In [11]:

```
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
```

In [12]:

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

In [13]:

```
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 [79]:

```
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():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='relu'))
    network.add(layers.Dense(3, activation='softmax'))

    #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

cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
```

In [80]:

```
neural_network = KerasClassifier(build_fn=create_network,
                                 epochs=10,
                                 batch_size=100,
                                 verbose=0)
```

In [75]:

```
print("BN, One layer Linear with 64 nodes and Softmax")
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))
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))))
```

```
BN, One layer Linear with 64 nodes and Softmax
Red
              precision    recall  f1-score   support

           1       0.70      0.79      0.74       744
           2       0.55      0.54      0.54       638
           3       0.51      0.32      0.39       217

   micro avg       0.63      0.63      0.63      1599
   macro avg       0.59      0.55      0.56      1599
weighted avg       0.62      0.63      0.62      1599

White
              precision    recall  f1-score   support

           1       0.64      0.57      0.60      1640
           2       0.53      0.67      0.59      2198
           3       0.56      0.35      0.43      1060

   micro avg       0.56      0.56      0.56      4898
   macro avg       0.58      0.53      0.54      4898
weighted avg       0.57      0.56      0.56      4898

RSME red
0.9594572166570099
RSME white
0.9712627605243774
```

In [78]:

```
print("BN, One layer Sigmoid with 64 nodes and Softmax")
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))
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))))
```

```
BN, One layer Sigmoid with 64 nodes and Softmax
Red
              precision    recall  f1-score   support

           1       0.65      0.81      0.73       744
           2       0.51      0.53      0.52       638
           3       0.69      0.04      0.08       217

   micro avg       0.60      0.60      0.60      1599
   macro avg       0.62      0.46      0.44      1599
weighted avg       0.60      0.60      0.56      1599

White
              precision    recall  f1-score   support

           1       0.65      0.56      0.60      1640
           2       0.53      0.71      0.61      2198
           3       0.60      0.31      0.41      1060

   micro avg       0.57      0.57      0.57      4898
   macro avg       0.59      0.53      0.54      4898
weighted avg       0.59      0.57      0.56      4898

RSME red
0.9010400202041945
RSME white
0.9538702544931809
```

In [81]:

```
print("BN, One layer Relu with 64 nodes and Softmax")
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))
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))))
```

```
BN, One layer Relu with 64 nodes and Softmax
Red
              precision    recall  f1-score   support

           1       0.68      0.79      0.73       744
           2       0.54      0.53      0.54       638
           3       0.56      0.30      0.40       217

   micro avg       0.62      0.62      0.62      1599
   macro avg       0.60      0.54      0.55      1599
weighted avg       0.61      0.62      0.61      1599

White
              precision    recall  f1-score   support

           1       0.66      0.61      0.63      1640
           2       0.55      0.68      0.61      2198
           3       0.61      0.37      0.46      1060

   micro avg       0.59      0.59      0.59      4898
   macro avg       0.60      0.55      0.57      4898
weighted avg       0.60      0.59      0.58      4898

RSME red
0.951804017566414
RSME white
0.9747912231867222
```

In [60]:

```
print("BN, One layer Selu with 64 nodes and Softmax")
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))
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))))
```

```
One layer Selu with 64 nodes and Softmax
Red
              precision    recall  f1-score   support

           1       0.58      0.66      0.62       744
           2       0.45      0.50      0.48       638
           3       0.37      0.06      0.11       217

   micro avg       0.52      0.52      0.52      1599
   macro avg       0.47      0.41      0.40      1599
weighted avg       0.50      0.52      0.49      1599

White
              precision    recall  f1-score   support

           1       0.50      0.55      0.52      1640
           2       0.48      0.54      0.50      2198
           3       0.46      0.26      0.33      1060

   micro avg       0.48      0.48      0.48      4898
   macro avg       0.48      0.45      0.45      4898
weighted avg       0.48      0.48      0.47      4898

RSME red
0.9073693082714352
RSME white
0.9940223760963623
```

In [ ]: