In [1]:

```
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
from copy import deepcopy
```

```
Using TensorFlow backend.
```

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 [4]:

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

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

# Create function returning a compiled network
def create_network():
    network = models.Sequential()
    #network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='linear'))
    #network.add(layers.Dense(units=9, activation = 'relu'))
    #network.add(layers.Dense(units=6, activation = 'relu'))
    #network.add(layers.Dense(units=18, 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


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

Start strategically exploration

In [5]:

```
def create_network1():
    network = models.Sequential()
    network.add(layers.Dense(units=64, activation='linear'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network1,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("One Layer Linear, 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 Linear, Softmax
Red
              precision    recall  f1-score   support

           1       0.66      0.81      0.73       744
           2       0.53      0.50      0.52       638
           3       0.51      0.22      0.30       217

   micro avg       0.60      0.60      0.60      1599
   macro avg       0.57      0.51      0.52      1599
weighted avg       0.59      0.60      0.59      1599

White
              precision    recall  f1-score   support

           1       0.51      0.65      0.57      1640
           2       0.50      0.44      0.47      2198
           3       0.39      0.33      0.36      1060

   micro avg       0.49      0.49      0.49      4898
   macro avg       0.47      0.47      0.47      4898
weighted avg       0.48      0.49      0.48      4898

RSME red
0.9432747904760156
RSME white
1.0494466486761753
```

In [6]:

```
def create_network2():
    network = models.Sequential()
    network.add(layers.Dense(units=64, activation='sigmoid'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network2,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("One Layer Sigmoid, 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 Sigmoid, Softmax
Red
              precision    recall  f1-score   support

           1       0.67      0.78      0.72       744
           2       0.52      0.53      0.53       638
           3       0.55      0.21      0.30       217

   micro avg       0.60      0.60      0.60      1599
   macro avg       0.58      0.51      0.52      1599
weighted avg       0.59      0.60      0.59      1599

White
              precision    recall  f1-score   support

           1       0.60      0.65      0.63      1640
           2       0.54      0.60      0.57      2198
           3       0.56      0.37      0.44      1060

   micro avg       0.57      0.57      0.57      4898
   macro avg       0.57      0.54      0.55      4898
weighted avg       0.57      0.57      0.56      4898

RSME red
0.9332164612306332
RSME white
1.002574092556401
```

In [7]:

```
def create_network3():
    network = models.Sequential()
    network.add(layers.Dense(units=64, activation='relu'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network3,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("One Layer Relu, 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 Relu, Softmax
Red
              precision    recall  f1-score   support

           1       0.66      0.77      0.71       744
           2       0.52      0.50      0.51       638
           3       0.51      0.29      0.37       217

   micro avg       0.59      0.59      0.59      1599
   macro avg       0.56      0.52      0.53      1599
weighted avg       0.58      0.59      0.58      1599

White
              precision    recall  f1-score   support

           1       0.61      0.57      0.59      1640
           2       0.52      0.65      0.58      2198
           3       0.50      0.28      0.36      1060

   micro avg       0.55      0.55      0.55      4898
   macro avg       0.54      0.50      0.51      4898
weighted avg       0.55      0.55      0.54      4898

RSME red
0.954875197479647
RSME white
0.9697571078030033
```

In [8]:

```
def create_network4():
    network = models.Sequential()
    network.add(layers.Dense(units=64, activation='selu'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network4,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("One Layer Selu, 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, Softmax
Red
              precision    recall  f1-score   support

           1       0.69      0.79      0.73       744
           2       0.54      0.52      0.53       638
           3       0.52      0.32      0.40       217

   micro avg       0.62      0.62      0.62      1599
   macro avg       0.58      0.54      0.55      1599
weighted avg       0.61      0.62      0.61      1599

White
              precision    recall  f1-score   support

           1       0.60      0.58      0.59      1640
           2       0.51      0.61      0.56      2198
           3       0.49      0.32      0.39      1060

   micro avg       0.54      0.54      0.54      4898
   macro avg       0.54      0.51      0.51      4898
weighted avg       0.54      0.54      0.53      4898

RSME red
0.9592790611636349
RSME white
0.9861262931045918
```

