845
Iteration 69, loss = 0.00275468
Iteration 70, loss = 0.00267187
Iteration 71, loss = 0.00248967
Iteration 72, loss = 0.00285149
Iteration 73, loss = 0.00239020
Iteration 74, loss = 0.00236550
Iteration 75, loss = 0.00254026
Iteration 76, loss = 0.00269264
Iteration 77, loss = 0.00259480
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 1.0
testing accuracy: 0.50625
confusion matrix:
 [[74 34 13 10]
 [28 54 15 16]
 [20 14 72 14]
 [39 18 16 43]]
              precision    recall  f1-score   support
```

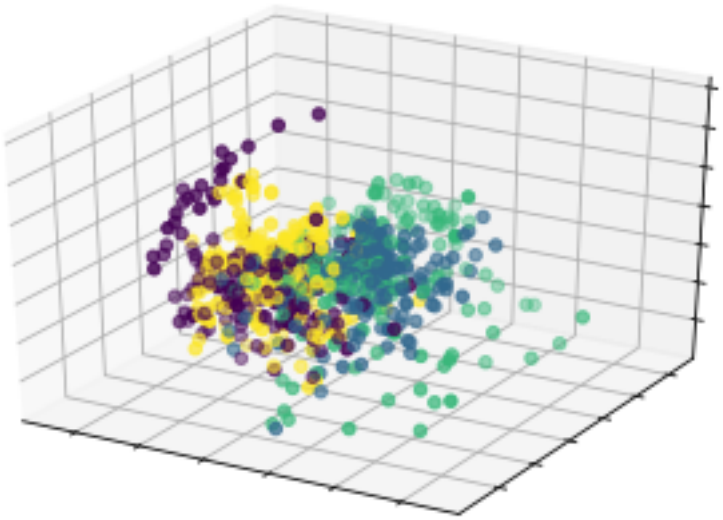
```
              0       0.46      0.56      0.51       131
              1       0.45      0.48      0.46       113
              2       0.62      0.60      0.61       120
              3       0.52      0.37      0.43       116

   micro avg       0.51      0.51      0.51       480
   macro avg       0.51      0.50      0.50       480
weighted avg       0.51      0.51      0.50       480

--- Model 10 ---
trainingData: (720, 1024), trainingLabels: (720,)
testingData: (480, 1024), testingLabels: (480,)
total explained variance: 0.9290548743870186
```

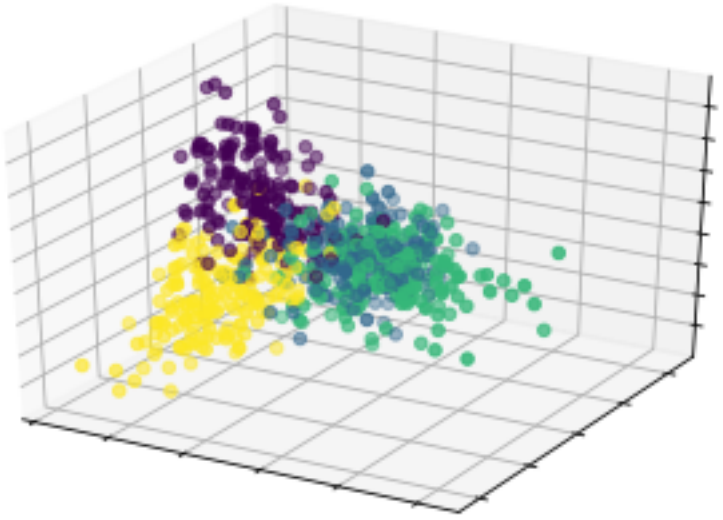



```
pca components: (128, 1024)
```




```
trainingData: (720, 128), trainingLabels: (720,)
testingData: (480, 128), testingLabels: (480,)
```





```
trainingData: (720, 3), trainingLabels: (720,)
testingData: (480, 3), testingLabels: (480,)
Iteration 1, loss = 1.02981327
Iteration 2, loss = 0.47433245
Iteration 3, loss = 0.37740764
Iteration 4, loss = 0.37732018
```

