

***Faculty of Science and Technology***

**Assignment Coversheet**

|  |  |
| --- | --- |
| **Student ID number &**  **Student Name** | u873301  Stephen Farrugia |
| **Unit name** | Software Technology 1 G |
| **Unit number** | 8895 |
| **Unit Tutor** | Dr Girija Chetty |
| **Assignment name** | ST1 Capstone Project – Semester 1 2023 |
| **Due date** | 12 May 2023 |
| **Date submitted** | 12 May 2023 |

**You must keep a photocopy or electronic copy of your assignment.**

**Student declaration**

I certify that the attached assignment is my own work. Material drawn from other sources has been appropriately and fully acknowledged as to author/creator, source and other bibliographic details.

**Signature of student: \_\_Stephen Farrugia Date: \_12 May 2023\_\_**

(electronically submitted)

Table of Contents

[Introduction 3](#_Toc134741880)

[Methodology 3](#_Toc134741881)

[Dataset Description 4](#_Toc134741882)

[Initial Exploratory Data Analysis 4](#_Toc134741883)

[Initial Predictive Data Analysis 11](#_Toc134741884)

[Data Cleansing 15](#_Toc134741885)

[Second Iteration of Predictive Data Analysis 17](#_Toc134741886)

[Data Model Export 19](#_Toc134741887)

[Algorithm Implementation in a desktop GUI 20](#_Toc134741888)

[Software Deployment on a web-page 20](#_Toc134741889)

[Conclusions 21](#_Toc134741890)

[References 22](#_Toc134741891)

# Introduction

This report outlines the work done and results of Software Technology 1 unit capstone project. The assignment handout can be found at [1]. I have chosen to work on the quality of red wine based on a dataset available on the Kaggle repository [2].

Australia is the fifth largest wine producer and exporter in the world and the industry contributes almost 165,000 jobs and over $45.5 billion to the Australian economy [3].

Wine quality is a very subjective thing, however, through large data groups an analysis is proposed to determine whether a combination of physiochemical properties consistently produce similar quality wine – whether that be poor, good or great.

This report includes the python code used as part of the exploratory data analysis, predictive analysis undertaken using the Google Colab platform [4]. This platform provides a high capacity for analysis and has many well integrated functions.

The subsequent user input approaches for a desktop GUI and web-based interface were implemented using PyCharm Professional [5] as this allowed the use of Tkinter for the desktop GUI and also provided a ready interface to flask for web-page development.

# Methodology

An adapted version of the seven step approach to machine learning outlined in [6] was used for both exploratory and predictive data analysis. These seven steps are:

1. Import the data
2. Clean the data
3. Split the data into training and test sets
4. Create a Model [or in this case multiple models]
5. Train the Models
6. Make Predictions
7. Evaluate [and compare the models] and improve

Seven predefined models from sklearn are compared to determine the preferred model.

Once the preferred model is defined, this is then used as part of both the desktop GUI implementation and the web-page deployment.

### Dataset Description

A single dataset is used which is available from Kaggle [2]. This red wine data set describes eleven wine constituent components and one wine quality assessment. The file contains components and quality assessments of 1,599 red wine samples. The objective is to determine whether for a given set of physiochemical properties, a wine quality can be determined.

### Initial Exploratory Data Analysis

Google Colab was selected to conduct the initial phase of software development. As per the Methodology, this involved loading the data, understanding the data and then conducting appropriate data cleansing.

I chose to load all the libraries, including those required for predictive data analysis, at the start of my code prior to loading the data file as per the following code.

# Capstone Model Development

# Load libraries

import matplotlib.pyplot

import numpy as np

from matplotlib import pyplot as plt

import seaborn as sbn

from pandas import read\_csv

from pandas import set\_option

from pandas.plotting import scatter\_matrix

from sklearn.preprocessing import StandardScaler

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

from sklearn.model\_selection import KFold

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from sklearn.metrics import accuracy\_score

from sklearn.pipeline import Pipeline

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.ensemble import AdaBoostClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import ExtraTreesClassifier

# load the data set

fileLocation = '/content/drive/MyDrive/Colab Notebooks/CapstoneWines/winequality-red.csv'

df = read\_csv(fileLocation, header=[0])

The following code and results provide an understanding of the data.

# interrogate the data

# Check first and last 5 rows

print(df.head())

print(df.tail())

fixed acidity volatile acidity citric acid residual sugar chlorides \

0 7.4 0.70 0.00 1.9 0.076

1 7.8 0.88 0.00 2.6 0.098

2 7.8 0.76 0.04 2.3 0.092

3 11.2 0.28 0.56 1.9 0.075

4 7.4 0.70 0.00 1.9 0.076

free sulfur dioxide total sulfur dioxide density pH sulphates \

0 11 34 0.9978 3.51 0.56

1 25 67 0.9968 3.20 0.68

2 15 54 0.9970 3.26 0.65

3 17 60 0.9980 3.16 0.58

4 11 34 0.9978 3.51 0.56

alcohol quality

0 9.4 5

1 9.8 5

2 9.8 5

3 9.8 6

4 9.4 5

fixed acidity volatile acidity citric acid residual sugar chlorides \

1594 6.2 0.600 0.08 2.0 0.090

1595 5.9 0.550 0.10 2.2 0.062

1596 6.3 0.510 0.13 2.3 0.076

1597 5.9 0.645 0.12 2.0 0.075

1598 6.0 0.310 0.47 3.6 0.067

free sulfur dioxide total sulfur dioxide density pH sulphates \

1594 32 44 0.99490 3.45 0.58

1595 39 51 0.99512 3.52 0.76

1596 29 40 0.99574 3.42 0.75

1597 32 44 0.99547 3.57 0.71

1598 18 42 0.99549 3.39 0.66

alcohol quality

1594 10.5 5

1595 11.2 6

1596 11.0 6

1597 10.2 5

1598 11.0 6

# confirm the number of rows and columns

print(df.shape)

(1599, 12)

# get the column names

print(df.columns)

Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',

'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',

'pH', 'sulphates', 'alcohol', 'quality'],

dtype='object')

# check how many unique values there are for each attribute

print(df.nunique())

fixed acidity 96

volatile acidity 143

citric acid 80

residual sugar 91

chlorides 153

free sulfur dioxide 57

total sulfur dioxide 143

density 436

pH 89

sulphates 96

alcohol 65

quality 6

dtype: int64

# obtain complete information on the data

print(df.info())

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1599 entries, 0 to 1598

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 fixed acidity 1599 non-null float64

1 volatile acidity 1599 non-null float64

2 citric acid 1599 non-null float64

3 residual sugar 1599 non-null float64

4 chlorides 1599 non-null float64

5 free sulfur dioxide 1599 non-null int64

6 total sulfur dioxide 1599 non-null int64

7 density 1599 non-null float64

8 pH 1599 non-null float64

9 sulphates 1599 non-null float64

10 alcohol 1599 non-null float64

11 quality 1599 non-null int64

dtypes: float64(9), int64(3)

memory usage: 150.0 KB

# check how many samples per quality

print(df.groupby(['quality']).size())

quality

3 10

4 53

5 681

6 638

7 199

8 18

dtype: int64

# Visualising data distribution in detail

df.hist(sharex=False, sharey=False, xlabelsize=5, ylabelsize=5, bins=30)

plt.show()

Diagram, engineering drawing

Description automatically generated

# show density plot

df.plot(kind='density', subplots=True, layout=(4,4), sharex=False, sharey=False, legend=True)

plt.show()

Diagram

Description automatically generated

# Correlation matrix

sbn.set(style="white")

plt.rcParams['figure.figsize'] = (15, 10)

sbn.set(font\_scale=0.63)

sbn.heatmap(df.corr(), annot = True, linewidths=.5, cmap="Greens")

plt.title('Correlation Between Variables', fontsize = 20)

plt.show()

Chart, table, treemap chart

Description automatically generated

# generate box plots to look for outliers

df.plot(kind='box', subplots=True,

        layout=(2,6), sharex=False, sharey=False, figsize=(20, 10), color='green');

plt.show()

Chart, box and whisker chart

Description automatically generated

Once the data was understood, even with the identified outliers, it was decided to run a range of models to determine how well they would work.

### Initial Predictive Data Analysis

Firstly, the data was formatted to be suitable for use with the models.

# Converting the data to be suitable for model development

# by separating the inputs and the quality output

array = df.values

X = array[:,0:11].astype(float)

Y = array[:,11]

test\_size = 0.20

seed = 7

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

# confirm subsets after splitting

print(np.shape(X\_train), np.shape(X\_test))

(1279, 11) (320, 11)

# Set up of algorithms

models = []

models.append(('DT', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC()))

models.append(('GBM', GradientBoostingClassifier()))

models.append(('RF', RandomForestClassifier()))

models.append(('ABC', AdaBoostClassifier()))

models.append(('ETC', ExtraTreesClassifier()))

# evaluate each model in turn

results = []

names = []

print("Performance on Training set")

for name, model in models:

  kfold = KFold(n\_splits=num\_folds, shuffle=True, random\_state=seed)

  cv\_results = cross\_val\_score(model, X\_train, Y\_train, cv=kfold, scoring=scoring)

  results.append(cv\_results)

  names.append(name)

  msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())

  msg += '\n'

  print(msg)

Performance on Training set

DT: 0.594234 (0.020203)

NB: 0.536345 (0.023217)

SVM: 0.498866 (0.028117)

GBM: 0.655993 (0.016522)

RF: 0.677117 (0.019159)

ABC: 0.541078 (0.024866)

ETC: 0.684145 (0.016706)

# Use a Box plot to compare the Algorithms' Performance

fig = plt.figure()

fig.suptitle('Algorithm Comparison - Raw data')

ax = fig.add\_subplot(111)

plt.boxplot(results)

ax.set\_xticklabels(names)

plt.show()

Chart, box and whisker chart

Description automatically generated

# now add the StandardScaler to improve the performance

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.fit\_transform(X\_test)

# now try the models again to evaluate each model in turn

results = []

names = []

print("Performance on Training set")

for name, model in models:

  kfold = KFold(n\_splits=num\_folds, shuffle=True, random\_state=seed)

  cv\_results = cross\_val\_score(model, X\_train, Y\_train, cv=kfold, scoring=scoring)

  results.append(cv\_results)

  names.append(name)

  msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())

  msg += '\n'

  print(msg)

Performance on Training set

DT: 0.589553 (0.021156)

NB: 0.541029 (0.026539)

SVM: 0.600472 (0.011318)

GBM: 0.649724 (0.010709)

RF: 0.679458 (0.014611)

ABC: 0.541078 (0.024866)

ETC: 0.681789 (0.011019)

# Use a Box plot to compare the Algorithms' Performance after StandardScaler applied

fig = plt.figure()

fig.suptitle('Algorithm Comparison Post StandardScaler')

ax = fig.add\_subplot(111)

plt.boxplot(results)

ax.set\_xticklabels(names)

plt.show()

Chart, box and whisker chart

Description automatically generated

Try training the models and see how it improves.

name = []

for name, model in models:

  model.fit(X\_train, Y\_train)

  print(name, str(model.score(X\_test, Y\_test)))

DT 0.553125

NB 0.53125

SVM 0.65

GBM 0.625

RF 0.634375

ABC 0.5625

ETC 0.690625

### Data Cleansing

Model performance isn’t as good as we’d like so it is worth removing the outliers and trying again.

# identify the outliers

# define continuous variables

continuous\_features = ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',

       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',

       'pH', 'sulphates', 'alcohol']

def outliers(df\_out, drop = False):

    for each\_feature in df\_out.columns:

        feature\_data = df\_out[each\_feature]

        # Calculate the outliers using the 1.5 \* IQR where IQR is the difference between the bottom 25 and top 75 Quartiles

        Q1 = np.percentile(feature\_data, 25.)

        Q3 = np.percentile(feature\_data, 75.)

        IQR = Q3-Q1

        outlier\_step = IQR \* 1.5

        outliers = feature\_data[~((feature\_data >= Q1 - outlier\_step) & (feature\_data <= Q3 + outlier\_step))].index.tolist()

        if not drop:

            print('For the feature {}, No of Outliers is {}'.format(each\_feature, len(outliers)))

        if drop:

            df.drop(outliers, inplace = True, errors = 'ignore')

            print('Outliers from {} feature removed'.format(each\_feature))

outliers(df[continuous\_features])

For the feature fixed acidity, No of Outliers is 49

For the feature volatile acidity, No of Outliers is 19

For the feature citric acid, No of Outliers is 1

For the feature residual sugar, No of Outliers is 155

For the feature chlorides, No of Outliers is 112

For the feature free sulfur dioxide, No of Outliers is 30

For the feature total sulfur dioxide, No of Outliers is 55

For the feature density, No of Outliers is 45

For the feature pH, No of Outliers is 35

For the feature sulphates, No of Outliers is 59

For the feature alcohol, No of Outliers is 13

#drop the outliers

outliers(df[continous\_features], drop = True)

Outliers from fixed acidity feature removed

Outliers from volatile acidity feature removed

Outliers from citric acid feature removed

Outliers from residual sugar feature removed

Outliers from chlorides feature removed

Outliers from free sulfur dioxide feature removed

Outliers from total sulfur dioxide feature removed

Outliers from density feature removed

Outliers from pH feature removed

Outliers from sulphates feature removed

Outliers from alcohol feature removed

#check if outliers got removed

df.plot(kind='box', subplots=True,

        layout=(2,6),sharex=False,sharey=False, figsize=(20, 10), color='green');

Calendar, box and whisker chart

Description automatically generated

This significantly reduced the number of outliers but interestingly didn’t completely remove them. Further study could include removal of a second round of outliers.

