cdv 2017-2018 impact of clustering

August 29, 2018

0.0.1 I) Loading data and preparing dataset - scope 2017-2018

In [1]:
```
from pathlib import Path
import pandas as pd
import numpy as np
from datetime import datetime
import time
import matplotlib.pyplot as plt
%matplotlib inline
#%pylab inline
import itertools
import pickle
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score, GridSearchCV
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import LabelBinarizer
from sklearn.preprocessing import OneHotEncoder
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
from sklearn.feature_selection import RFECV
```

In [2]:
```
path_project = Path.home() / Path('Google Drive/Felix')
path_data = path_project / Path("data")
path_dump = path_project / Path("dump")
```

In [3]:
```
# loading cdv data
file = path_data / Path("felix.csv")
with Path.open(file, 'rb') as fp:
    cdv = pd.read_csv(fp,  encoding='cp1252',low_memory=False, index_col = 0)
# loadind cdv data without format
file = path_data / Path("felix_ssfmt.csv")
with Path.open(file, 'rb') as fp:
    cdv_ssfmt = pd.read_csv(fp,  encoding='cp1252',low_memory=False, index_col = 0)
```

```
In [4]: # load various variable set
        filename = path_dump / Path("dict_var_groups.sav")
        with open(filename, 'rb') as fp:
             dict_var_groups = pickle.load(fp)

        scope_2017_2018_var = dict_var_groups['scope_2017_2018_var']
        pred_var = dict_var_groups['pred_var']
        com_var = dict_var_groups['com_var']
        tech_var = dict_var_groups['tech_var']
        text_var = dict_var_groups['text_var']
        bizz_var = dict_var_groups['bizz_var']
        cat_var = dict_var_groups['cat_var']
        cat_max9_var = dict_var_groups['cat_max9_var']
        cat_min10_var = dict_var_groups['cat_min10_var']
        quant_var = dict_var_groups['quant_var']

In [5]: exclusion = com_var | tech_var | bizz_var | text_var
        scope_2017_2018_var_kept = scope_2017_2018_var - exclusion
        cat_var_kept = cat_max9_var & scope_2017_2018_var_kept
        scope_quant_var = (quant_var & scope_2017_2018_var_kept)
        quant_null = np.sum(cdv_ssfmt.loc[:,scope_quant_var].isnull())
        quant_var_kept = set(quant_null[quant_null < 200].index)

        print(f"Out of {cdv.shape[1]} variable {len(scope_2017_2018_var)} \
        are used in 2017 and 2018 ")
        print(f"{len(scope_2017_2018_var & exclusion)} of 'technical' variable \
        such as 'inseenum' are excluded ")
        print(f"{len(scope_2017_2018_var_kept)} are remaining :")
        print(f"\t{len(cat_var & scope_2017_2018_var_kept)} \
        categorial variables : ")
        print(f"\t\t{len(cat_max9_var & scope_2017_2018_var_kept)} \
        with maximum 9 modalities  ")
        print(f"\t\t{len(cat_min10_var & scope_2017_2018_var_kept)} \
        with more modalities ... excluded")
        print(f"\t{len(quant_var & scope_2017_2018_var_kept)} \
        variables are quantitative ")
        print(f"\t\t{len(quant_var_kept)} have less than 200 missing values")
        print(f"\t\t{len(scope_quant_var)-len(quant_var_kept)} \
        have more ... excluded")

        scope = cat_var_kept | quant_var_kept
        df = cdv_ssfmt.loc[cdv_ssfmt['ANNEEFUZ'].isin({39,40}),scope]
        df.loc[:,cat_var_kept - {"HEUREUX"}] = cdv.loc[:,cat_var_kept - {"HEUREUX"}]
        print(f"\nFinal number of variable kept : {df.shape[1]}")

Out of 353 variable 297 are used in 2017 and 2018
31 of 'technical' variable such as 'inseenum' are excluded
266 are remaining :
```

```
        180 categorial variables :
                165 with maximum 9 modalities
                15 with more modalities ... excluded
        86 variables are quantitative
                60 have less than 200 missing values
                26 have more ... excluded

Final number of variable kept : 225
```

```
In [6]: p = df.shape[1]
        print(f"{p} columns out of which {len(cat_var_kept)-1} \
        are corresponding to categorial features")
```

```
225 columns out of which 164 are corresponding to categorial features
```

```
In [7]: df = pd.get_dummies(df,
                            columns=cat_var_kept - {"HEUREUX"},
                            dummy_na = True,
                            drop_first=1)
```

```
In [8]: q = df.shape[1]
        print(f"{q} columns after encoding of {len(cat_var_kept)-1} categorial \
        variables in {len(cat_var_kept)-1+q-p} binary variables \
        (K-1 one hot encoding)")
```

```
645 columns after encoding of 164 categorial variables in 584 binary variables (K-1 one hot enco
```

```
In [9]: # encoding of "HEUREUX" '[nsp]'
        df.loc[df["HEUREUX"]==5,"HEUREUX"]= None
        df = df.loc[np.isfinite(df['HEUREUX']).index,:]


        # treating remaining missing values
        features = df.columns.drop(['HEUREUX'])
        df_tmp = df.loc[:,set(features) | {"HEUREUX"}].dropna()

        X = df_tmp.loc[:,features]
        y = df_tmp["HEUREUX"]

        X_train, X_test, y_train, y_test = train_test_split(X,
                                                            y,
                                                            test_size=0.2,
                                                            random_state=42
                                                           )

        scaler = StandardScaler().fit(X_train)
```

```
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

print(f"Number exemple: {y.shape[0]}\n\
- training set: {y_train.shape[0]}\n\
- test set: {y_test.shape[0]}")
print(f"Number of features: p={X_train.shape[1]}")
print(f"Number of class: {len(np.unique(y))}")
for c in np.unique(y):
    print(f"class {c:0.0f} : {100*np.sum(y==c)/len(y):0.1f}%")
```

```
Number exemple: 5682
- training set: 4545
- test set: 1137
Number of features: p=644
Number of class: 4
class 1 : 2.0%
class 2 : 34.8%
class 3 : 47.9%
class 4 : 15.3%
```

