CETM72 Assignment 2

By Giselle Vicatos

bh86zm

**Wine Quality Prediction Using Machine Learning**

(total words:1596)

**Introduction**

The purpose of this study is to develop a model that can predict the quality of wine. Wine has a variety of features, like pH, sulphates, chlorides, acidity and other acids (Gupta and C, 2021). Machine learning can identify which of these features are important in controlling the quality of wine, which in turn will predict the wine quality. This study aims to identify an algorithm/s that can provide an accurate model/s to predict the quality of wine and to identify the top four most important variables that contribute to a high wine quality. To achieve this a series of classification algorithms will be used and their performance metrics will be compared. The similarities and/or differences between different types of wine will also be compared. This study will first describe the data, the different types of algorithms used and how their performance will be measured. Then, the study will discuss the pre-processing methods, how the machine learning algorithms were built, the results of the prediction and finally the concluding remarks.

**Data used**

The dataset (Cortez *et al.*, 2009) that was chosen for classifying different wine types into quality categories was obtained from Kaggle. The dataset consists of 4898 white wines and 1599 red wines from the Portuguese “Vinho Verde” wines (Fig.1).

Chart, bar chart

Description automatically generated

Fig.1 Number of red and white wines in dataset.

Each wine has eleven physiochemical properties, namely fixed acidity, volatile acidity, total sulfur, chlorides, pH, free sulfur dioxide, density, residual sugar, citric acid, sulphates, and alcohol. A sensory score was included to rate the quality on a continuous scale from 0 (poor) to 10 (excellent) (Cortez *et al.*, 2009). The physiochemical property measurements are numeric, the quality rating is an integer, and the wine types are categorical variables stored as factors. Due to this type of data, either regression or classification algorithms can be used (Nebot, Mugica and Escobet, 2015). However, to perform classification algorithms on each wine type, the quality rating must be categorical and therefore must be converted to a factor. Then, for each wine type, the quality rating will be the dependant variable and the eleven features will be the independent variables.

**Machine learning methods used**

Four classification algorithms have been chosen to predict the quality of wine, namely Decision Trees (DT), K-Nearest Neighbors (KNN), Random Forest (RF), and Support Vector Machines (SVM). A comparison analysis will be done, whereby the performance of each algorithm will be evaluated and the algorithm with the best performance will be singled out. The quality of wine will be split into a high and low quality, and so the system is classed as a binary classification task. To evaluate the performance of each algorithm, a bench-mark metric is the confusion matrix, from which the Accuracy, Sensitivity/Recall, Specificity, Precision, F1-Score, and Area Under the ROC Curve (AUC) can be calculated and used as performance metrics (Gupta and C, 2021). The confusion matrix is defined using the terms, True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN), and the mentioned performance metrics are calculated from these four terms.

**Practical: Pre-processing of data**

The dataset was read in the form of a .csv file. Pre-processing started with removing the rows with null values and reassigning the quality rating to make it more understandable, so 0-5 became 0 (low quality), and 6-10 became 1 (high quality). Additionally, the white and red wines were separated. A diagram depicting these changes can be seen in Fig.2.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Chart, bar chart

Description automatically generated

Fig.2. Quality distribution for red and white wine datasets.

The data was checked for outliers by plotting boxplots for each feature and assessing the distribution of the values (Fig.3). Only extreme lone values were identified as outliers and removed. The method involved replacing the outlier with a null value and then removing the affected rows.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Calendar

Description automatically generated with medium confidence

Fig.3. Distribution of values for each feature using boxplots

The values in the feature variables are widely spread, so normalization was performed to transform the data to comparable scales. This will prevent large values from causing dominance in a model’s performance (Dahal *et al.*, 2021).

Graphical user interface, application

Description automatically generated

From Fig.2, the data distribution between the quality classes is imbalanced, which can cause bias and the misclassification of classes (Hu *et al.*, 2016). Therefore, the training datasets were resampled with the SMOTE method.

Graphical user interface, application

Description automatically generated

**Practical: R Programming content**

The wine datasets were split 80/20, into a training and testing dataset to estimate the performance of the algorithms. The training set is used to fit the model and the testing set is used to introduce new data to test the quality of the model (O’Neil and Schutt, 2013).

Graphical user interface, application

Description automatically generated

To help prevent overfitting, k-fold cross validation was used for each model, whereby the training set was divided into 5 subsets from which the model was iteratively trained on 4 folds and tested on the remaining fold. This whole process was repeated three times and allowed hyperparameters to tune each model. The tuneLength parameter was added so that the algorithm can automatically tune by trying five different default values instead of only the main parameter, and a preProcess function with the centre and scale methods was added to estimate the location and scale of the predictors (*The caret Package*, no date).

Training and testing the models for the four algorithms, as well as calculating their performance metrics, meant that there would be a lot of repetitive code. Therefore, a function was written to minimize the repetition. The function takes in three parameters, the training set, the test set, and the algorithm name, and by using if/else if statements, the function allows models to be trained using k-fold cross validation specific for each algorithm. The function then tests each model and calculates the performance metrics. The function also creates a results table where all the calculated performance metrics are tabulated for each model. Additionally, in the if/else if statements for each model, the variables of importance are determined. The function does not recognise wine types, so the function must be called for each wine type separately.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Practical: Display of data/results**

After calling the function for each wine type, the performance results for each model can be viewed in Table 1.

Graphical user interface, application

Description automatically generated

Table 1. Performance results for white and red wine, where rpart is DT and svmRadial is SVM.

Graphical user interface, application

Description automatically generated

These results are calculated from the confusion matrices, which can be seen in Fig.4a and Fig.4b for white and red wine respectively.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

|  |  |
| --- | --- |
| (a) | (b) |

Fig.4. Confusion matrix for (a) white wine models and (b) red wine models.

Each confusion matrix shows the distribution of data that has been sorted into the categories TP, TN, FP and FN. From these matrices, Accuracy, Sensitivity/Recall, Specificity, Precision, F1-Score and AUC have been calculated and can be seen in Table 1.

Sensitivity/Recall measures the rate of true positives, i.e. how often class 1 can be correctly predicted. Specificity measures the rate of true negatives, i.e. how often class 0 can be correctly predicted. An unbalanced dataset would have produced an imbalance in values between Sensitivity/Recall and Specificity, but since balancing methods were applied, the values are similar and their magnitudes (70-80%), indicate that both classes have a good chance of being accurately predicted. This also causes the overall Accuracy rating of each model to be more reliable and thus the model with the highest probability (approximately 80%) of predicting the correct class for both wine types, is the RF model.

Precision shows the ability of the model to find the relevant data points and the F1-Score measures the test’s accuracy. These Precision and F1-Score values are approximately 75-90% and 73-86% respectively for all models. Both metrics show that the models are of high quality. These high quality ratings are also reflected in the AUC values. When considering all the performance metrics, the RF model has the highest performance for both wine types, the DT model has the lowest, and the SVM and KNN models appear to have a similar middling performance. Additionally, red wine has slightly lower performance metrics than white wine, which could be due to a smaller red wine dataset.

The top four variables of importance for white wine are alcohol, density, volatile acidity and chlorides, while the top four variables of importance for red wine are alcohol, sulphates, volatile acidity and total sulfur dioxide (Fig.5). The similar variables for both red and white wine are alcohol and volatile acidity, which can be attributed to the fact that alcohol contributes to the taste and texture of wine (Dahal *et al.*, 2021) and volatile acidity has a negative impact due to acetic acid (vinegar)(Cortez *et al.*, 2009).

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

|  |  |
| --- | --- |
| (a) | Graphical user interface, application  Description automatically generated  (b) |

Fig.5. Variables of Importance for (a) white wine and (b) red wine.

**Conclusions**

Machine learning algorithms were able to predict the quality of wine by analysing the features of wine. Two types of wine, red and white, were studied and each wine had eleven features which acted as the independent variables. The rating for the quality of wine, which was either high (1) or low (0), acted as the dependant variable. The classification algorithms, DT, KNN, RF and SVM were used to build the models and from the four algorithms, RF was found to be the best at predicting the quality of wine for both white and red wine. The metrics that were used to determine these results were Accuracy, Sensitivity/Recall, Specificity, Precision, F1-Score and AUC. Finally, the top four variables of importance were determined for red and white wine, and it was found that alcohol and volatile acidity were the top common variables. Additionally, white wine had chlorides and density as top variables, while red wine had sulphates and total sulfur dioxide.

