In [10]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers
import keras as keras
from keras import models
from keras import layers
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import SGD
from keras.utils.np_utils import to_categorical
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from sklearn.neural_network import MLPClassifier
from keras import metrics
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, log_lo
```

In [2]:

```
red = pd.read_csv('data/winequality-red1.csv')
white = pd.read_csv('data/winequality-white1.csv')
```

In [3]:

```
red['quality'].replace(to_replace=[0,1,2,3,4,5], value=1, inplace=True)
red['quality'].replace(to_replace=[6], value=2, inplace=True)
red['quality'].replace(to_replace=[7,8,9,10], value=3, inplace=True)
X_red = red[
['fixed acidity',
 'volatile acidity',
 'citric acid',
 'residual sugar',
 'chlorides',
 'free sulfur dioxide',
 'total sulfur dioxide',
 'density',
 'pH',
 'sulphates',
 'alcohol']]
Y_red = red[['quality']]
white['quality'].replace(to_replace=[0,1,2,3,4,5], value=1, inplace=True)
white['quality'].replace(to_replace=[6], value=2, inplace=True)
white['quality'].replace(to_replace=[7,8,9,10], value=3, inplace=True)
X_white = white[
['fixed acidity',
 'volatile acidity',
 'citric acid',
 'residual sugar',
 'chlorides',
 'free sulfur dioxide',
 'total sulfur dioxide',
 'density',
 'pH',
 'sulphates',
 'alcohol']]

Y_white = white[['quality']]
X_red = X_red.values
Y_red = Y_red.values
X_white = X_white.values
Y_white = Y_white.values
number_of_features = 11
```

In [41]:

```
def rsme(targets, outputs):
    return tf.sqrt(tf.reduce_mean(tf.square(tf.subtract(targets, outputs))))

# Create function returning a compiled network
def create_network():

    # Start neural network
    network = models.Sequential()

    network.add(layers.Dense(units=6, activation='linear', input_shape=(number_of_features,
    #network.add(layers.Dense(units=10, activation='linear'))
    #network.add(layers.BatchNormalization())
    #network.add(layers.Dense(units=4, activation='linear'))
    #network.add(layers.Dense(units=7, activation='linear'))
    #network.add(layers.Dense(units=4, activation='linear'))
    #network.add(layers.Dense(units=5, activation='linear'))
    #network.add(layers.Dense(units=6, activation='linear'))
    #network.add(layers.Dense(units=4, activation='linear'))
    #network.add(layers.Dense(units=32, activation='selu'))

    network.add(layers.Dense(3, activation='softmax'))

    # Compile neural network
    #network.compile(loss='binary_crossentropy', # Cross-entropy
   #                 optimizer='rmsprop', # Root Mean Square Propagation
    #                metrics=['accuracy']) # Accuracy performance metric
    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])

    # Return compiled network
    return network
```

In [42]:

```
neural_network = KerasClassifier(build_fn=create_network,
                                 epochs=10,
                                 batch_size=100,
                                 verbose=0)
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
```

In [23]:

```
#four hidden layers one as BatchNormalize 64units h1, 10units h2, bn h3, 32units h4, 3 soft
print("Red")
prediction_red = cross_val_predict(neural_network, X_red, Y_red, cv=cv)
print(classification_report(Y_red, prediction_red))
print("White")
prediction_white = cross_val_predict(neural_network, X_white, Y_white, cv=cv)
print(classification_report(Y_white, prediction_white))
```

```
Red
             precision    recall  f1-score   support

          1       0.68      0.55      0.61       744
          2       0.47      0.56      0.51       638
          3       0.39      0.43      0.41       217

avg / total       0.56      0.54      0.54      1599

White
             precision    recall  f1-score   support

          1       0.58      0.51      0.54      1640
          2       0.50      0.62      0.55      2198
          3       0.47      0.31      0.37      1060

avg / total       0.52      0.52      0.51      4898
```

In [26]:

```
# one selu 64 nodes, bn, softmax3
print("Red")
prediction_red = cross_val_predict(neural_network, X_red, Y_red, cv=cv)
print(classification_report(Y_red, prediction_red))
print("White")
prediction_white = cross_val_predict(neural_network, X_white, Y_white, cv=cv)
print(classification_report(Y_white, prediction_white))
```

```
Red
             precision    recall  f1-score   support

          1       0.71      0.57      0.63       744
          2       0.50      0.63      0.55       638
          3       0.41      0.38      0.40       217

avg / total       0.59      0.57      0.57      1599

White
             precision    recall  f1-score   support

          1       0.59      0.62      0.60      1640
          2       0.54      0.56      0.55      2198
          3       0.52      0.43      0.47      1060

avg / total       0.55      0.55      0.55      4898
```

In [29]:

