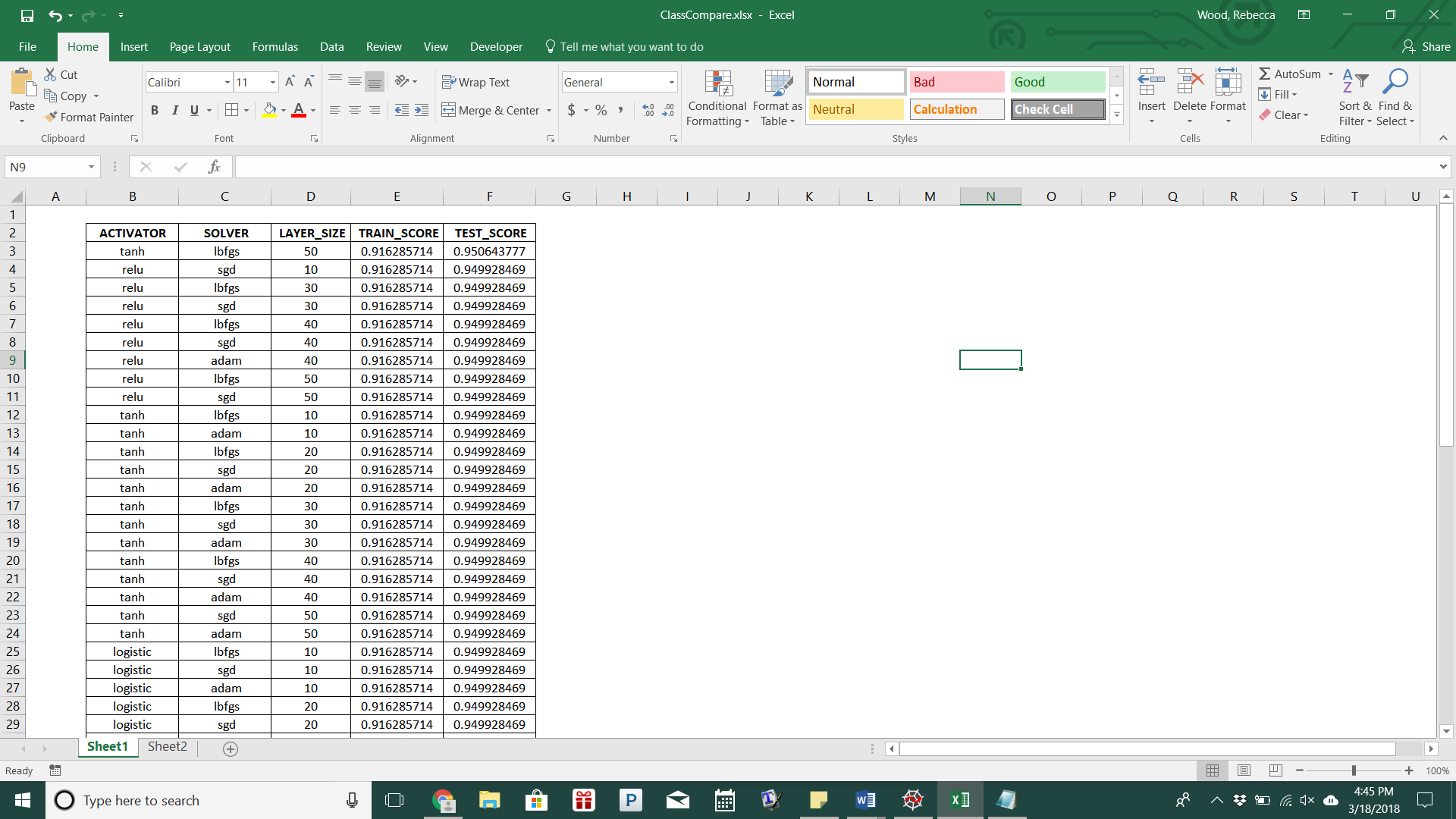
ANN Classification Assignment – White

# Experimentation

Using the white wine data set, the quality column was split into three bins. Analyzing the distribution of the wine quality gave a somewhat binomial distribution, leading to the assumption that three bins would be sufficient where a quality rating less than 4 was in a “low quality” bin, less than 7 was in a “medium quality” bin, and everything else was in a “high quality” bin. This was placed in a column called “labs”, and used as the y-response variable. Using a shuffle index, the training and test sets were created with 3500 observations. The initial model filters through the set of activator options: ‘Relu’, ‘Tanh’, ‘Logistic’, and ‘Identity’; filters through the set of solver options: ‘lbfgs’, ‘adam’, and ‘sgd’; as well as filters through a set of five hidden layer size options ranging from 10 to 50. The maximum number of iterations was set to 50, alpha to .0001, and the random state to 2016. At the conclusion, the given activator, solver, layer size, training score, and test score were printed to be later converted into an Excel spreadsheet to determine the model with the best test score.



Code

**”””PACKAGE IMPORT”””**

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import confusion\_matrix

from sklearn.model\_selection import train\_test\_split

from sklearn import preprocessing

import numpy as np

import pandas as pd

**”””DATA IMPORT”””**

white\_data = pd.read\_csv("winequality-white.csv", encoding = 'utf-8',

delimiter = ';')

**”””PRETREATING”””**

x = white\_data.iloc[:, 0:11]

qual = white\_data['quality']

white\_data['labs'] = pd.cut(qual, bins = ([2, 4, 7, 9]), labels =

['low', 'medium', 'high'])y = white\_data['labs']

x, y = x.as\_matrix(), y.as\_matrix()

x\_train, x\_test, y\_train, y\_test = x[:3500], x[3500:], y[:3500],

y[3500:]

shuffle\_index = np.random.permutation(3500)

x\_train, y\_train = x\_train[shuffle\_index], y\_train[shuffle\_index]

**”””MODEL DEVELOPMENT”””**

start\_time = time.time()

size = [10, 20, 30, 40, 50]

activators = ['relu', 'tanh', 'logistic', 'identity']

solvers = ['lbfgs', 'sgd', 'adam']

print("ACTIVATOR", "SOLVER", "LAYER\_SIZE", "TRAIN\_SCORE",

"TEST\_SCORE")

for act in activators:

for siz in size:

for solv in solvers:

mlp = MLPClassifier(activation = act, hidden\_layer\_sizes =

(siz,), max\_iter = 50, alpha = 1e-4, solver = solv, verbose = False, tol = 1e-4, random\_state = 2016)

pred = mlp.fit(x\_train, y\_train).predict(x\_test)

train\_score = mlp.score(x\_train, y\_train)

test\_score = mlp.score(x\_test, y\_test)

print(act, solv, siz, train\_score, test\_score)

stop\_time = time.time()

print("Time Required for Optimization:", stop\_time - start\_time)

**”””FINAL MODEL”””**

final\_mlp = MLPClassifier(activation = 'tanh', hidden\_layer\_sizes =

(40,), max\_iter = 50, alpha = 1e-4, solver = 'adam', verbose = False, tol = 1e-4, random\_state = 2016)

# final\_fit = final\_mlp.fit(x\_train, y\_train)

# final\_pred = final\_fit.predict(x\_test)

# print("Train Set Score: %f" % final\_mlp.score(x\_train, y\_train)) print("Test Set Score: %f" % final\_mlp.score(x\_test, y\_test))

# Final Results

The best produced model has ‘Tanh’ as the activation, ‘lbfgs’ as the solver with a hidden layer size of 50. This produces a training accuracy rating of 91.62857% and a test accuracy rating of 95.06438%. This high accuracy rating is supported by the small number of bins and the seemingly binomially distributed wine qualities.