```
Iteration 5, loss = 0.37198401
Iteration 6, loss = 0.36466301
Iteration 7, loss = 0.35097083
Iteration 8, loss = 0.34696205
Iteration 9, loss = 0.34256429
Iteration 10, loss = 0.34211853
Iteration 11, loss = 0.34144038
Iteration 12, loss = 0.33613148
Iteration 13, loss = 0.33181166
Iteration 14, loss = 0.32947555
Iteration 15, loss = 0.32744083
Iteration 16, loss = 0.32580543
Iteration 17, loss = 0.32324200
Iteration 18, loss = 0.32347231
Iteration 19, loss = 0.31922149
Iteration 20, loss = 0.31934619
Iteration 21, loss = 0.31751195
Iteration 22, loss = 0.31502836
Iteration 23, loss = 0.31499656
Iteration 24, loss = 0.31129709
Iteration 25, loss = 0.30992488
Iteration 26, loss = 0.30981643
Iteration 27, loss = 0.30572001
Iteration 28, loss = 0.30553384
Iteration 29, loss = 0.30305305
Iteration 30, loss = 0.30386718
Iteration 31, loss = 0.30131446
Iteration 32, loss = 0.29965098
Iteration 33, loss = 0.29840836
Iteration 34, loss = 0.29839277
Iteration 35, loss = 0.29565798
Iteration 36, loss = 0.29603645
Iteration 37, loss = 0.29519936
Iteration 38, loss = 0.29287691
Iteration 39, loss = 0.29227968
Iteration 40, loss = 0.29123461
Iteration 41, loss = 0.28730115
Iteration 42, loss = 0.28983175
Iteration 43, loss = 0.28710082
Iteration 44, loss = 0.28175880
Iteration 45, loss = 0.28682509
Iteration 46, loss = 0.27860272
Iteration 47, loss = 0.28302011
Iteration 48, loss = 0.27606090
Iteration 49, loss = 0.28154043
Iteration 50, loss = 0.27807316
Iteration 51, loss = 0.27569498
Iteration 52, loss = 0.27307704
```

```
Iteration 53, loss = 0.27059676
Iteration 54, loss = 0.26751913
Iteration 55, loss = 0.26918348
Iteration 56, loss = 0.26759031
Iteration 57, loss = 0.26861300
Iteration 58, loss = 0.26752020
Iteration 59, loss = 0.26712720
Iteration 60, loss = 0.26377799
Iteration 61, loss = 0.26721492
Iteration 62, loss = 0.26565390
Iteration 63, loss = 0.25966407
Iteration 64, loss = 0.25960834
Iteration 65, loss = 0.26527433
Iteration 66, loss = 0.25876642
Iteration 67, loss = 0.25747173
Iteration 68, loss = 0.25734856
Iteration 69, loss = 0.25010097
Iteration 70, loss = 0.25156004
Iteration 71, loss = 0.24551777
Iteration 72, loss = 0.25108205
Iteration 73, loss = 0.24862841
Iteration 74, loss = 0.24986048
Iteration 75, loss = 0.24417075
Iteration 76, loss = 0.24657727
Iteration 77, loss = 0.24268561
Iteration 78, loss = 0.24574433
Iteration 79, loss = 0.24079642
Iteration 80, loss = 0.24227092
Iteration 81, loss = 0.23656737
Iteration 82, loss = 0.24133798
Iteration 83, loss = 0.23600206
Iteration 84, loss = 0.23078960
Iteration 85, loss = 0.23235499
Iteration 86, loss = 0.23425049
Iteration 87, loss = 0.23246441
Iteration 88, loss = 0.23752917
Iteration 89, loss = 0.23163417
Iteration 90, loss = 0.22705845
Iteration 91, loss = 0.22962105
Iteration 92, loss = 0.23162277
Iteration 93, loss = 0.22453058
Iteration 94, loss = 0.21935317
Iteration 95, loss = 0.22639379
Iteration 96, loss = 0.22283948
Iteration 97, loss = 0.21936554
Iteration 98, loss = 0.21634676
Iteration 99, loss = 0.22061709
Iteration 100, loss = 0.22177780
```

```
Iteration 101, loss = 0.21398108
Iteration 102, loss = 0.21782153
Iteration 103, loss = 0.21243173
Iteration 104, loss = 0.21022441
Iteration 105, loss = 0.20780106
Iteration 106, loss = 0.20920867
Iteration 107, loss = 0.21045328
Iteration 108, loss = 0.20517528
Iteration 109, loss = 0.20586204
Iteration 110, loss = 0.20471729
Iteration 111, loss = 0.20037841
Iteration 112, loss = 0.20483138
Iteration 113, loss = 0.19643329
Iteration 114, loss = 0.19998439
Iteration 115, loss = 0.19786168
Iteration 116, loss = 0.19695688
Iteration 117, loss = 0.19228679
Iteration 118, loss = 0.19216150
Iteration 119, loss = 0.18959064
Iteration 120, loss = 0.18803844
Iteration 121, loss = 0.19110496
Iteration 122, loss = 0.18787457
Iteration 123, loss = 0.18894181
Iteration 124, loss = 0.18620694
Iteration 125, loss = 0.18849560
Iteration 126, loss = 0.19107012
Iteration 127, loss = 0.20069554
Iteration 128, loss = 0.19806119
Iteration 129, loss = 0.19408475
Iteration 130, loss = 0.18652309
Iteration 131, loss = 0.18779921
Iteration 132, loss = 0.18874752
Iteration 133, loss = 0.18295626
Iteration 134, loss = 0.17678955
Iteration 135, loss = 0.17195089
Iteration 136, loss = 0.18007077
Iteration 137, loss = 0.17434415
Iteration 138, loss = 0.17449059
Iteration 139, loss = 0.17862485
Iteration 140, loss = 0.17164322
Iteration 141, loss = 0.16835358
Iteration 142, loss = 0.17386525
Iteration 143, loss = 0.17357227
Iteration 144, loss = 0.17840539
Iteration 145, loss = 0.18065795
Iteration 146, loss = 0.17422481
Iteration 147, loss = 0.16561040
Iteration 148, loss = 0.16342241
```