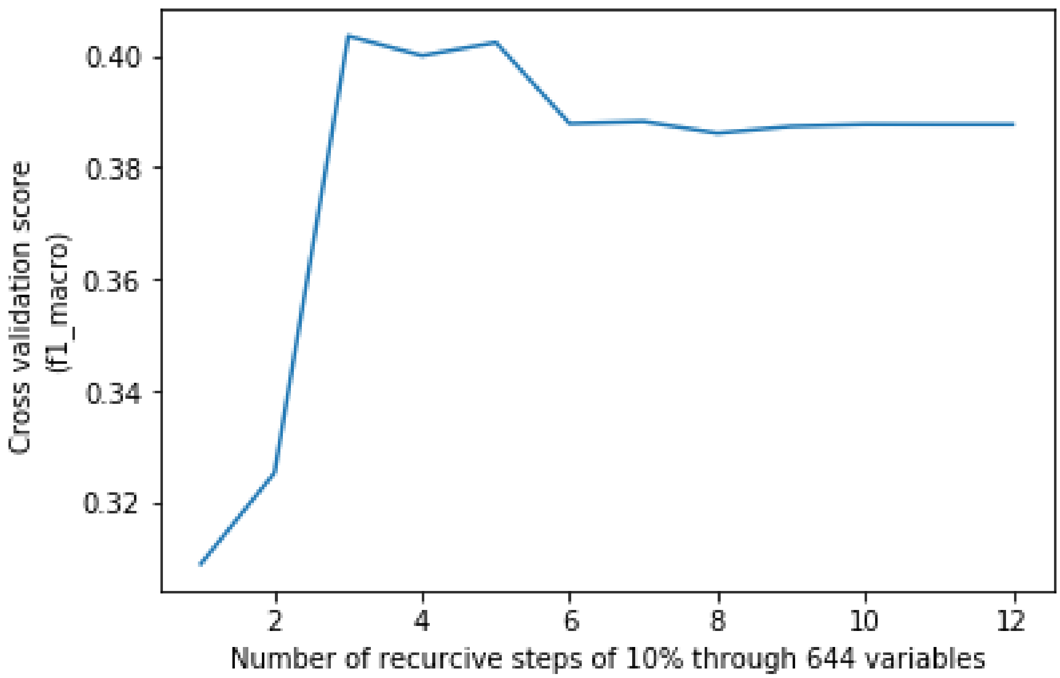
0.0.2 II) Feature selection

```
In [10]: startTime = time.time()

         scoring='f1_macro'
         step = 0.1

         clf = LogisticRegression(C=1,
                                  penalty='l1',
                                  class_weight='balanced',
                                  random_state=42)

         rfecv = RFECV(estimator=clf, step=step, cv=StratifiedKFold(2),
                       scoring=scoring)

         rfecv.fit(X_train, y_train)
         print(f"done in {time.time() - startTime:0.1f} s")
         print("Optimal number of features : %d" % rfecv.n_features_)

         # Plot number of features VS. cross-validation scores
         plt.figure()
         plt.xlabel(f"Number of recurcive steps of {100*step:0.0f}% through {X_train.shape[1]} v
         plt.ylabel(f"Cross validation score \n({scoring})")
         plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_)
         plt.show()
```

```
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
```

```
  'precision', 'predicted', average, warn_for)
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
```

```
done in 635.6 s
Optimal number of features : 68
```




```
In [11]: lasso_mask = rfecv.support_
         X_train = X_train[:,lasso_mask]
         X_test = X_test[:,lasso_mask]
         print(f"Number of features: p={X_train.shape[1]}")
```

```
Number of features: p=68
```

0.0.3 III) Model valuation

a) Random Forest

```
In [12]: startTime = time.time()
         n_estimators_range = [16,32,64,128,256]
         max_depth_range = [2,4,8,16,32,64,128,256]
         param_grid = dict(n_estimators=n_estimators_range, max_depth = max_depth_range)
```

```python
params = {'max_features' :'sqrt',
          'random_state' : 32,
          'min_samples_split' : 2,
          'class_weight' : 'balanced'
         }

clf = RandomForestClassifier(**params)

grid = GridSearchCV(clf,
                    scoring='f1_macro',
                    param_grid=param_grid)

grid.fit(X_train, y_train)
print(f"Determination of optimal hyperparameters in \
{time.time() - startTime:0.1f} s")
print(f"Optimal values are {grid.best_params_} \n\
F1 weighted Score of cross valdation {100*grid.best_score_:0.2f}%")

# Learning on full training set with optimals hyperparameters and score on test set
params = {'max_features' :'sqrt', 'random_state' : 32,
          'min_samples_split' : 2, 'class_weight' : 'balanced',
          'n_estimators' : grid.best_params_['n_estimators'],
          'max_depth' : grid.best_params_['max_depth']}
clf = RandomForestClassifier(**params).fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)
y_test_pred = clf.predict(X_test)
print(f"Random Forest, p={X_train.shape[1]}")
print(f"Accuracy: {accuracy*100:0.2f}%")
print(f"... done in {time.time() - startTime:0.1f}")
```

```
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
```