```
# one sigmoid 64 nodes, bn, softmax3
print("Red")
prediction_red = cross_val_predict(neural_network, X_red, Y_red, cv=cv)
print(classification_report(Y_red, prediction_red))
print("White")
prediction_white = cross_val_predict(neural_network, X_white, Y_white, cv=cv)
print(classification_report(Y_white, prediction_white))
```

```
Red
             precision    recall  f1-score   support

          1       0.60      0.77      0.67       744
          2       0.49      0.40      0.44       638
          3       0.43      0.26      0.32       217

avg / total       0.53      0.55      0.53      1599

White
             precision    recall  f1-score   support

          1       0.55      0.53      0.54      1640
          2       0.50      0.66      0.57      2198
          3       0.56      0.21      0.30      1060

avg / total       0.53      0.52      0.50      4898
```

In [34]:

```
# a lot of linears, nodes, linear again, bn, softmax3
print("Red")
prediction_red = cross_val_predict(neural_network, X_red, Y_red, cv=cv)
print(classification_report(Y_red, prediction_red))
print("White")
prediction_white = cross_val_predict(neural_network, X_white, Y_white, cv=cv)
print(classification_report(Y_white, prediction_white))
```

```
Red
             precision    recall  f1-score   support

          1       0.63      0.67      0.65       744
          2       0.47      0.52      0.49       638
          3       0.32      0.17      0.22       217

avg / total       0.52      0.54      0.53      1599

White
             precision    recall  f1-score   support

          1       0.45      0.65      0.53      1640
          2       0.49      0.26      0.34      2198
          3       0.32      0.41      0.36      1060

avg / total       0.44      0.43      0.41      4898
```

In [37]:

```
# a lot of linears, softmax3
print("Red")
prediction_red = cross_val_predict(neural_network, X_red, Y_red, cv=cv)
print(classification_report(Y_red, prediction_red))
print("White")
prediction_white = cross_val_predict(neural_network, X_white, Y_white, cv=cv)
print(classification_report(Y_white, prediction_white))
```

```
Red
             precision    recall  f1-score   support

          1       0.59      0.60      0.59       744
          2       0.44      0.58      0.50       638
          3       0.00      0.00      0.00       217

avg / total       0.45      0.51      0.48      1599

White
             precision    recall  f1-score   support

          1       0.51      0.29      0.37      1640
          2       0.47      0.69      0.56      2198
          3       0.28      0.20      0.23      1060

avg / total       0.44      0.45      0.42      4898
```

In [40]:

```
# two linears, softmax3
print("Red")
prediction_red = cross_val_predict(neural_network, X_red, Y_red, cv=cv)
print(classification_report(Y_red, prediction_red))
print("White")
prediction_white = cross_val_predict(neural_network, X_white, Y_white, cv=cv)
print(classification_report(Y_white, prediction_white))
```

```
Red
             precision    recall  f1-score   support

          1       0.59      0.56      0.57       744
          2       0.44      0.61      0.51       638
          3       0.00      0.00      0.00       217

avg / total       0.45      0.50      0.47      1599

White
             precision    recall  f1-score   support

          1       0.48      0.40      0.44      1640
          2       0.47      0.65      0.55      2198
          3       0.45      0.19      0.27      1060

avg / total       0.47      0.47      0.45      4898
```

In [43]:

```
# one linear, softmax3
print("Red")
prediction_red = cross_val_predict(neural_network, X_red, Y_red, cv=cv)
print(classification_report(Y_red, prediction_red))
print("White")
prediction_white = cross_val_predict(neural_network, X_white, Y_white, cv=cv)
print(classification_report(Y_white, prediction_white))
```

```
Red
             precision    recall  f1-score   support

          1       0.50      0.56      0.53       744
          2       0.42      0.41      0.41       638
          3       0.17      0.11      0.13       217

avg / total       0.42      0.44      0.43      1599

White
             precision    recall  f1-score   support

          1       0.42      0.29      0.34      1640
          2       0.46      0.67      0.54      2198
          3       0.33      0.15      0.21      1060

avg / total       0.42      0.43      0.40      4898
```

In [22]:

```
print("RSME red")
print(sqrt(np.mean((Y_red-prediction_red)*(Y_red-prediction_red))))
print('RSME white')
print(sqrt(np.mean((Y_white-prediction_white)*(Y_white-prediction_white))))
```

```
RSME red
0.9614523774621145
RSME white
0.9699337529605507
```

In [ ]: