# Critical Analysis Task

Student Name: Muhammad Muqarrab Ghori

StudentID: 47111496

## Introduction

This report is a thorough evaluation of the critical analysis task Jupyter notebook, it aims to examine embedded issues within the notebook and further provided more appropriate methods in contexts of those issues. Addressing these issues will be done via through, knowledge attained in comp2200. Notebook is a study on the factors contributing to the quality of wine. There are many features within the dataset to help predict the quality of wine.

## Column Name issue

It is important when working with any sort of data the column name do no contain white spaces or special characters between words. Replacing white spaces and special characters by ‘\_’ or no space using camelCase makes it easier to read the data and avoid errors as some tools don’t support special characters or may misinterpret the input string.

It would be suggested that these columns names are converted using rename() and replace() functions.

**Sample Code**

for columnName in data:

data.rename(columns = {columnName:columnName.replace(" ", "\_")}, inplace = True)

data.head()

Example: ‘fixed acidity’ -> ‘fixed\_acidity’

## Feature selection issue

Its crucial to select appropriate features to train the neutral networks on as it directly affects the accuracy of the model. Using histograms to check for density of a feature is not the best way to determine its effects on quality, although more evenly distributed features are preferred it doesn’t give a definite answer. There are multiple ways to find features that are appropriate, this could be done using correlation tables/heatmaps, chi-square test, etc.

I would suggest using a correlation heatmap to determine which features are to be selected to train the model on. Using the correlation heatmap I would remove all the negatively correlated features as the are deem useless for training a model, normally we set a threshold for correlation to be above a certain value to be selected as feature, but in this case, we would simply use all the positively correlated features within .01 tolerance to train a model.

**Sample Code**

cor = data.corr()

plt.figure(figsize = (10,6))

sns.heatmap(cor, annot = True)

Chart, treemap chart

Description automatically generated **Figure 1: Correlation Heatmap**

for columnName in data:

correlation = data[columnName].corr(data['quality'])

#inp = [columnName,round(correlation, 4)]

if str(columnName) != 'quality':

mylist.append(round(correlation, 4))

print('Correlation between Quality and %s: %.4f' % (columnName.title(), correlation))

Correlation between Quality and Fixed\_Acidity: 0.1220

Correlation between Quality and Volatile\_Acidity: -0.3740

Correlation between Quality and Citric\_Acid: 0.2408

Correlation between Quality and Residual\_Sugar: 0.0220

Correlation between Quality and Chlorides: -0.0623

Correlation between Quality and Free\_Sulfur\_Dioxide: -0.0633

Correlation between Quality and Total\_Sulfur\_Dioxide: -0.1833

Correlation between Quality and Density: 0.0012

Correlation between Quality and Ph: -0.0525

Correlation between Quality and Sulphates: 0.2577

Correlation between Quality and Alcohol: 0.4849

Using the above correlation values, we would select features: alcohol, sulphates, fixed\_acidity, citric\_acid, residual\_sugar.

## Setting up train test split

As mentioned, before we used to correlation to select the features we want to train on the model. As the notebook suggests features chlorides and density should be dropped as they’re too concentrated, when dropping features we have to specify them when initiating x\_ex1 variable.

I would suggest dropping all the features that are not selected in the above example included the features the model being trained on.

**Sample Code**

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

data.sort\_values(by=['quality'],ascending=False,inplace=True)

print(data['quality'].value\_counts())

x\_ex1 = data.copy().drop(columns=['quality', 'volatile\_acidity', 'chlorides', 'free\_sulfur\_dioxide', 'total\_sulfur\_dioxide', 'pH'])

y\_ex1 = data.copy()['quality']

x\_ex1\_array = x\_ex1.values

y\_ex1\_array = y\_ex1.values

x\_train = x\_ex1\_array[0:int((len(y\_ex1\_array)+1)\*0.9),:]

x\_test = x\_ex1\_array[int((len(y\_ex1\_array)+1)\*0.9):,:]

y\_train = y\_ex1\_array[0:int((len(y\_ex1\_array)+1)\*0.9)]

y\_test = y\_ex1\_array[int((len(y\_ex1\_array)+1)\*0.9):]

**Figure 2: Accuracy & Loss Graph Before Figure 3: Accuracy & Loss graph after**

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Above the shows the significant improvement in test accuracy and a bit of reduction in loss graph. Hence improving the accuracy of the model.

## Conclusion

This report was aimed to overcome the issues within the Jupyter notebook to represent the data in a more accurate way. The suggested fixes allow for more accurate interpretation of the Wine Quality dataset. Above sample code provides a more insightful way of conducting study on the quality of Wine, which can be further improved through more accurate models.