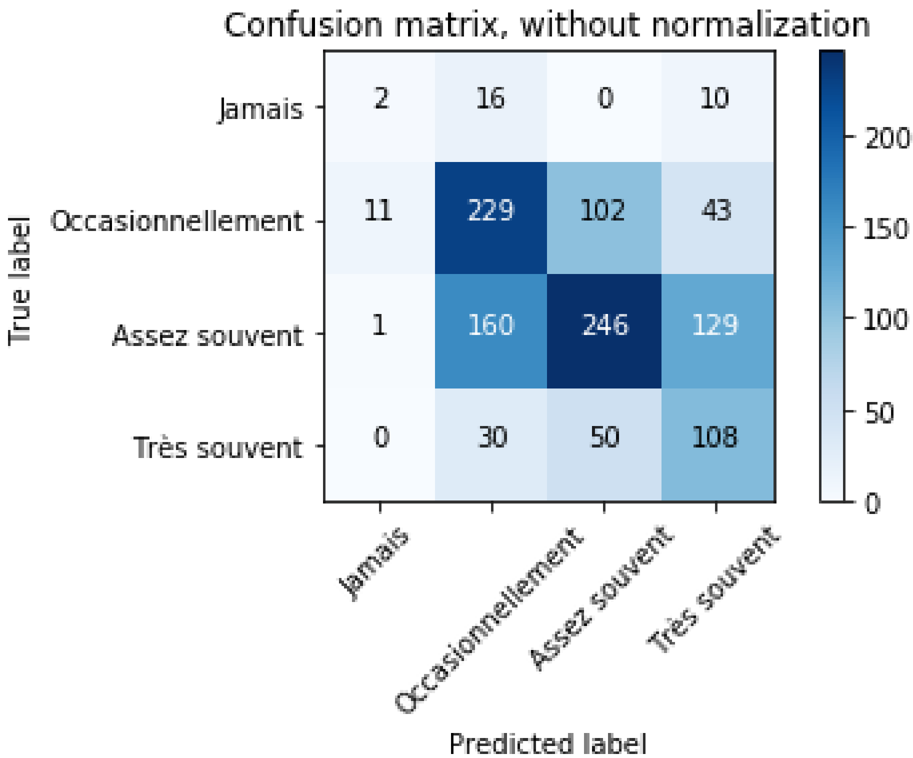
```
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Determination of optimal hyperparameters in 118.0 s
Optimal values are {'max_depth': 8, 'n_estimators': 64}
F1 weighted Score of cross valdation 41.66%
Random Forest, p=68
Accuracy: 51.45%
... done in 118.8
```

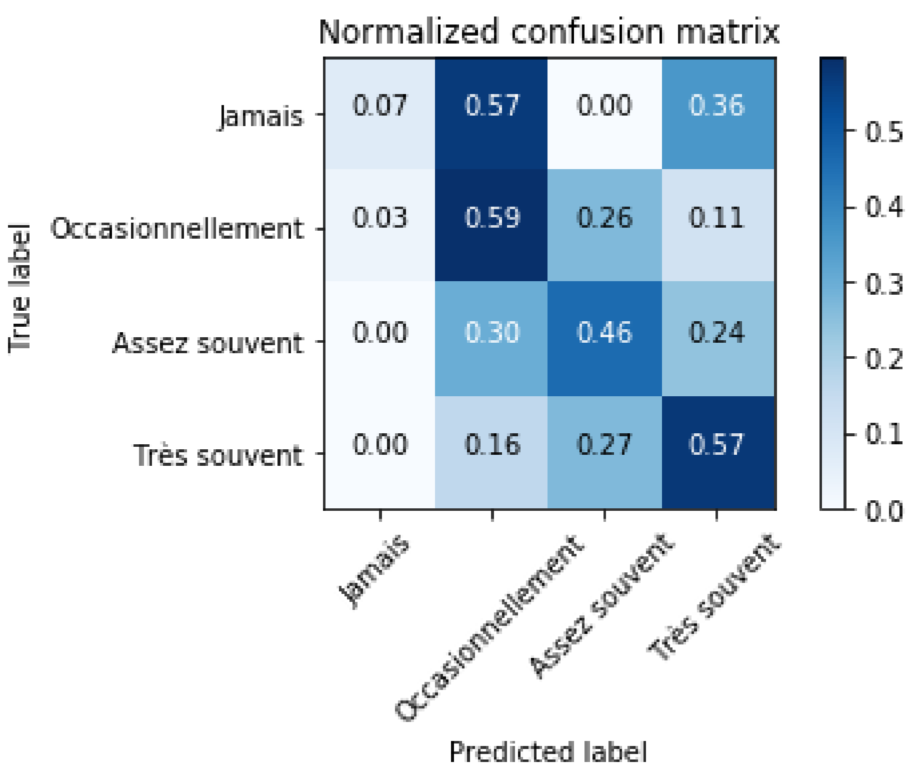
In [13]:
```python
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
```

```python
                plt.text(j, i, format(cm[i, j], fmt),
                         horizontalalignment="center",
                         color="white" if cm[i, j] > thresh else "black")

            plt.tight_layout()
            plt.ylabel('True label')
            plt.xlabel('Predicted label')
```

In [14]:
```python
class_names = ["Jamais",
               "Occasionnellement",
               "Assez souvent",
               "Très souvent"]
# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, y_test_pred)
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title='Confusion matrix, without normalization')

# Plot normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize=True,
                      title='Normalized confusion matrix')

plt.show()
```

```
Confusion matrix, without normalization
[[  2  16   0  10]
 [ 11 229 102  43]
 [  1 160 246 129]
 [  0  30  50 108]]
Normalized confusion matrix
[[ 0.07  0.57  0.    0.36]
 [ 0.03  0.59  0.26  0.11]
 [ 0.    0.3   0.46  0.24]
 [ 0.    0.16  0.27  0.57]]
```


Confusion matrix, without normalization
True label
Jamais
Occasionnellement
Assez souvent
Très souvent
2
16
0
10
11
229
102
43
1
160
246
129
0
30
50
108
0
50
100
150
200
Predicted label


