1 Python correction

The following imports may be considered.

```python
# image manipulation and features construction
import glob
from skimage import measure, io
import numpy as np
import matplotlib.pyplot as plt

# preprocessing data and normalization
from sklearn.preprocessing import scale
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import minmax_scale
from sklearn.preprocessing import MaxAbsScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing.data import QuantileTransformer

# learning methods
from sklearn import svm
from sklearn.cluster import KMeans
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix

# plot confusion matrix
import seaborn as sn
import pandas as pd
```

1.1 Feature extraction

We make a loop on the whole database to extract some features of each image. The 9 features used here are: area, convex area, eccentricity, equivalent diameter, extent, major axis length, minor axis length, perimeter and solidity.

```python
# Definitions of the database, classes and images
rep = '../matlab/images_Kimia216/';
classes = ['bird','bone','brick','camel','car','children',
    'classic','elephant','face','fork','fountain',
    'glass','hammer','heart','key','misk','ray','turtle'];
nbClasses = len(classes);
nbImages = 12;

# The features are manually computed
properties = np.zeros((nbClasses*nbImages,9));
target = np.zeros(nbClasses * nbImages);
index=0;
for ind_c, c in enumerate(classes):
    filelist = glob.glob(rep+c+'*');
    for filename in filelist:
        I = io.imread(filename);
        prop = measure.regionprops(I);
        properties[index, 0] = prop[0].area;
        properties[index, 1] = prop[0].convex_area;
        properties[index, 2] = prop[0].eccentricity;
        properties[index, 3] = prop[0].equivalent_diameter;
        properties[index, 4] = prop[0].extent;
        properties[index, 5] = prop[0].major_axis_length;
        properties[index, 6] = prop[0].minor_axis_length;
        properties[index, 7] = prop[0].perimeter;
        properties[index, 8] = prop[0].solidity;
        target[index] = ind_c;

        index = index+1;
```

Note that in the same time, the target array (required in the following) is built within this loop. It represents the true classes of the objects.

1.2 Classification

We used a training set of 75% of the database and 25% for the test set.

```python
# percentage of the data used for splitting into train/test
percentTest = .25;

# MLP Classifier (Multi-Layer Perceptron)
# the data are first scaled
propertiesMLP = StandardScaler().fit_transform(properties);
prop_train, prop_test, target_train, target_test = train_test_split(
    ↪ propertiesMLP, target, test_size=percentTest, random_state=0);
```

The network is created with 1 hidden layers of 10 neurons.

```python
# feedforward neural network
# max_iter should be extended max_iter=100000 for adam or sgd solvers
mlp = MLPClassifier(hidden_layer_sizes=(10,), solver='lbfgs');

target_pred = mlp.fit(prop_train, target_train).predict(prop_test)

print("Training set score: %f" % mlp.score(prop_train, target_train))
```

The training score should be around 1, according to the training dataset and the different parameters employed.

```
Training set score: 1.000000
```

1.3 Performance

The confusion matrix gives an idea of the overall performance. The following function uses the modules seaborn and pandas to generate a heatmap that displays the confusion matrix.

```python
def plot_cm(cm, classes, normalize=False, cmap=plt.cm.Blues):
    """
    Plot confusion matrix
    cm: confusion matrix, as ouput by sklearn.metrics.confusion_matrix
    classes: labels to be used
    normalize: display number (False by default) or fraction (True)
    cmap: colormap

    returns: figure that can be used for pdf export
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    fmt = '.1f' if normalize else 'd'
    df_cm = pd.DataFrame(cm, index = classes, columns = classes);

    fig = plt.figure();
    sn.set(font_scale=.8)
    sn.heatmap(df_cm, annot=True, cmap = cmap, fmt=fmt);
    plt.xlabel('Target label')
    plt.ylabel('True label')

    return fig;
```

The previous function is then used, and illustrated in Fig.1.

```
cnf_matrix = confusion_matrix(target_test, target_pred);
fig=plot_cm(cnf_matrix, classes, normalize=True);
```

sklearn.metrics.classification_report can be used to display the performance.

```
             precision    recall  f1-score   support

       bird       0.75      1.00      0.86         3
       bone       1.00      1.00      1.00         5
      brick       1.00      1.00      1.00         1
      camel       1.00      1.00      1.00         3
        car       1.00      1.00      1.00         1
   children       1.00      1.00      1.00         2
    classic       1.00      1.00      1.00         5
   elephant       1.00      1.00      1.00         3
       face       1.00      1.00      1.00         4
       fork       1.00      1.00      1.00         2
   fountain       1.00      1.00      1.00         3
      glass       1.00      1.00      1.00         5
     hammer       1.00      0.75      0.86         4
      heart       1.00      1.00      1.00         2
        key       1.00      1.00      1.00         3
       misk       1.00      1.00      1.00         3
        ray       1.00      1.00      1.00         4
     turtle       1.00      1.00      1.00         1

avg / total       0.99      0.98      0.98        54
```

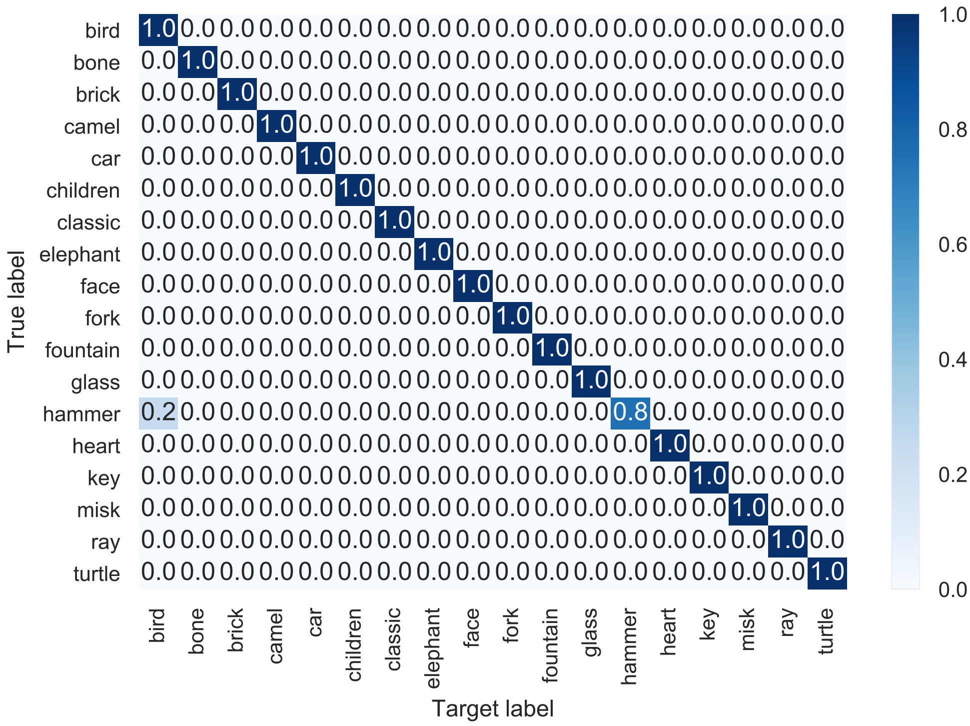

Figure 1: Normalized confusion matrix of the classification result.