In [9]:

```
def create_network5():
    network = models.Sequential()
    network.add(layers.Dense(units=64, activation='tanh'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network5,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("One Layer Tanh, 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 Tanh, Softmax
Red
              precision    recall  f1-score   support

           1       0.68      0.76      0.72       744
           2       0.54      0.53      0.54       638
           3       0.53      0.32      0.40       217

   micro avg       0.61      0.61      0.61      1599
   macro avg       0.58      0.54      0.55      1599
weighted avg       0.60      0.61      0.60      1599

White
              precision    recall  f1-score   support

           1       0.60      0.67      0.63      1640
           2       0.54      0.61      0.57      2198
           3       0.57      0.31      0.40      1060

   micro avg       0.56      0.56      0.56      4898
   macro avg       0.57      0.53      0.53      4898
weighted avg       0.56      0.56      0.55      4898

RSME red
0.9566088800945204
RSME white
0.9911319404022657
```

In [10]:

```
def create_network6():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='linear'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network6,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Linear, 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))))
```

```
Batch Normalize Layer, one Layer Linear, Softmax
Red
              precision    recall  f1-score   support

           1       0.71      0.77      0.74       744
           2       0.55      0.56      0.56       638
           3       0.55      0.35      0.43       217

   micro avg       0.63      0.63      0.63      1599
   macro avg       0.60      0.56      0.57      1599
weighted avg       0.62      0.63      0.62      1599

White
              precision    recall  f1-score   support

           1       0.65      0.57      0.61      1640
           2       0.53      0.69      0.60      2198
           3       0.58      0.34      0.43      1060

   micro avg       0.57      0.57      0.57      4898
   macro avg       0.59      0.53      0.55      4898
weighted avg       0.58      0.57      0.57      4898

RSME red
0.9579679687671662
RSME white
0.9658236833668847
```

In [11]:

```
def create_network7():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='sigmoid'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network7,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Sigmoid, 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))))
```

```
Batch Normalize Layer, one Layer Sigmoid, Softmax
Red
              precision    recall  f1-score   support

           1       0.71      0.78      0.74       744
           2       0.55      0.55      0.55       638
           3       0.54      0.36      0.43       217

   micro avg       0.63      0.63      0.63      1599
   macro avg       0.60      0.56      0.58      1599
weighted avg       0.62      0.63      0.62      1599

White
              precision    recall  f1-score   support

           1       0.66      0.62      0.64      1640
           2       0.55      0.66      0.60      2198
           3       0.59      0.41      0.48      1060

   micro avg       0.59      0.59      0.59      4898
   macro avg       0.60      0.56      0.57      4898
weighted avg       0.59      0.59      0.59      4898

RSME red
0.9636043539000998
RSME white
0.9897264319519342
```

In [12]:

```
def create_network8():
    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='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network8,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Relu, 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))))
```

```
Batch Normalize Layer, one Layer Relu, Softmax
Red
              precision    recall  f1-score   support

           1       0.71      0.78      0.74       744
           2       0.58      0.58      0.58       638
           3       0.59      0.42      0.49       217

   micro avg       0.65      0.65      0.65      1599
   macro avg       0.63      0.59      0.61      1599
weighted avg       0.64      0.65      0.64      1599

White
              precision    recall  f1-score   support

           1       0.68      0.63      0.66      1640
           2       0.57      0.66      0.61      2198
           3       0.60      0.48      0.53      1060

   micro avg       0.61      0.61      0.61      4898
   macro avg       0.62      0.59      0.60      4898
weighted avg       0.62      0.61      0.61      4898

RSME red
0.9672411317413743
RSME white
1.0012813167550536
```

In [15]:

```
def create_network9():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='selu'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network9,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Selu, 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))))
```

```
Batch Normalize Layer, one Layer Selu, Softmax
Red
              precision    recall  f1-score   support

           1       0.72      0.78      0.75       744
           2       0.57      0.58      0.58       638
           3       0.61      0.40      0.48       217

   micro avg       0.65      0.65      0.65      1599
   macro avg       0.63      0.59      0.60      1599
weighted avg       0.64      0.65      0.64      1599

White
              precision    recall  f1-score   support

           1       0.67      0.63      0.65      1640
           2       0.56      0.66      0.61      2198
           3       0.60      0.43      0.50      1060

   micro avg       0.60      0.60      0.60      4898
   macro avg       0.61      0.57      0.59      4898
weighted avg       0.61      0.60      0.60      4898

RSME red
0.9604501230990757
RSME white
0.9911368189199857
```

In [16]:

```
def create_network10():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='tanh'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network10,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Tanh, 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))))
```

```
Batch Normalize Layer, one Layer Tanh, Softmax
Red
              precision    recall  f1-score   support

           1       0.71      0.78      0.74       744
           2       0.57      0.57      0.57       638
           3       0.57      0.38      0.46       217

   micro avg       0.64      0.64      0.64      1599
   macro avg       0.62      0.57      0.59      1599
weighted avg       0.63      0.64      0.63      1599

White
              precision    recall  f1-score   support

           1       0.67      0.65      0.66      1640
           2       0.57      0.64      0.60      2198
           3       0.58      0.45      0.51      1060

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

RSME red
0.9612265798846926
RSME white
1.004153413699297
```

Including Feature Selection

In [17]:

```
## selection from naive bayes notebook
X_red_sel = red[
[#'fixed acidity', -> low importances, thus removed
 'volatile acidity',
 #'citric acid', -> low importances, thus removed
 #'residual sugar', -> low importances, thus removed
 #'chlorides', -> low importances, thus removed
 #'free sulfur dioxide', -> low importances, thus removed
 'total sulfur dioxide',
 'density',
 #'pH', -> low importances, thus removed
 'sulphates',
 'alcohol']]
Y_red_sel = red[['quality']]
X_white_sel = white[
['fixed acidity',
 'volatile acidity',
 #'citric acid', -> low importances, thus removed
 'residual sugar',
 #'chlorides', -> low importances, thus removed
 'free sulfur dioxide',
 'total sulfur dioxide',
 'density',
 #'pH', -> low importances, thus removed
 #'sulphates', -> low importances, thus removed
 'alcohol']]

Y_white_sel = white[['quality']]
X_red_sel = X_red_sel.values
Y_red_sel = Y_red_sel.values
X_white_sel = X_white_sel.values
Y_white_sel = Y_white_sel.values
```

In [18]:

```
neural_network = KerasClassifier(build_fn=create_network6,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Linaer, Softmax")
print("Red")
prediction_red = cross_val_predict(neural_network, X_red_sel, Y_red_sel, cv=cv)
print(classification_report(Y_red_sel, prediction_red))
print("White")
prediction_white = cross_val_predict(neural_network, X_white_sel, Y_white_sel, cv=cv)
print(classification_report(Y_white_sel, prediction_white))
print("RSME red")
print(sqrt(np.mean((Y_red_sel-prediction_red)*(Y_red_sel-prediction_red))))
print('RSME white')
print(sqrt(np.mean((Y_white_sel-prediction_white)*(Y_white_sel-prediction_white))))
```

```
Batch Normalize Layer, one Layer Linaer, Softmax
Red
              precision    recall  f1-score   support

           1       0.69      0.77      0.73       744
           2       0.54      0.55      0.55       638
           3       0.58      0.32      0.41       217

   micro avg       0.62      0.62      0.62      1599
   macro avg       0.60      0.55      0.56      1599
weighted avg       0.62      0.62      0.61      1599

White
              precision    recall  f1-score   support

           1       0.64      0.58      0.61      1640
           2       0.52      0.67      0.59      2198
           3       0.55      0.31      0.40      1060

   micro avg       0.56      0.56      0.56      4898
   macro avg       0.57      0.52      0.53      4898
weighted avg       0.57      0.56      0.56      4898

RSME red
0.9494199654155756
RSME white
0.9656902286037582
```

In [19]:

```
neural_network = KerasClassifier(build_fn=create_network7,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Sigmoid, Softmax")
print("Red")
prediction_red = cross_val_predict(neural_network, X_red_sel, Y_red_sel, cv=cv)
print(classification_report(Y_red_sel, prediction_red))
print("White")
prediction_white = cross_val_predict(neural_network, X_white_sel, Y_white_sel, cv=cv)
print(classification_report(Y_white_sel, prediction_white))
print("RSME red")
print(sqrt(np.mean((Y_red_sel-prediction_red)*(Y_red_sel-prediction_red))))
print('RSME white')
print(sqrt(np.mean((Y_white_sel-prediction_white)*(Y_white_sel-prediction_white))))
```

```
Batch Normalize Layer, one Layer Sigmoid, Softmax
Red
              precision    recall  f1-score   support

           1       0.70      0.78      0.74       744
           2       0.55      0.56      0.56       638
           3       0.60      0.31      0.41       217

   micro avg       0.63      0.63      0.63      1599
   macro avg       0.62      0.55      0.57      1599
weighted avg       0.63      0.63      0.62      1599

White
              precision    recall  f1-score   support

           1       0.66      0.59      0.62      1640
           2       0.54      0.68      0.60      2198
           3       0.59      0.36      0.45      1060

   micro avg       0.58      0.58      0.58      4898
   macro avg       0.60      0.55      0.56      4898
weighted avg       0.59      0.58      0.58      4898

RSME red
0.9461355728573025
RSME white
0.9720562654519462
```

In [20]:

```
neural_network = KerasClassifier(build_fn=create_network8,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Relu, Softmax")
print("Red")
prediction_red = cross_val_predict(neural_network, X_red_sel, Y_red_sel, cv=cv)
print(classification_report(Y_red_sel, prediction_red))
print("White")
prediction_white = cross_val_predict(neural_network, X_white_sel, Y_white_sel, cv=cv)
print(classification_report(Y_white_sel, prediction_white))
print("RSME red")
print(sqrt(np.mean((Y_red_sel-prediction_red)*(Y_red_sel-prediction_red))))
print('RSME white')
print(sqrt(np.mean((Y_white_sel-prediction_white)*(Y_white_sel-prediction_white))))
```

```
Batch Normalize Layer, one Layer Relu, Softmax
Red
              precision    recall  f1-score   support

           1       0.71      0.76      0.73       744
           2       0.56      0.59      0.57       638
           3       0.64      0.39      0.48       217

   micro avg       0.64      0.64      0.64      1599
   macro avg       0.64      0.58      0.60      1599
weighted avg       0.64      0.64      0.64      1599

White
              precision    recall  f1-score   support

           1       0.67      0.64      0.66      1640
           2       0.56      0.67      0.61      2198
           3       0.59      0.40      0.48      1060

   micro avg       0.60      0.60      0.60      4898
   macro avg       0.61      0.57      0.58      4898
weighted avg       0.60      0.60      0.60      4898

RSME red
0.9523657826343893
RSME white
0.9880992898958296
```

In [21]:

```
neural_network = KerasClassifier(build_fn=create_network9,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Selu, Softmax")
print("Red")
prediction_red = cross_val_predict(neural_network, X_red_sel, Y_red_sel, cv=cv)
print(classification_report(Y_red_sel, prediction_red))
print("White")
prediction_white = cross_val_predict(neural_network, X_white_sel, Y_white_sel, cv=cv)
print(classification_report(Y_white_sel, prediction_white))
print("RSME red")
print(sqrt(np.mean((Y_red_sel-prediction_red)*(Y_red_sel-prediction_red))))
print('RSME white')
print(sqrt(np.mean((Y_white_sel-prediction_white)*(Y_white_sel-prediction_white))))
```

```
Batch Normalize Layer, one Layer Selu, Softmax
Red
              precision    recall  f1-score   support

           1       0.71      0.78      0.74       744
           2       0.56      0.58      0.57       638
           3       0.62      0.35      0.45       217

   micro avg       0.64      0.64      0.64      1599
   macro avg       0.63      0.57      0.59      1599
weighted avg       0.64      0.64      0.63      1599

White
              precision    recall  f1-score   support

           1       0.65      0.65      0.65      1640
           2       0.56      0.66      0.60      2198
           3       0.60      0.38      0.47      1060

   micro avg       0.60      0.60      0.60      4898
   macro avg       0.60      0.56      0.57      4898
weighted avg       0.60      0.60      0.59      4898

RSME red
0.9509162253401282
RSME white
0.9884392033174249
```

In [22]:

```
neural_network = KerasClassifier(build_fn=create_network10,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Tanh, Softmax")
print("Red")
prediction_red = cross_val_predict(neural_network, X_red_sel, Y_red_sel, cv=cv)
print(classification_report(Y_red_sel, prediction_red))
print("White")
prediction_white = cross_val_predict(neural_network, X_white_sel, Y_white_sel, cv=cv)
print(classification_report(Y_white_sel, prediction_white))
print("RSME red")
print(sqrt(np.mean((Y_red_sel-prediction_red)*(Y_red_sel-prediction_red))))
print('RSME white')
print(sqrt(np.mean((Y_white_sel-prediction_white)*(Y_white_sel-prediction_white))))
```

```
Batch Normalize Layer, one Layer Tanh, Softmax
Red
              precision    recall  f1-score   support

           1       0.70      0.77      0.74       744
           2       0.55      0.57      0.56       638
           3       0.62      0.37      0.46       217

   micro avg       0.63      0.63      0.63      1599
   macro avg       0.62      0.57      0.59      1599
weighted avg       0.63      0.63      0.63      1599

White
              precision    recall  f1-score   support

           1       0.66      0.64      0.65      1640
           2       0.56      0.65      0.60      2198
           3       0.58      0.40      0.47      1060

   micro avg       0.59      0.59      0.59      4898
   macro avg       0.60      0.56      0.57      4898
weighted avg       0.60      0.59      0.59      4898

RSME red
0.95319314066789
RSME white
0.9926887770921784
```

experimental adding of layers

In [23]:

```
def create_network11():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='relu'))
    network.add(layers.Dense(units=9, activation='relu'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network11,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, two Layer Relu, 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))))
```

```
Batch Normalize Layer, two Layer Relu, Softmax
Red
              precision    recall  f1-score   support

           1       0.72      0.78      0.75       744
           2       0.60      0.59      0.59       638
           3       0.61      0.48      0.54       217

   micro avg       0.66      0.66      0.66      1599
   macro avg       0.65      0.62      0.63      1599
weighted avg       0.66      0.66      0.66      1599

White
              precision    recall  f1-score   support

           1       0.68      0.66      0.67      1640
           2       0.59      0.65      0.62      2198
           3       0.60      0.52      0.56      1060

   micro avg       0.62      0.62      0.62      4898
   macro avg       0.63      0.61      0.62      4898
weighted avg       0.63      0.62      0.62      4898

RSME red
0.9743645940296556
RSME white
1.0152425975564503
```

In [24]:

```
def create_network12():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='linear'))
    network.add(layers.Dense(units=9, activation='relu'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network12,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Linear, one Layer Relu, 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))))
```

```
Batch Normalize Layer, one Layer Linear, one Layer Relu, Softmax
Red
              precision    recall  f1-score   support

           1       0.72      0.78      0.75       744
           2       0.58      0.58      0.58       638
           3       0.59      0.41      0.48       217

   micro avg       0.65      0.65      0.65      1599
   macro avg       0.63      0.59      0.61      1599
weighted avg       0.65      0.65      0.65      1599

White
              precision    recall  f1-score   support

           1       0.67      0.63      0.65      1640
           2       0.56      0.67      0.61      2198
           3       0.61      0.42      0.50      1060

   micro avg       0.60      0.60      0.60      4898
   macro avg       0.61      0.57      0.59      4898
weighted avg       0.61      0.60      0.60      4898

RSME red
0.9645262893258956
RSME white
0.9878382123418706
```

In [25]:

```
def create_network13():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='relu'))
    network.add(layers.Dense(units=9, activation='linear'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network13,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Relu, one Layer Linear, 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))))
```

```
Batch Normalize Layer, one Layer Relu, one Layer Linear, Softmax
Red
              precision    recall  f1-score   support

           1       0.72      0.78      0.75       744
           2       0.61      0.58      0.60       638
           3       0.62      0.52      0.56       217

   micro avg       0.67      0.67      0.67      1599
   macro avg       0.65      0.63      0.64      1599
weighted avg       0.66      0.67      0.66      1599

White
              precision    recall  f1-score   support

           1       0.66      0.66      0.66      1640
           2       0.58      0.63      0.61      2198
           3       0.60      0.49      0.54      1060

   micro avg       0.61      0.61      0.61      4898
   macro avg       0.61      0.59      0.60      4898
weighted avg       0.61      0.61      0.61      4898

RSME red
0.9804359787974584
RSME white
1.012755036930025
```

In [26]:

```
def create_network14():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='sigmoid'))
    network.add(layers.Dense(units=9, activation='relu'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network14,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Sigmoid, one Layer Relu, 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))))
```

```
Batch Normalize Layer, one Layer Sigmoid, one Layer Relu, Softmax
Red
              precision    recall  f1-score   support

           1       0.71      0.78      0.74       744
           2       0.56      0.57      0.57       638
           3       0.56      0.37      0.45       217

   micro avg       0.64      0.64      0.64      1599
   macro avg       0.61      0.57      0.59      1599
weighted avg       0.63      0.64      0.63      1599

White
              precision    recall  f1-score   support

           1       0.67      0.63      0.65      1640
           2       0.56      0.66      0.60      2198
           3       0.61      0.45      0.52      1060

   micro avg       0.60      0.60      0.60      4898
   macro avg       0.61      0.58      0.59      4898
weighted avg       0.61      0.60      0.60      4898

RSME red
0.9618623391245619
RSME white
0.9949403068365282
```

In [27]:

```
def create_network15():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='linear'))
    network.add(layers.Dense(units=9, activation='sigmoid'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network15,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Linear, one Layer Sigmoid, 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))))
```

```
Batch Normalize Layer, one Layer Linear, one Layer Sigmoid, Softmax
Red
              precision    recall  f1-score   support

           1       0.71      0.77      0.74       744
           2       0.56      0.57      0.57       638
           3       0.52      0.37      0.44       217

   micro avg       0.63      0.63      0.63      1599
   macro avg       0.60      0.57      0.58      1599
weighted avg       0.63      0.63      0.63      1599

White
              precision    recall  f1-score   support

           1       0.65      0.62      0.64      1640
           2       0.55      0.65      0.59      2198
           3       0.57      0.39      0.47      1060

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

RSME red
0.965697265762282
RSME white
0.9889824677486547
```

In [28]:

```
def create_network16():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='selu'))
    network.add(layers.Dense(units=9, activation='relu'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network16,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, one Layer Selu, one Layer Relu, 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))))
```

```
Batch Normalize Layer, one Layer Selu, one Layer Relu, Softmax
Red
              precision    recall  f1-score   support

           1       0.71      0.79      0.75       744
           2       0.57      0.57      0.57       638
           3       0.60      0.40      0.48       217

   micro avg       0.65      0.65      0.65      1599
   macro avg       0.63      0.58      0.60      1599
weighted avg       0.64      0.65      0.64      1599

White
              precision    recall  f1-score   support

           1       0.66      0.65      0.66      1640
           2       0.56      0.66      0.61      2198
           3       0.61      0.42      0.49      1060

   micro avg       0.60      0.60      0.60      4898
   macro avg       0.61      0.57      0.59      4898
weighted avg       0.61      0.60      0.60      4898

RSME red
0.962430831049828
RSME white
0.9923341893751506
```

In [29]:

```
def create_network17():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='selu'))
    network.add(layers.Dense(units=9, activation='selu'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network17,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, two Layer Selu, 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))))
```

```
Batch Normalize Layer, two Layer Selu, Softmax
Red
              precision    recall  f1-score   support

           1       0.71      0.78      0.74       744
           2       0.58      0.57      0.57       638
           3       0.61      0.42      0.50       217

   micro avg       0.65      0.65      0.65      1599
   macro avg       0.63      0.59      0.60      1599
weighted avg       0.64      0.65      0.64      1599

White
              precision    recall  f1-score   support

           1       0.67      0.65      0.66      1640
           2       0.57      0.65      0.61      2198
           3       0.60      0.46      0.52      1060

   micro avg       0.61      0.61      0.61      4898
   macro avg       0.62      0.59      0.60      4898
weighted avg       0.61      0.61      0.61      4898

RSME red
0.9648869126188652
RSME white
1.001568480889946
```

In [30]:

```
def create_network18():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='linear'))
    network.add(layers.Dense(units=9, activation='linear'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network18,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, two Layer Linear, 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))))
```

```
Batch Normalize Layer, two Layer Linear, Softmax
Red
              precision    recall  f1-score   support

           1       0.70      0.77      0.73       744
           2       0.55      0.55      0.55       638
           3       0.54      0.34      0.41       217

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

White
              precision    recall  f1-score   support

           1       0.64      0.58      0.61      1640
           2       0.53      0.67      0.59      2198
           3       0.57      0.34      0.42      1060

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

RSME red
0.9574556516161713
RSME white
0.9706098648392691
```

In [31]:

```
def create_network19():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='sigmoid'))
    network.add(layers.Dense(units=9, activation='sigmoid'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network19,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, two Layer Sigmoid, 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))))
```

```
Batch Normalize Layer, two Layer Sigmoid, Softmax
Red
              precision    recall  f1-score   support

           1       0.70      0.78      0.74       744
           2       0.56      0.54      0.55       638
           3       0.54      0.40      0.46       217

   micro avg       0.63      0.63      0.63      1599
   macro avg       0.60      0.57      0.58      1599
weighted avg       0.62      0.63      0.62      1599

White
              precision    recall  f1-score   support

           1       0.67      0.63      0.65      1640
           2       0.55      0.67      0.60      2198
           3       0.60      0.40      0.48      1060

   micro avg       0.59      0.59      0.59      4898
   macro avg       0.61      0.56      0.58      4898
weighted avg       0.60      0.59      0.59      4898

RSME red
0.9707980153405646
RSME white
0.9843621078354636
```

In [32]:

```
def create_network20():
    network = models.Sequential()
    network.add(layers.BatchNormalization())
    network.add(layers.Dense(units=64, activation='tanh'))
    network.add(layers.Dense(units=9, activation='tanh'))
    network.add(layers.Dense(3, activation='softmax'))

    network.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy', rsme])
    return network

neural_network = KerasClassifier(build_fn=create_network20,
                                 epochs=100,
                                 batch_size=100,
                                 verbose=0)
print("Batch Normalize Layer, two Layer Tanh, 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))))
```

```
Batch Normalize Layer, two Layer Tanh, Softmax
Red
              precision    recall  f1-score   support

           1       0.69      0.76      0.73       744
           2       0.56      0.54      0.55       638
           3       0.55      0.41      0.47       217

   micro avg       0.63      0.63      0.63      1599
   macro avg       0.60      0.57      0.58      1599
weighted avg       0.62      0.63      0.62      1599

White
              precision    recall  f1-score   support

           1       0.67      0.64      0.66      1640
           2       0.57      0.64      0.60      2198
           3       0.58      0.48      0.52      1060

   micro avg       0.61      0.61      0.61      4898
   macro avg       0.61      0.59      0.59      4898
weighted avg       0.61      0.61      0.60      4898

RSME red
0.9720565979858783
RSME white
1.0082808302525095
```