Normalized confusion matrix
True label
Jamais
Occasionnellement
Assez souvent
Très souvent
0.07
0.57
0.00
0.36
0.03
0.59
0.26
0.11
0.00
0.30
0.46
0.24
0.00
0.16
0.27
0.57
0.0
0.1
0.2
0.3
0.4
0.5
Predicted label

```
In [15]: acurracy = clf.score(X_test, y_test)
         print(f"Score :{acurracy*100:0.4f} %")
```

```
Score :51.4512 %
```

```
In [16]: y_test_pred = clf.predict(X_test)
         f1_score(y_test, y_test_pred, labels = [1,2,3,4], average=None)
```

```
Out[16]: array([ 0.1 ,  0.56,  0.53,  0.45])
```

```
In [17]: f1_macro = f1_score(y_test, y_test_pred, average='macro')
         f1_weighted = f1_score(y_test, y_test_pred, average='weighted')
         print(f"Score :\nf1 macro : {f1_macro*100:0.4f} %\n\
         f1 weighted : {f1_weighted*100:0.4f} %\nacurracy : {acurracy*100:0.4f} %")
```

```
Score :
f1 macro : 40.8106 %
f1 weighted : 51.4515 %
acurracy : 51.4512 %
```

b) Support Vector Machine with gausian kernel

```
In [18]: startTime = time.time()

         nb_value = 4
         C_log = np.logspace(-2,2,nb_value)
         gamma_log = np.logspace(-4,0, nb_value)
         param_grid = dict(C=C_log, gamma=gamma_log)

         params = { 'kernel' :'rbf', 'class_weight' : 'balanced'}

         clf = SVC(**params)
         grid = GridSearchCV(clf, scoring='f1_micro', param_grid=param_grid)
         grid.fit(X_train, y_train)
         print(f"Determination of optimal hyperparameters in {time.time() - startTime:0.1f} s")
         print(f"Optimal values are {grid.best_params_} \nAccuracy of cross valdation {100*grid.

         # Learning on full training set with optimals hyperparameters and score on test set
         params = {'kernel' :'rbf',
                   'C' : grid.best_params_['C'],
                   "gamma" : grid.best_params_['gamma'],
                   'class_weight' : 'balanced'}
         clf = SVC(**params).fit(X_train, y_train)
         y_test_pred = clf.predict(X_test)
```

```
        print(f"... done in {time.time() - startTime:0.1f}")
        print(f"SVM with Gaussian kernel, p={X_train.shape[1]}")
        accuracy = clf.score(X_test, y_test_pred)
        f1_macro = f1_score(y_test, y_test_pred, average='macro')
        f1_weighted = f1_score(y_test, y_test_pred, average='weighted')
        print(f"Score :\nf1 macro : {f1_macro*100:0.4f} %\n\
        f1 weighted : {f1_weighted*100:0.4f} %\nacurracy : {acurracy*100:0.4f} %")
```

```
Determination of optimal hyperparameters in 308.1 s
Optimal values are {'C': 4.6415888336127775, 'gamma': 0.046415888336127774}
Accuracy of cross valdation 51.97%
... done in 317.2
SVM with Gaussian kernel, p=68
Score :
f1 macro : 33.0561 %
f1 weighted : 48.7099 %
acurracy : 51.4512 %
```

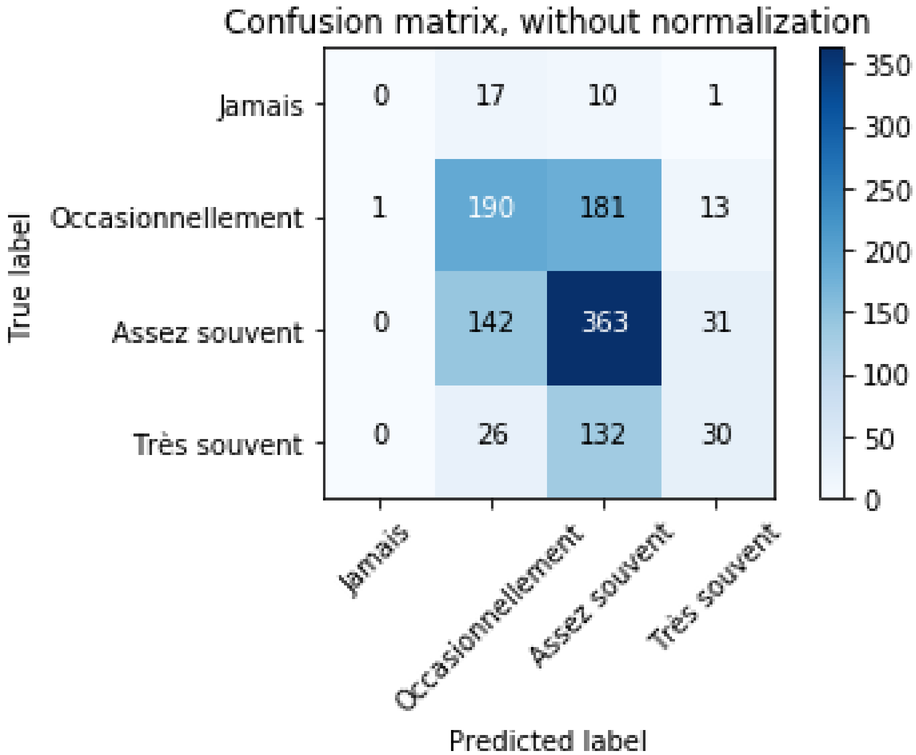
In [19]:
```
f1_score(y_test, y_test_pred, labels = [1,2,3,4], average=None)
```

