**ANN Regressor Assignment**

Mengting Ding

**Executive Summary**

The objective of this assignment is to create a regression Artificial Neural Network on the white wine quality dataset from archive.ics.uci.edu website in order to predict the white wine quality.

There are 12 variables in the dataset, besides the target variable, quality which is in ordinal scale with 0 to 10, there are 11 predicting variables, including fixed acidity, volatile acidity, citric acid, etc.

Parameters are tuned for trying different models and comparing them to have the best model possible, theses parameters are epoch and batchsize. The metrics used to compare models is MSE, the best model has the lowest MSE. The best model in this assignment has 200 epochs, 20 batch size and testing MSE 0.535.

**ANN model in code**

The data is pretreated before building the ANN regression model. After reading in the data, it is split into X and Y and using train test split function to split the independent and dependent variables into 80 percent of training dataset and 20 percent of testing dataset.

import datetime

import numpy as np

import pandas as pd

import tensorflow

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

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

import matplotlib.pyplot as plt

data = pd.read\_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv",sep=";")

dataset = data.values

Y = dataset[:,11]; X = dataset[:,:11]

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X,Y,test\_size=0.2, random\_state=42)

X\_train = X\_train.astype('float32')

X\_test = X\_test.astype('float32')

np.random.seed(5072)

start\_time = datetime.datetime.now()

model=Sequential()

model.add(Dense(30, input\_dim=11, activation='relu'))

model.add(Dense(20, activation='relu'))

model.add(Dense(10, activation='relu'))

model.add(Dense(1))

#opt = tensorflow.keras.optimizers.Adam(learning\_rate=0.002)

model.compile(optimizer="adam", loss='mean\_squared\_error', metrics =['mse','mae'])

history=model.fit(X\_train,Y\_train, validation\_data=(X\_test,Y\_test),

epochs=200, batch\_size=20, verbose=0)

stop\_time = datetime.datetime.now()

print ("Time required for training:",stop\_time - start\_time)

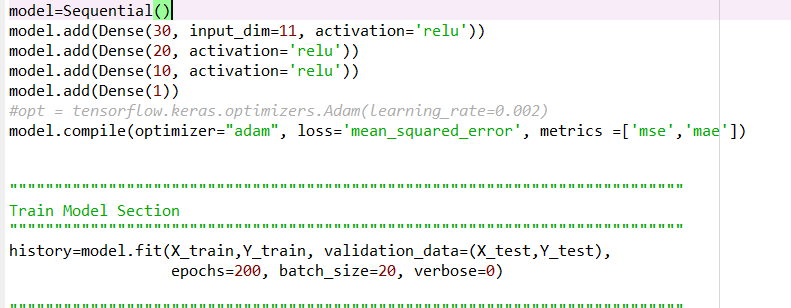
print("Training:",model.evaluate(X\_train,Y\_train))

print("Validation:", model.evaluate(X\_test,Y\_test))

# Visualize model history - diagnostic plot

**Experimental plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epochs | Layers | Batch size | Run time | Testing mse |
| **200** | **2** | **20** | **1m15s** | **0.535** |
| 150 | 1 | 30 | 40s | 0.577 |
| 200 | 2 | 20 | 1m 12s | 0.615 |
| 400 | 3 | 20 | 2m50s | 0.544 |

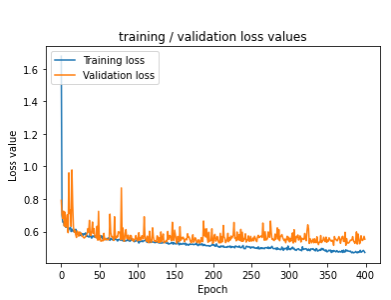


The best experimented model is the first one, having the smallest testing MSE among all. This model has two hidden layers and 200 epochs with batch size of 20.

**Metrics**

Mean squared error (MSE) and mean absolute error (MAE) are the chosen metrics for the model as both of them are widely used metrics in regression models.

**Graph**

****