Logistic Regression

In [19]:
```
% 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[19]:

	STG	SCG	STR	LPR	PEG	UNS	grade
191	0.42	0.700	0.72	0.30	0.80	High	3.0
220	0.61	0.258	0.56	0.62	0.24	Low	1.0
136	0.38	0.100	0.40	0.48	0.26	Low	1.0
60	0.09	0.610	0.53	0.75	0.01	Low	1.0
208	0.55	0.170	0.71	0.48	0.11	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 [20]:
```
#y = list(df['UNS'])
y = df.grade

# feature selection, dropping SCG
#X = df.drop(columns=['SCG']).iloc[:,0:4]
# keeping all features
X = df.iloc[:,0:5]

X.sample()
```

Out[20]:

	STG	SCG	STR	LPR	PEG
184	0.38	0.59	0.31	0.62	0.2

In [21]:
```
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_sta
te=0)

# separating transformed
sc = StandardScaler()
X_train_sc=sc.fit_transform(X_train)
X_test_sc=sc.transform(X_test)
```

Linear regression

In [22]:
```
from sklearn.linear_model import LogisticRegression

reg = LogisticRegression()
reg_pca = LogisticRegression()

reg.fit(X_train, y_train)
reg_y_pred=reg.predict(X_test)
print("Logistic Regression accuracy: ", reg.score(X_test, y_test))
```

```
Logistic Regression accuracy:  0.720930232558
```

In [23]:
```
from sklearn import metrics
print("MAE: ", metrics.mean_absolute_error(y_test, reg_y_pred))
print("MSE: ", metrics.mean_squared_error(y_test, reg_y_pred))
print("RMSE: ", np.sqrt(metrics.mean_squared_error(y_test, reg_y_pred)))
```

```
MAE:  0.279069767442
MSE:  0.279069767442
RMSE:  0.528270543795
```

Linear regression + SVC approach

In [24]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_sc)
X_test_pca = pca.transform(X_test_sc)

reg_pca.fit(X_train_pca, y_train)
reg_y_pred_pca = reg_pca.predict(X_test_pca)
print("Logistic Regression (with PCA) accuracy: ", reg_pca.score(X_test_pca, y_test
))
```

```
Logistic Regression (with PCA) accuracy:  0.546511627907
```

MAE, MSE, RMSE

In [25]:
```
from sklearn import metrics
print("MAE: ", metrics.mean_absolute_error(y_test, reg_y_pred_pca))
print("MSE: ", metrics.mean_squared_error(y_test, reg_y_pred_pca))
print("RMSE: ", np.sqrt(metrics.mean_squared_error(y_test, reg_y_pred_pca)))
```

```
MAE:  0.476744186047
MSE:  0.523255813953
RMSE:  0.723364233256
```

Feature selection. Dropping SCG to see whether RMSE improves

In [26]:
```
# feature selection, dropping SCG to see whether RMSE improves
X = df.drop(columns=['SCG']).iloc[:,0:4]

X.sample()

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_sta
te=0)

# separating transformed
sc = StandardScaler()
X_train_sc=sc.fit_transform(X_train)
X_test_sc=sc.transform(X_test)

# Linear regression
from sklearn.linear_model import LinearRegression

reg = LinearRegression()
reg_pca = LinearRegression()

reg.fit(X_train, y_train)
reg_y_pred=reg.predict(X_test)
print("Logistic Regression accuracy: ", reg.score(X_test, y_test))

from sklearn import metrics
print("MAE: ", metrics.mean_absolute_error(y_test, reg_y_pred))
print("MSE: ", metrics.mean_squared_error(y_test, reg_y_pred))
print("RMSE: ", np.sqrt(metrics.mean_squared_error(y_test, reg_y_pred)))

# Linear regression + SVC approach
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_sc)
X_test_pca = pca.transform(X_test_sc)

reg_pca.fit(X_train_pca, y_train)
reg_y_pred_pca = reg_pca.predict(X_test_pca)
print("Logistic Regression (with PCA) accuracy: ", reg_pca.score(X_test_pca, y_test
))

from sklearn import metrics
print("MAE: ", metrics.mean_absolute_error(y_test, reg_y_pred_pca))
print("MSE: ", metrics.mean_squared_error(y_test, reg_y_pred_pca))
print("RMSE: ", np.sqrt(metrics.mean_squared_error(y_test, reg_y_pred_pca)))
```

```
Logistic Regression accuracy:  0.932686320712
MAE:  0.169140096604
MSE:  0.0551997653979
RMSE:  0.234946303223
Logistic Regression (with PCA) accuracy:  0.477349803784
MAE:  0.546350442153
MSE:  0.428592947546
RMSE:  0.65467010589
```

Logistic Regression with 4 columns increases significantly. However Logistic Regression with PCA decreases.