```
Iteration 149, loss = 0.16665426
Iteration 150, loss = 0.16288130
Iteration 151, loss = 0.15785272
Iteration 152, loss = 0.15739514
Iteration 153, loss = 0.15399154
Iteration 154, loss = 0.15603001
Iteration 155, loss = 0.15403024
Iteration 156, loss = 0.15017955
Iteration 157, loss = 0.15376653
Iteration 158, loss = 0.14732591
Iteration 159, loss = 0.15502368
Iteration 160, loss = 0.14950613
Iteration 161, loss = 0.14674229
Iteration 162, loss = 0.14585952
Iteration 163, loss = 0.15076190
Iteration 164, loss = 0.14467121
Iteration 165, loss = 0.15256823
Iteration 166, loss = 0.14048497
Iteration 167, loss = 0.13991758
Iteration 168, loss = 0.14171590
Iteration 169, loss = 0.14088944
Iteration 170, loss = 0.13770872
Iteration 171, loss = 0.13650052
Iteration 172, loss = 0.13378659
Iteration 173, loss = 0.14068620
Iteration 174, loss = 0.13627825
Iteration 175, loss = 0.13654749
Iteration 176, loss = 0.13877393
Iteration 177, loss = 0.14244424
Iteration 178, loss = 0.13708549
Iteration 179, loss = 0.13633138
Iteration 180, loss = 0.13415139
Iteration 181, loss = 0.12941064
Iteration 182, loss = 0.12917773
Iteration 183, loss = 0.13206962
Iteration 184, loss = 0.13012097
Iteration 185, loss = 0.12490804
Iteration 186, loss = 0.12556654
Iteration 187, loss = 0.12980989
Iteration 188, loss = 0.12269135
Iteration 189, loss = 0.13394867
Iteration 190, loss = 0.12657987
Iteration 191, loss = 0.15002922
Iteration 192, loss = 0.16122779
Iteration 193, loss = 0.14448178
Iteration 194, loss = 0.13412279
Iteration 195, loss = 0.13473380
Iteration 196, loss = 0.15014890
```

```
Iteration 197, loss = 0.13823515
Iteration 198, loss = 0.12330188
Iteration 199, loss = 0.12494636
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(1024, 1024), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=0, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
training accuracy: 0.9569444444444445
testing accuracy: 0.7083333333333334
confusion matrix:
 [[98  9  7 17]
 [16 72 16  9]
 [ 3 16 95  6]
 [25  9  7 75]]
              precision    recall  f1-score   support

           0       0.69      0.75      0.72       131
           1       0.68      0.64      0.66       113
           2       0.76      0.79      0.78       120
           3       0.70      0.65      0.67       116

   micro avg       0.71      0.71      0.71       480
   macro avg       0.71      0.71      0.71       480
weighted avg       0.71      0.71      0.71       480
```

## 12 tSNE

```
In [84]: def tsneVisualize(data, labels):
             tsne = TSNE(verbose=1, random_state=0, n_iter=2000, n_components=3)
             data = tsne.fit_transform(data)

             scatter3d(data[:, 0], data[:, 1], data[:, 2], labels)
```

```
In [85]: %matplotlib notebook
         tsneVisualize(*unpickleData())
         %matplotlib inline
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 500 samples in 0.025s...
[t-SNE] Computed neighbors for 500 samples in 0.576s...
[t-SNE] Computed conditional probabilities for sample 500 / 500
[t-SNE] Mean sigma: 2.517696
```

```
[t-SNE] KL divergence after 250 iterations with early exaggeration: 90.198715
[t-SNE] KL divergence after 2000 iterations: 0.708619


<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>


In [184]: tsneVisualize(*unpickleAugmentedData())

[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 1200 samples in 0.085s...
[t-SNE] Computed neighbors for 1200 samples in 3.200s...
[t-SNE] Computed conditional probabilities for sample 1000 / 1200
[t-SNE] Computed conditional probabilities for sample 1200 / 1200
[t-SNE] Mean sigma: 2.137217
[t-SNE] KL divergence after 250 iterations with early exaggeration: 78.764046
[t-SNE] KL divergence after 2000 iterations: 1.173605
```