**References**

Cortez, P. *et al.* (2009) ‘Modeling wine preferences by data mining from physicochemical properties’, *Decision Support Systems*, 47(4), pp. 547–553. doi: 10.1016/j.dss.2009.05.016.

Dahal, K. R. *et al.* (2021) ‘Prediction of Wine Quality Using Machine Learning Algorithms’, *Open Journal of Statistics*, 11(02), pp. 278–289. doi: 10.4236/ojs.2021.112015.

Gupta, M. and C, V. (2021) ‘A Study and Analysis of Machine Learning Techniques in Predicting Wine Quality’, *International Journal of Recent Technology and Engineering*, 10(1), pp. 314–321. doi: 10.35940/ijrte.A5854.0510121.

Hu, G. *et al.* (2016) ‘Classification of wine quality with imbalanced data’, in *2016 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, pp. 1712–1217. doi: 10.1109/ICIT.2016.7475021.

Nebot, À., Mugica, F. and Escobet, A. (2015) ‘Modeling Wine Preferences from Physicochemical Properties using Fuzzy Techniques’, in *Proceedings of the 5th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*. SCITEPRESS - Science and and Technology Publications, pp. 501–507. doi: 10.5220/0005551905010507.

O’Neil, C. and Schutt, R. (2013) ‘Doing Data Science’, in. Sebastopol,: O’Reilly Media, Inc., p. 77.

*The caret Package* (no date). Available at: https://topepo.github.io/caret/pre-processing.html (Accessed: 20 October 2021).

**Source code**

# install packages

install.packages("ggplot2")

install.packages("tibble")

install.packages("caret")

install.packages("gridExtra")

install.packages("huxtable")

install.packages("data.table")

install.packages("performanceEstimation ")

install.packages("ROCR ")

install.packages("grid ")

# call packages

library(ggplot2)

library(tibble)

library(caret)

library(gridExtra)

library(huxtable)

library(data.table)

library(performanceEstimation)

library(ROCR)

library(grid)

#get data

winequality <- read.table("https://raw.githubusercontent.com/GiselleVicatos/Sunderland\_CETM72/main/winequalityN.csv",

header=TRUE, stringsAsFactors = TRUE, sep=',')

str(winequality)

table(winequality$type)

# Barplot of wine type

ggplot(data = winequality) +

geom\_bar(aes(type, fill = type)) +

ggtitle("Prevalence of Red and White Wines") +

theme(

plot.title = (element\_text(hjust = 0.5,color="black", size =30, face="bold" )),

axis.title.x = element\_text(color="black", size=30, face="bold"),

axis.title.y = element\_text(color="black", size=30, face="bold"),

text = element\_text(size = 30))

#### Data Cleaning

###remove null values

#says how many null values there are

sum(is.na(winequality))

#shows where NaN values are

summary(winequality)

#takes out row with null values and assigns a new dataset

wine3 <- na.omit(winequality)

#no NAN values

sum(is.na(wine3))

head(wine3)

###replace existing grading system with a binary high and low evaluation

wine3$quality[wine3$quality==3] <- 0

wine3$quality[wine3$quality==4] <- 0

wine3$quality[wine3$quality==5] <- 0

wine3$quality[wine3$quality==6] <- 1

wine3$quality[wine3$quality==7] <- 1

wine3$quality[wine3$quality==8] <- 1

wine3$quality[wine3$quality==9] <- 1

#convert wine quality to a factor

wine3$quality = as.factor(wine3$quality)

#check conversion

str(wine3)

table(wine3$quality)

###separate into red and white wine

red\_wine <- wine3[which(wine3$type == "red"),]

white\_wine <- wine3[which(wine3$type == "white"),]

#check seperation

table(red\_wine$quality)

table(white\_wine$quality)

# Barplot of quality for red wine

qr <- ggplot(data = red\_wine) +

geom\_bar(aes(quality, fill = quality)) +

ggtitle("Quality Distribution for Red Wine") +

scale\_fill\_manual(values=c('purple','orange'))+

theme(

plot.title = (element\_text(hjust = 0.5,color="black", size =25, face="bold" )),

axis.title.x = element\_text(color="black", size=25, face="bold"),

axis.title.y = element\_text(color="black", size=25, face="bold"),

text = element\_text(size = 25))

qr

# Barplot of quality for white wine

qw <-ggplot(data = white\_wine) +

geom\_bar(aes(quality, fill = quality)) +

ggtitle("Quality Distribution for White Wine") +

scale\_fill\_manual(values=c('purple','orange'))+

theme(

plot.title = (element\_text(hjust = 0.5,color="black", size =25, face="bold" )),

axis.title.x = element\_text(color="black", size=25, face="bold"),

axis.title.y = element\_text(color="black", size=25, face="bold"),

text = element\_text(size = 25))

qw

#put both bars on one plot

grid.arrange(qr, qw,

ncol = 2, nrow = 1)

### Check for outliers

##create a boxplot

#take out the quality column

wine4 <- wine3[-13]

head(wine4)

#plot the boxplots

wine4.m <- reshape2::melt(wine4, id.var = "type") # use melt to restructure data so that I can plot

ggplot(data = wine4.m , aes(x=variable, y=value)) +

geom\_boxplot(aes(fill=type)) +

facet\_wrap( ~ variable, scales="free") +

xlab("Features") + ylab("Value") + ggtitle("Distribution of Values by Feature for White and Red Wine") +

theme(

plot.title = (element\_text(hjust = 0.5,color="black", size =22, face="bold" )),

axis.title.x = element\_text(color="black", size=22, face="bold"),

axis.title.y = element\_text(color="black", size=22, face="bold"),

text = element\_text(size = 22))

##function to remove outliers

outlierReplace <- function(dataframe, cols, rows, newValue = NA) {

if (any(rows)) {

set(dataframe, rows, cols, newValue)}}

##remove outliers of selected features

#make a subset with only the features

ro\_red\_wine <- subset(red\_wine, select=-c(1,13))

ro\_white\_wine <-subset(white\_wine, select=-c(1,13))

head(ro\_red\_wine)

head(ro\_white\_wine)

# replace identified outliers with null values

#from white wine - residual sugar

outlierReplace(ro\_white\_wine, "residual.sugar", which(ro\_white\_wine$residual.sugar > 40), NA)

boxplot(ro\_white\_wine$residual.sugar)

# from red wine - chloride

outlierReplace(ro\_red\_wine, "chlorides", which(ro\_red\_wine$chlorides > 0.55), NA)

boxplot(ro\_red\_wine$chlorides)

#from white wine - free sulfur dioxide

outlierReplace(ro\_white\_wine, "free.sulfur.dioxide", which(ro\_white\_wine$free.sulfur.dioxide > 250), NA)

boxplot(ro\_white\_wine$free.sulfur.dioxide)

#from white wine - density

outlierReplace(ro\_white\_wine, "density", which(ro\_white\_wine$density > 1.03), NA)

boxplot(ro\_white\_wine$density)

#put quality column back into subset data set for each wine type

ro\_red\_wine2 <- data.frame(ro\_red\_wine, red\_wine[13])

head(ro\_red\_wine2)

ro\_white\_wine2 <- data.frame(ro\_white\_wine, white\_wine[13])

head(ro\_white\_wine2)

#take out null values

sum(is.na(ro\_red\_wine2))

sum(is.na(ro\_white\_wine2))

#takes out row with null values and assigns a new dataset

new\_red\_wine <- na.omit(ro\_red\_wine2)

sum(is.na(new\_red\_wine))

head(new\_red\_wine)

new\_white\_wine <- na.omit(ro\_white\_wine2)

sum(is.na(new\_white\_wine))

head(new\_white\_wine)

