KNN - one-label vs rest with confusion matrix

Attribute Information

- STG (The degree of study time for goal object materails), (input value)
- SCG (The degree of repetition number of user for goal object materails) (input value)
- STR (The degree of study time of user for related objects with goal object) (input value)
- LPR (The exam performance of user for related objects with goal object) (input value)
- PEG (The exam performance of user for goal objects) (input value)
- UNS (The knowledge level of user) (target value)
  - Very Low: 50
  - Low:129
  - Middle: 122
  - High 130

In [130]:
```
% matplotlib inline

from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))
import numpy as np
import pandas as pd
pd.set_option('display.max_columns', 100)

df = pd.read_csv('User_Knowledge.csv')

df.loc[df.UNS == 'very_low','grade'] = 0
df.loc[df.UNS == 'Low','grade'] = 1
df.loc[df.UNS == 'Middle','grade'] = 2
df.loc[df.UNS == 'High','grade'] = 3

df.loc[df.UNS == 'very_low','vlow'] = 1
df.loc[df.UNS == 'Low','low'] = 1
df.loc[df.UNS == 'Middle','mid'] = 1
df.loc[df.UNS == 'High','high'] = 1

df.fillna(0, inplace=True)
df.sample(5)
```

Out[130]:

	STG	SCG	STR	LPR	PEG	UNS	grade	vlow	low	mid	high
239	0.520	0.44	0.82	0.30	0.52	Middle	2.0	0.0	0.0	1.0	0.0
254	0.780	0.61	0.71	0.19	0.60	Middle	2.0	0.0	0.0	1.0	0.0
87	0.270	0.31	0.32	0.41	0.28	Low	1.0	0.0	1.0	0.0	0.0
190	0.445	0.70	0.82	0.16	0.64	Middle	2.0	0.0	0.0	1.0	0.0
256	0.500	0.75	0.81	0.61	0.26	Middle	2.0	0.0	0.0	1.0	0.0

In [131]:
```
# defining different labels
y1 = df.vlow
y2 = df.low
y3 = df.mid
y4 = df.high

# feature selection, dropping SCG
X = df.drop(columns=['SCG']).iloc[:,0:4]

X.sample()
```

Out[131]:

	STG	STR	LPR	PEG
20	0.12	0.2	0.78	0.2

In [132]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train, X_test, y_train, y_test = train_test_split(X, y1, test_size=0.33, random_st
ate=0)
sc = StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

KNN one label

In [133]:
```
# Classifier implementing the k-nearest neighbors vote

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn_pca = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)
y_pred=knn.predict(X_test)

print("KNN Classfier (y1)",knn.score(X_test,y_test))
```

```
KNN Classfier (y1) 0.976744186047
```

In [134]:
```
from sklearn.metrics import confusion_matrix
label = ['very_low', 'Low','Middle','High']

confusion = confusion_matrix(y_test, y_pred)
print(confusion)

TP = confusion[1, 1]
TN = confusion[0, 0]
FP = confusion[0, 1]
FN = confusion[1, 0]

accuracy = (TP+TN)/(TP+TN+FP+FN)
sensitivity = TP/(FN+TP)
specificity = TN/(TN+FP)
false_p = FP/(FP+TN)
precision = TP/(FP+TP)

print("\nAccuracy: %f\nSensitivty: %f\nSpecificity: %f\nFalse Positive: %f\nPrecisio
n: %f" % (accuracy,sensitivity,specificity,false_p,precision))
```

```
[[79  0]
 [ 2  5]]

Accuracy: 0.976744
Sensitivty: 0.714286
Specificity: 1.000000
False Positive: 0.000000
Precision: 1.000000
```

KNN (PCA transformed)

In [135]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

knn_pca.fit(X_train_pca, y_train)
y_pred=knn_pca.predict(X_test_pca)

print("MLP (PCA transformed) accuracy (y1): ", knn_pca.score(X_test_pca, y_test))
```

```
MLP (PCA transformed) accuracy (y1):  0.918604651163
```

In [136]:
```
from sklearn.metrics import confusion_matrix
label = ['very_low', 'Low','Middle','High']

confusion_matrix(y_test, y_pred)
print(confusion)

TP = confusion[1, 1]
TN = confusion[0, 0]
FP = confusion[0, 1]
FN = confusion[1, 0]

accuracy = (TP+TN)/(TP+TN+FP+FN)
sensitivity = TP/(FN+TP)
specificity = TN/(TN+FP)
false_p = FP/(FP+TN)
precision = TP/(FP+TP)

print("\nAccuracy: %f\nSensitivty: %f\nSpecificity: %f\nFalse Positive: %f\nPrecisio
n: %f" % (accuracy,sensitivity,specificity,false_p,precision))
```

```
[[79  0]
 [ 2  5]]

Accuracy: 0.976744
Sensitivty: 0.714286
Specificity: 1.000000
False Positive: 0.000000
Precision: 1.000000
```

against y2. Grade = low

In [137]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y2, test_size=0.33, random_st
ate=0)
sc = StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)

# KNN
# Classifier implementing the k-nearest neighbors vote

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn_pca = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)
y_pred=knn.predict(X_test)

print("KNN Classfier (y2)",knn.score(X_test,y_test))
```

```
KNN Classfier (y2) 0.953488372093
```

KNN (PCA transformed)

In [138]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

knn_pca.fit(X_train_pca, y_train)
y_pred=knn_pca.predict(X_test_pca)

print("MLP (PCA transformed) accuracy (y2): ", knn_pca.score(X_test_pca, y_test))
```

```
MLP (PCA transformed) accuracy (y2):  0.790697674419
```

against y3. Grade = Mid

In [139]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y3, test_size=0.33, random_st
ate=0)
sc = StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)

# KNN
# Classifier implementing the k-nearest neighbors vote

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn_pca = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)
y_pred=knn.predict(X_test)

print("KNN Classfier (y3)",knn.score(X_test,y_test))
```

```
KNN Classfier (y3) 0.976744186047
```

KNN (PCA transformed)

In [140]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

knn_pca.fit(X_train_pca, y_train)
y_pred=knn_pca.predict(X_test_pca)

print("MLP (PCA transformed) accuracy (y3): ", knn_pca.score(X_test_pca, y_test))
```

```
MLP (PCA transformed) accuracy (y3):  0.639534883721
```

against y4. Grade = High

In [141]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y4, test_size=0.33, random_st
ate=0)
sc = StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)

# KNN
# Classifier implementing the k-nearest neighbors vote

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn_pca = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)
y_pred=knn.predict(X_test)

print("KNN Classfier (y4)",knn.score(X_test,y_test))
```

```
KNN Classfier (y4) 0.988372093023
```

KNN (PCA transformed)

In [142]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

knn_pca.fit(X_train_pca, y_train)
y_pred=knn_pca.predict(X_test_pca)

print("MLP (PCA transformed) accuracy (y4): ", knn_pca.score(X_test_pca, y_test))
```

```
MLP (PCA transformed) accuracy (y4):  0.697674418605
```