# check the shape of the resulting data

df.shape

(1194, 12)

### Second Iteration of Predictive Data Analysis

# check the performance after outlier removal

print("Performance on Training set after outlier removal")

for name, model in models:

  kfold = KFold(n\_splits=num\_folds, shuffle=True, random\_state=seed)

  cv\_results = cross\_val\_score(model, X\_train, Y\_train, cv=kfold, scoring=scoring)

  results.append(cv\_results)

  names.append(name)

  msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())

  msg += '\n'

  print(msg, str(model.score(X\_test, Y\_test)))

Performance on Training set after outlier removal

DT: 0.577816 (0.017693)

0.559375

NB: 0.541029 (0.026539)

0.53125

SVM: 0.600472 (0.011318)

0.65

GBM: 0.648943 (0.009733)

0.625

RF: 0.671642 (0.022730)

0.65625

ABC: 0.541078 (0.024866)

0.5625

ETC: 0.677114 (0.018222)

0.678125

# Use a Box plot to compare the Algorithms' Performance

fig = plt.figure()

fig.suptitle('Algorithm Comparison - Pre and Post Outlier removal')

ax = fig.add\_subplot(111)

plt.boxplot(results)

ax.set\_xticklabels(names)

plt.show()

Chart, box and whisker chart

Description automatically generated

### Data Model Export

Whilst still not ideal, as we are talking about subjective wine quality rather than a life-threatening disease, I have chosen to proceed and select the ExtraTreesClassifier as the preferred model to export.

import pandas as pd

from sklearn.ensemble import ExtraTreesClassifier

import pickle

model = ExtraTreesClassifier()

model.fit(X\_train, Y\_train)

print("Best model result on test input: " + str(model.score(X\_test, Y\_test)))

pickle.dump(model, open('/content/drive/MyDrive/Colab Notebooks/CapstoneWines/winequality\_predictor.h5', 'wb'))

Best model result on test input: 0.7

from traitlets.config import loader

#reload model to confirm it is working

with open('/content/drive/MyDrive/Colab Notebooks/CapstoneWines/winequality\_predictor.h5', 'rb') as file:

  loaded\_model = pickle.load(file)

#validate median and standard deviation of test data

print(str(loaded\_model.score(X\_test, Y\_test)))

0.7

# reload file from github and make sure it works

! apt-get install git

! git clone https://github.com/farruge/ST1Capstone.git

from traitlets.config import loader

#reload model to confirm it is working

with open('/content/ST1Capstone/winequality\_predictor.h5', 'rb') as file:

  loaded\_model = pickle.load(file)

#validate median and standard deviation of test data

print(str(loaded\_model.score(X\_test, Y\_test)))

Reading package lists... Done

Building dependency tree

Reading state information... Done

git is already the newest version (1:2.25.1-1ubuntu3.11).

0 upgraded, 0 newly installed, 0 to remove and 24 not upgraded.

Cloning into 'ST1Capstone'...

remote: Enumerating objects: 30, done.

remote: Counting objects: 100% (30/30), done.

remote: Compressing objects: 100% (28/28), done.

remote: Total 30 (delta 8), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (30/30), 5.64 MiB | 2.71 MiB/s, done.

0.7

### Algorithm Implementation in a desktop GUI

The best performing algorithm has been packaged and available for use in a desktop GUI program.

This desktop GUI was developed using Pycharm Professional using the Tkinter package. The project is available at the following git-hub link.

<https://github.com/farruge/ST1Capstone.git>

### Software Deployment on a web-page

Again, using the same packaged prediction model, a web-page was developed.

Pycharm Professional with flask interface and html were used to develop a web-page to obtain the various components and provide a prediction of the wine quality. This can be found at the following git-hub link.

<https://github.com/farruge/ST1Capstone.git>

# Conclusions

Through the use of exploratory data analysis and predictive data analytics, a machine learning model has been developed to predict the expected wine quality based on the various component elements. This model has been implemented in Python to be available on a desktop using a simple GUI and as a web-page. The practical use of this model would primarily be for the wine industry as wine consumers are unlikely to have access to this detailed wine components included in the analysis. This analysis could be used by winemakers to optimise their wine components for cost minimisation and maximum quality. For wine consumers, a different set of inputs - which are readily available – such as region, winery, grape varieties, alcohol content and year are more likely to be inputs to a usable wine quality calculator.

# References

1. <https://uclearn.canberra.edu.au/courses/13571/assignments/105232>
2. P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.  
   Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

<https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009>

1. <https://www.wineaustralia.com/market-insights/australian-wine-sector-at-a-glance>
2. Google Colab platform - <https://colab.research.google.com/>
3. Pycharm Professional - <https://www.jetbrains.com/pycharm/>
4. Machine Learning with Python - <https://colab.research.google.com/>
5. <https://www.geeksforgeeks.org/retrieving-html-from-data-using-flask/>
6. <https://www.tutorialspoint.com/How-to-center-text-in-HTML>