MLP

In [30]:
```
% 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.sample(5)
```

Out[30]:

	STG	SCG	STR	LPR	PEG	UNS	grade
195	0.550	0.100	0.27	0.25	0.29	Low	1.0
227	0.580	0.348	0.06	0.29	0.31	Low	1.0
95	0.255	0.305	0.86	0.62	0.15	Low	1.0
192	0.370	0.600	0.77	0.40	0.50	Middle	2.0
85	0.248	0.300	0.31	0.20	0.03	very_low	0.0

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 [31]:
```
y = list(df['UNS'])
y_grade = df.grade

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

X.sample()
```

Out[31]:

	STG	STR	LPR	PEG
211	0.8	0.06	0.31	0.51

In [32]:
```
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, y, test_size=0.33, random_
state=0)
sc = StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

MLP

In [33]:
```
from sklearn.neural_network import MLPClassifier

MLP_clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(10,5), rando
m_state=1)

MLP_clf = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(7,2), random_
state=1)
MLP_clf_pca = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(7,2), ran
dom_state=1)

MLP_clf.fit(X_train,y_train)
#y_pred=clf.predict(X_test)

#print("MLP accuracy :",metrics.accuracy_score(y_test, y_pred))
```

Out[33]:
```
MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(7, 2), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)
```

In [34]:
```
print("MLP accuracy: ",MLP_clf.score(X_test,y_test))
```

```
MLP accuracy:  0.93023255814
```

PCA + MLP approach

In [35]:
```
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)

MLP_clf_pca.fit(X_train_pca, y_train)
print("MLP (PCA transformed) accuracy: ", MLP_clf_pca.score(X_test_pca, y_test))
```

```
MLP (PCA transformed) accuracy:  0.604651162791
```