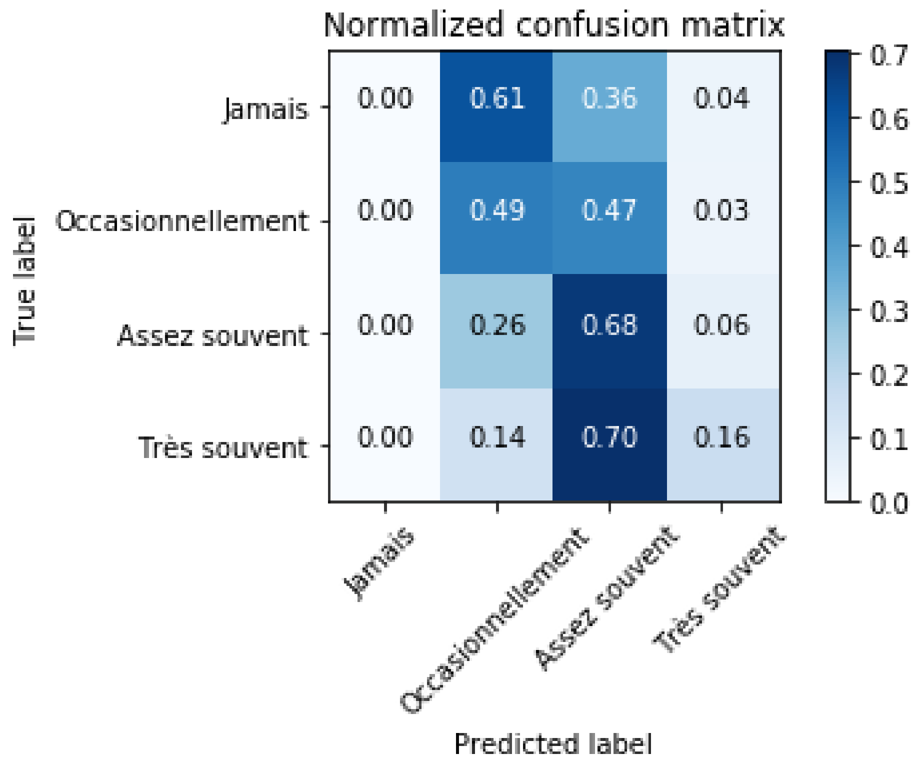
Out[19]:
```
array([ 0.  ,  0.5 ,  0.59,  0.23])
```

In [20]:
```
class_names = ["Jamais",
               "Occasionnellement",
               "Assez souvent",
               "Très souvent"]
# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, y_test_pred)
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title='Confusion matrix, without normalization')

# Plot normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize=True,
                      title='Normalized confusion matrix')

plt.show()
```

```
Confusion matrix, without normalization
[[  0  17  10   1]
 [  1 190 181  13]
 [  0 142 363  31]
 [  0  26 132  30]]
Normalized confusion matrix
```

```
[[ 0.    0.61  0.36  0.04]
 [ 0.    0.49  0.47  0.03]
 [ 0.    0.26  0.68  0.06]
 [ 0.    0.14  0.7   0.16]]
```





0.0.4 IV) Load, learn and valuate model on clusters

```
In [21]: # loading cdv data
         file = path_data / Path("clustTest1.csv")
         with Path.open(file, 'rb') as fp:
             clustTest1 = pd.read_csv(fp,  encoding='utf-8',low_memory=False, sep=";", index_col
```

```
In [22]: clustTest1.head()
```

```
Out[22]:         clust1  clust2  clust3  clust4  clust5
         INTER6
         390001       1       4       5     1.0     2.0
         390002       2       6       4     7.0     5.0
         390003       2       5       4     3.0     2.0
         390004       3       6       5     7.0     5.0
         390005       3       1       1     4.0     1.0
```

```
In [83]: n_estimators_range = [16,32,64,128]
         max_depth_range = [2,4,8,16,32,64]
         param_grid = dict(n_estimators=n_estimators_range, max_depth = max_depth_range)
         params = {'max_features' :'sqrt',
                   'random_state' : 32,
                   'min_samples_split' : 2,
```

```
          'class_weight' : 'balanced'
         }

for method in clustering_methods:
    print(f"\nAnalysis cluster method {method}")
    cluster_list = clustTest1[method].unique()[0:2]
    print(f"liste of clusters : {cluster_list}")
    for cluster in cluster_list:
        index_scope = clustTest1.loc[clustTest1[method]==cluster,:].index
        print(f"cluster {cluster} : {len(index_scope)} elements")

        # treating remaining missing values
        features = df.columns.drop(['HEUREUX'])[lasso_mask]
        df_tmp = df.loc[index_scope,set(features) | {"HEUREUX"}].dropna()

        X = df_tmp.loc[:,features]
        y = df_tmp["HEUREUX"]

        X_train, X_test, y_train, y_test = train_test_split(X,
                                                             y,
                                                             test_size=0.2,
                                                             random_state=42
                                                            )

        scaler = StandardScaler().fit(X_train)
        X_train = scaler.transform(X_train)
        X_test = scaler.transform(X_test)

        print(f"Number exemple: {y.shape[0]}\n\
        - training set: {y_train.shape[0]}\n\
        - test set: {y_test.shape[0]}")
        print(f"Number of features: p={X_train.shape[1]}")
        print(f"Number of class: {len(np.unique(y))}")
        for c in np.unique(y):
            print(f"class {c:0.0f} : {100*np.sum(y==c)/len(y):0.1f}%")


            startTime = time.time()
            clf = RandomForestClassifier(**params)
            grid = GridSearchCV(clf,
                                scoring='f1_micro',
                                param_grid=param_grid)

            grid.fit(X_train, y_train)
            print(f"Determination of optimal hyperparameters in \
            {time.time() - startTime:0.1f} s")
            print(f"Optimal values are {grid.best_params_} \n\
            F1 weighted Score of cross valdation {100*grid.best_score_:0.2f}%")
```

```
# Learning on full training set with optimals hyperparameters and score on
params_opt = {'max_features' :'sqrt', 'random_state' : 32,
              'min_samples_split' : 2, 'class_weight' : 'balanced',
              'n_estimators' : grid.best_params_['n_estimators'],
              'max_depth' : grid.best_params_['max_depth']}
clf = RandomForestClassifier(**params_opt).fit(X_train, y_train)

accuracy = clf.score(X_test, y_test)
y_test_pred = clf.predict(X_test)
f1_scores = f1_score(y_test, y_test_pred, labels = [1,2,3,4], average=None)
f1_macro = f1_score(y_test, y_test_pred, average='macro')
f1_weighted = f1_score(y_test, y_test_pred, average='weighted')
print(f"Random Forest, p={X_train.shape[1]}")
print(f"f1 scores: {f1_scores}")
print(f"Score :\nf1 macro : {f1_macro*100:0.4f} %\n\
f1 weighted : {f1_weighted*100:0.4f} %\nacurracy : {acurracy*100:0.4f} %")
```

```
Analysis cluster method clust1
liste of clusters : [1 2]
cluster 1 : 295 elements
Number exemple: 292
        - training set: 233
        - test set: 59
Number of features: p=68
Number of class: 4
class 1 : 5.1%
Determination of optimal hyperparameters in             11.1 s
Optimal values are {'max_depth': 4, 'n_estimators': 16}
            F1 weighted Score of cross valdation 44.64%
Random Forest, p=68
f1 scores: [ 0.    0.47  0.53  0.38]
Score :
f1 macro : 34.4170 %
            f1 weighted : 47.0447 %
acurracy : 51.4512 %
class 2 : 41.1%
Determination of optimal hyperparameters in             14.2 s
Optimal values are {'max_depth': 4, 'n_estimators': 16}
            F1 weighted Score of cross valdation 44.64%
Random Forest, p=68
f1 scores: [ 0.    0.47  0.53  0.38]
Score :
f1 macro : 34.4170 %
            f1 weighted : 47.0447 %
acurracy : 51.4512 %
class 3 : 39.0%
```

```
Determination of optimal hyperparameters in             16.3 s
Optimal values are {'max_depth': 4, 'n_estimators': 16}
            F1 weighted Score of cross valdation 44.64%
Random Forest, p=68
f1 scores: [ 0.    0.47  0.53  0.38]
Score :
f1 macro : 34.4170 %
            f1 weighted : 47.0447 %
acurracy : 51.4512 %
class 4 : 14.7%
Determination of optimal hyperparameters in             10.8 s
Optimal values are {'max_depth': 4, 'n_estimators': 16}
            F1 weighted Score of cross valdation 44.64%
Random Forest, p=68
f1 scores: [ 0.    0.47  0.53  0.38]
Score :
f1 macro : 34.4170 %
            f1 weighted : 47.0447 %
acurracy : 51.4512 %
cluster 2 : 1729 elements
Number exemple: 1725
        - training set: 1380
        - test set: 345
Number of features: p=68
Number of class: 4
class 1 : 2.3%
Determination of optimal hyperparameters in             18.3 s
Optimal values are {'max_depth': 16, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.29%
```

```
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
```

```
Random Forest, p=68
f1 scores: [ 0.    0.59  0.64  0.13]
Score :
f1 macro : 33.8844 %
            f1 weighted : 54.8042 %
acurracy : 51.4512 %
class 2 : 37.9%
Determination of optimal hyperparameters in             19.0 s
Optimal values are {'max_depth': 16, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.29%
```

```
//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
```

```
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.59  0.64  0.13]
Score :
f1 macro : 33.8844 %
            f1 weighted : 54.8042 %
acurracy : 51.4512 %
class 3 : 46.8%
Determination of optimal hyperparameters in             18.7 s
Optimal values are {'max_depth': 16, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.29%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.59  0.64  0.13]
Score :
f1 macro : 33.8844 %
            f1 weighted : 54.8042 %
acurracy : 51.4512 %
class 4 : 13.0%
Determination of optimal hyperparameters in             17.2 s
Optimal values are {'max_depth': 16, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.29%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.59  0.64  0.13]
Score :
f1 macro : 33.8844 %
            f1 weighted : 54.8042 %
acurracy : 51.4512 %

Analysis cluster method clust2
liste of clusters : [4 6]
cluster 4 : 212 elements
Number exemple: 211
        - training set: 168
        - test set: 43
Number of features: p=68
```

```
Number of class: 4
class 1 : 8.1%
Determination of optimal hyperparameters in             10.6 s
Optimal values are {'max_depth': 16, 'n_estimators': 32}
            F1 weighted Score of cross valdation 50.00%
Random Forest, p=68
f1 scores: [ 0.    0.56  0.5   0.44]
Score :
f1 macro : 37.7137 %
            f1 weighted : 49.2844 %
acurracy : 51.4512 %
class 2 : 42.7%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Determination of optimal hyperparameters in             13.7 s
Optimal values are {'max_depth': 16, 'n_estimators': 32}
            F1 weighted Score of cross valdation 50.00%
Random Forest, p=68
f1 scores: [ 0.    0.56  0.5   0.44]
Score :
f1 macro : 37.7137 %
            f1 weighted : 49.2844 %
acurracy : 51.4512 %
class 3 : 32.2%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Determination of optimal hyperparameters in             10.5 s
Optimal values are {'max_depth': 16, 'n_estimators': 32}
            F1 weighted Score of cross valdation 50.00%
Random Forest, p=68
f1 scores: [ 0.    0.56  0.5   0.44]
Score :
f1 macro : 37.7137 %
            f1 weighted : 49.2844 %
acurracy : 51.4512 %
class 4 : 17.1%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)
```

```
Determination of optimal hyperparameters in             10.0 s
Optimal values are {'max_depth': 16, 'n_estimators': 32}
            F1 weighted Score of cross valdation 50.00%
Random Forest, p=68
f1 scores: [ 0.    0.56  0.5   0.44]
Score :
f1 macro : 37.7137 %
            f1 weighted : 49.2844 %
acurracy : 51.4512 %
cluster 6 : 1137 elements
Number exemple: 1132
        - training set: 905
        - test set: 227
Number of features: p=68
Number of class: 4
class 1 : 1.8%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Determination of optimal hyperparameters in             14.9 s
Optimal values are {'max_depth': 8, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.14%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.58  0.63  0.34]
Score :
f1 macro : 38.8312 %
            f1 weighted : 55.3072 %
acurracy : 51.4512 %
class 2 : 33.3%
Determination of optimal hyperparameters in             13.6 s
Optimal values are {'max_depth': 8, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.14%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.58  0.63  0.34]
```

```
Score :
f1 macro : 38.8312 %
            f1 weighted : 55.3072 %
acurracy : 51.4512 %
class 3 : 48.2%
Determination of optimal hyperparameters in             14.8 s
Optimal values are {'max_depth': 8, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.14%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.58  0.63  0.34]
Score :
f1 macro : 38.8312 %
            f1 weighted : 55.3072 %
acurracy : 51.4512 %
class 4 : 16.7%
Determination of optimal hyperparameters in             21.8 s
Optimal values are {'max_depth': 8, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.14%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.58  0.63  0.34]
Score :
f1 macro : 38.8312 %
            f1 weighted : 55.3072 %
acurracy : 51.4512 %

Analysis cluster method clust3
liste of clusters : [5 4]
cluster 5 : 373 elements
Number exemple: 373
        - training set: 298
        - test set: 75
Number of features: p=68
Number of class: 4
class 1 : 4.3%
Determination of optimal hyperparameters in             19.7 s
Optimal values are {'max_depth': 8, 'n_estimators': 16}
            F1 weighted Score of cross valdation 48.32%
```

```
Random Forest, p=68
f1 scores: [ 0.    0.57  0.48  0.5 ]
Score :
f1 macro : 38.8547 %
            f1 weighted : 51.1429 %
acurracy : 51.4512 %
class 2 : 41.6%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Determination of optimal hyperparameters in             15.3 s
Optimal values are {'max_depth': 8, 'n_estimators': 16}
            F1 weighted Score of cross valdation 48.32%
Random Forest, p=68
f1 scores: [ 0.    0.57  0.48  0.5 ]
Score :
f1 macro : 38.8547 %
            f1 weighted : 51.1429 %
acurracy : 51.4512 %
class 3 : 37.5%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Determination of optimal hyperparameters in             11.6 s
Optimal values are {'max_depth': 8, 'n_estimators': 16}
            F1 weighted Score of cross valdation 48.32%
Random Forest, p=68
f1 scores: [ 0.    0.57  0.48  0.5 ]
Score :
f1 macro : 38.8547 %
            f1 weighted : 51.1429 %
acurracy : 51.4512 %
class 4 : 16.6%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Determination of optimal hyperparameters in             11.7 s
Optimal values are {'max_depth': 8, 'n_estimators': 16}
            F1 weighted Score of cross valdation 48.32%
Random Forest, p=68
```

```
f1 scores: [ 0.    0.57  0.48  0.5 ]
Score :
f1 macro : 38.8547 %
            f1 weighted : 51.1429 %
acurracy : 51.4512 %
cluster 4 : 2682 elements
Number exemple: 2674
        - training set: 2139
        - test set: 535
Number of features: p=68
Number of class: 4
class 1 : 1.4%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Determination of optimal hyperparameters in             22.1 s
Optimal values are {'max_depth': 16, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.59%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.53  0.64  0.29]
Score :
f1 macro : 36.4679 %
            f1 weighted : 54.0350 %
acurracy : 51.4512 %
class 2 : 34.3%
Determination of optimal hyperparameters in             20.8 s
Optimal values are {'max_depth': 16, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.59%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.53  0.64  0.29]
Score :
f1 macro : 36.4679 %
            f1 weighted : 54.0350 %
acurracy : 51.4512 %
```

```
class 3 : 49.0%
Determination of optimal hyperparameters in             22.1 s
Optimal values are {'max_depth': 16, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.59%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.53  0.64  0.29]
Score :
f1 macro : 36.4679 %
            f1 weighted : 54.0350 %
acurracy : 51.4512 %
class 4 : 15.3%
Determination of optimal hyperparameters in             24.4 s
Optimal values are {'max_depth': 16, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.59%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.53  0.64  0.29]
Score :
f1 macro : 36.4679 %
            f1 weighted : 54.0350 %
acurracy : 51.4512 %

Analysis cluster method clust4
liste of clusters : [ 1.  7.]
cluster 1.0 : 556 elements
Number exemple: 556
        - training set: 444
        - test set: 112
Number of features: p=68
Number of class: 4
class 1 : 3.2%
Determination of optimal hyperparameters in             12.7 s
Optimal values are {'max_depth': 8, 'n_estimators': 32}
            F1 weighted Score of cross valdation 48.87%
Random Forest, p=68
f1 scores: [ 0.67  0.37  0.54  0.47]
Score :
f1 macro : 51.0766 %
```

```
            f1 weighted : 47.7461 %
acurracy : 51.4512 %
class 2 : 21.8%
Determination of optimal hyperparameters in             12.7 s
Optimal values are {'max_depth': 8, 'n_estimators': 32}
            F1 weighted Score of cross valdation 48.87%
Random Forest, p=68
f1 scores: [ 0.67  0.37  0.54  0.47]
Score :
f1 macro : 51.0766 %
            f1 weighted : 47.7461 %
acurracy : 51.4512 %
class 3 : 44.8%
Determination of optimal hyperparameters in             12.5 s
Optimal values are {'max_depth': 8, 'n_estimators': 32}
            F1 weighted Score of cross valdation 48.87%
Random Forest, p=68
f1 scores: [ 0.67  0.37  0.54  0.47]
Score :
f1 macro : 51.0766 %
            f1 weighted : 47.7461 %
acurracy : 51.4512 %
class 4 : 30.2%
Determination of optimal hyperparameters in             11.8 s
Optimal values are {'max_depth': 8, 'n_estimators': 32}
            F1 weighted Score of cross valdation 48.87%
Random Forest, p=68
f1 scores: [ 0.67  0.37  0.54  0.47]
Score :
f1 macro : 51.0766 %
            f1 weighted : 47.7461 %
acurracy : 51.4512 %
cluster 7.0 : 786 elements
Number exemple: 786
        - training set: 628
        - test set: 158
Number of features: p=68
Number of class: 4
class 1 : 2.8%
Determination of optimal hyperparameters in             13.4 s
Optimal values are {'max_depth': 32, 'n_estimators': 64}
            F1 weighted Score of cross valdation 57.17%
Random Forest, p=68
f1 scores: [ 0.    0.63  0.55  0.  ]
Score :
f1 macro : 29.7071 %
            f1 weighted : 53.3956 %
acurracy : 51.4512 %
```

```
class 2 : 45.9%

//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)



Determination of optimal hyperparameters in             12.9 s
Optimal values are {'max_depth': 32, 'n_estimators': 64}
            F1 weighted Score of cross valdation 57.17%
Random Forest, p=68
f1 scores: [ 0.    0.63  0.55  0.  ]


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Score :
f1 macro : 29.7071 %
            f1 weighted : 53.3956 %
acurracy : 51.4512 %
class 3 : 43.3%
Determination of optimal hyperparameters in             12.8 s
Optimal values are {'max_depth': 32, 'n_estimators': 64}
            F1 weighted Score of cross valdation 57.17%
Random Forest, p=68
f1 scores: [ 0.    0.63  0.55  0.  ]


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Score :
f1 macro : 29.7071 %
            f1 weighted : 53.3956 %
acurracy : 51.4512 %
class 4 : 8.0%
Determination of optimal hyperparameters in             12.9 s
Optimal values are {'max_depth': 32, 'n_estimators': 64}
            F1 weighted Score of cross valdation 57.17%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.63  0.55  0.  ]
```

```
Score :
f1 macro : 29.7071 %
            f1 weighted : 53.3956 %
acurracy : 51.4512 %

Analysis cluster method clust5
liste of clusters : [ 2.  5.]
cluster 2.0 : 1064 elements
Number exemple: 1062
        - training set: 849
        - test set: 213
Number of features: p=68
Number of class: 4
class 1 : 1.3%
Determination of optimal hyperparameters in             15.7 s
Optimal values are {'max_depth': 32, 'n_estimators': 128}
            F1 weighted Score of cross valdation 49.71%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.58  0.67  0.18]
Score :
f1 macro : 35.7590 %
            f1 weighted : 55.1294 %
acurracy : 51.4512 %
class 2 : 33.3%
Determination of optimal hyperparameters in             14.5 s
Optimal values are {'max_depth': 32, 'n_estimators': 128}
            F1 weighted Score of cross valdation 49.71%


//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)


Random Forest, p=68
f1 scores: [ 0.    0.58  0.67  0.18]
Score :
f1 macro : 35.7590 %
            f1 weighted : 55.1294 %
acurracy : 51.4512 %
class 3 : 46.9%
Determination of optimal hyperparameters in             14.5 s
Optimal values are {'max_depth': 32, 'n_estimators': 128}
```

```
            F1 weighted Score of cross valdation 49.71%

//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)

Random Forest, p=68
f1 scores: [ 0.    0.58  0.67  0.18]
Score :
f1 macro : 35.7590 %
            f1 weighted : 55.1294 %
acurracy : 51.4512 %
class 4 : 18.5%
Determination of optimal hyperparameters in             13.8 s
Optimal values are {'max_depth': 32, 'n_estimators': 128}
            F1 weighted Score of cross valdation 49.71%

//anaconda/envs/py36/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Undefin
  'precision', 'predicted', average, warn_for)

Random Forest, p=68
f1 scores: [ 0.    0.58  0.67  0.18]
Score :
f1 macro : 35.7590 %
            f1 weighted : 55.1294 %
acurracy : 51.4512 %
cluster 5.0 : 972 elements
Number exemple: 971
        - training set: 776
        - test set: 195
Number of features: p=68
Number of class: 4
class 1 : 3.6%
Determination of optimal hyperparameters in             13.8 s
Optimal values are {'max_depth': 8, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.93%
Random Forest, p=68
f1 scores: [ 0.36  0.59  0.55  0.4 ]
Score :
f1 macro : 47.4979 %
            f1 weighted : 53.5142 %
acurracy : 51.4512 %
class 2 : 42.8%
Determination of optimal hyperparameters in             13.5 s
Optimal values are {'max_depth': 8, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.93%
```

```
Random Forest, p=68
f1 scores: [ 0.36  0.59  0.55  0.4 ]
Score :
f1 macro : 47.4979 %
            f1 weighted : 53.5142 %
acurracy : 51.4512 %
class 3 : 40.6%
Determination of optimal hyperparameters in             14.0 s
Optimal values are {'max_depth': 8, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.93%
Random Forest, p=68
f1 scores: [ 0.36  0.59  0.55  0.4 ]
Score :
f1 macro : 47.4979 %
            f1 weighted : 53.5142 %
acurracy : 51.4512 %
class 4 : 13.0%
Determination of optimal hyperparameters in             17.5 s
Optimal values are {'max_depth': 8, 'n_estimators': 128}
            F1 weighted Score of cross valdation 55.93%
Random Forest, p=68
f1 scores: [ 0.36  0.59  0.55  0.4 ]
Score :
f1 macro : 47.4979 %
            f1 weighted : 53.5142 %
acurracy : 51.4512 %
```