### data transformation

#normalize data

normalize <- function(x) {

return ((x - min(x)) / (max(x) - min(x)))}

# red wine

new\_red\_wine$fixed.acidity <- normalize(new\_red\_wine$fixed.acidity)

new\_red\_wine$volatile.acidity <- normalize(new\_red\_wine$volatile.acidity)

new\_red\_wine$citric.acid <- normalize(new\_red\_wine$citric.acid)

new\_red\_wine$residual.sugar <- normalize(new\_red\_wine$residual.sugar)

new\_red\_wine$chlorides <- normalize(new\_red\_wine$chlorides)

new\_red\_wine$free.sulfur.dioxide <- normalize(new\_red\_wine$free.sulfur.dioxide)

new\_red\_wine$total.sulfur.dioxide <- normalize(new\_red\_wine$total.sulfur.dioxide)

new\_red\_wine$density <- normalize(new\_red\_wine$density)

new\_red\_wine$pH <- normalize(new\_red\_wine$pH)

new\_red\_wine$sulphates <- normalize(new\_red\_wine$sulphates)

new\_red\_wine$alcohol <- normalize(new\_red\_wine$alcohol)

head(new\_red\_wine)

# white wine

new\_white\_wine$fixed.acidity <- normalize(new\_white\_wine$fixed.acidity)

new\_white\_wine$volatile.acidity <- normalize(new\_white\_wine$volatile.acidity)

new\_white\_wine$citric.acid <- normalize(new\_white\_wine$citric.acid)

new\_white\_wine$residual.sugar <- normalize(new\_white\_wine$residual.sugar)

new\_white\_wine$chlorides <- normalize(new\_white\_wine$chlorides)

new\_white\_wine$free.sulfur.dioxide <- normalize(new\_white\_wine$free.sulfur.dioxide)

new\_white\_wine$total.sulfur.dioxide <- normalize(new\_white\_wine$total.sulfur.dioxide)

new\_white\_wine$density <- normalize(new\_white\_wine$density)

new\_white\_wine$pH <- normalize(new\_white\_wine$pH)

new\_white\_wine$sulphates <- normalize(new\_white\_wine$sulphates)

new\_white\_wine$alcohol <- normalize(new\_white\_wine$alcohol)

head(new\_white\_wine)

###split data into train and test set

#red wine

set.seed(5)

index <- sample(nrow(new\_red\_wine), 0.8 \* nrow(new\_red\_wine))

red\_winetrain <- new\_red\_wine[index,]

red\_winetest <- new\_red\_wine[-index,]

#white wine

set.seed(5)

index <- sample(nrow(new\_white\_wine), 0.8 \* nrow(new\_white\_wine))

white\_winetrain <- new\_white\_wine[index,]

white\_winetest <- new\_white\_wine[-index,]

#check quality distribution of train data

table(red\_winetrain$quality)

table(white\_winetrain$quality)

###oversampling using SMOTE

#white wine

white\_winetrain\_re <- smote(quality~., white\_winetrain, perc.over = 1, perc.under = 2, k=5)

table(white\_winetrain\_re$quality)

#red wine

red\_winetrain\_re <- smote(quality~., red\_winetrain, perc.over = 0.15, perc.under = 7.73, k=5)

table(red\_winetrain\_re$quality)

###FUNCTION

# create tibble for results

quality\_results <- tibble(wine\_type = NA,

Model = NA,

Accuracy = NA,

Sensitivity\_or\_Recall = NA,

Specificity = NA,

Precision = NA,

F1\_Score = NA,

AUC = NA)

#define function

alorgithm\_predictions <- function(model, train, test){

{ctrl <- trainControl(method = "repeatedcv", number =5, repeats= 3)

set.seed(333)}

if (model=="knn"){

fit.cv <- train(quality~., data = train, method =model,

trControl = ctrl, preProcess= c("center", "scale"), tuneLength =5)

variable\_importance\_knn <<- varImp(fit.cv, scale = TRUE)}

else if (model=="rf"){

fit.cv <- train(quality~., data = train, method =model,

trControl = ctrl, preProcess= c("center", "scale"), tuneLength =5, metric= "Accuracy")

variable\_importance\_rf <<- varImp(fit.cv, scale = TRUE)}

else if (model=="svmRadial"){

fit.cv <- train(quality~., data= train, method=model,

trControl=ctrl, tuneLength =5, preProcess = c("center", "scale"))

variable\_importance\_svmRadial <<- varImp(fit.cv, scale = TRUE)}

else if (model =="rpart"){

fit.cv <- train(quality~., data = train, method =model,

trControl = ctrl, preProcess= c("center", "scale"), tuneLength =5)

variable\_importance\_rpart <<- varImp(fit.cv, scale = TRUE)}

#prediction

pred <- predict(fit.cv, test[-12])

#confusion matrix

confM <- confusionMatrix(table(pred, test$quality), positive='1')

# Accuracy

acc <- mean(pred == test$quality)

# Sensitivity/Recall

sen <- caret::confusionMatrix(pred, test$quality, positive='1')$byClass["Sensitivity"]

# Specificity

spe <- caret::confusionMatrix(pred, test$quality, positive='1')$byClass["Specificity"]

# Pos Pred Value/Precision

pos <- caret::confusionMatrix(pred, test$quality, positive='1')$byClass["Pos Pred Value"]

# F1 score

F1 <- 2\* (pos\*sen/(pos+sen))

#ROC and AUC

roc\_pred <- prediction(as.numeric(pred), as.numeric(test$quality))

roc\_perf <- performance(roc\_pred , "tpr" , "fpr")

auc\_ROCR <- performance(roc\_pred, measure = "auc")

auc\_ROCR <- auc\_ROCR@y.values[[1]]

#Red/white wine label

red\_white\_lable <- as.character(substitute(train))

red\_white\_lable2 <- substr(red\_white\_lable, 1,1)

# gets performance measures for each metric for each model

quality\_results <<- rbind(quality\_results,

tibble(wine\_type = red\_white\_lable2,

Model = model,

Accuracy = acc,

Sensitivity\_or\_Recall = sen,

Specificity = spe,

Precision = pos,

F1\_Score = F1,

AUC = auc\_ROCR))

return(confM)

}

###define the models to be used

models <- c("rpart", "knn", "rf", "svmRadial")

###call the function for white wine

for (i in 1:4){

print(alorgithm\_predictions(models[i] , white\_winetrain\_re, white\_winetest))}

###plot the variable of importance for white wine per model

aa <- plot(variable\_importance\_rpart, cex = 1, main= "DT model")

bb <- plot(variable\_importance\_knn, cex = 1, main= "KNN model")

cc <- plot(variable\_importance\_rf, cex = 1, main= "RF model")

dd <- plot(variable\_importance\_svmRadial, cex = 1,main= "SVM model")

#put all the white wine plots on one grid

grid.arrange(aa, bb, cc, dd,

ncol = 2, nrow = 2, top = textGrob("Variables of Importance for White Wine per Model",gp=gpar(fontsize=20)))

###call the function for red wine

for (i in 1:4){

print(alorgithm\_predictions(models[i] , red\_winetrain\_re, red\_winetest))}

###plot the variable of importance for red wine per model

ee <- plot(variable\_importance\_rpart, cex = 1, main= "DT model")

ff <- plot(variable\_importance\_knn, cex = 1, main= "KNN model")

gg <- plot(variable\_importance\_rf, cex = 1, main= "RF model")

hh <- plot(variable\_importance\_svmRadial, cex = 1,main= "SVM model")

#put all the red wine plots on one grid

grid.arrange(ee, ff, gg, hh,

ncol = 2, nrow = 2, top = textGrob("Variables of Importance for Red Wine per Model",gp=gpar(fontsize=20)))

### create table of metric results for white and red wine

as\_hux(quality\_results) %>%

# Add colour to red and white wine label

set\_text\_color(3:6, 1, "blue") %>%

set\_text\_color(7:10, 1, "red") %>%

# Colour code the models

set\_text\_color(3, 2, "purple") %>%

set\_text\_color(4, 2, "orange") %>%

set\_text\_color(5, 2, "green") %>%

set\_text\_color(6, 2, "black") %>%

set\_text\_color(7, 2, "purple") %>%

set\_text\_color(8, 2, "orange") %>%

set\_text\_color(9, 2, "green") %>%

set\_text\_color(10, 2, "black") %>%

# Format numbers

set\_number\_format(row= -1, value = 3) %>%

# Format header row

set\_top\_border(row = 1, col = everywhere) %>%

set\_bottom\_border(row = 1, col = everywhere) %>%

set\_bold(row = 1, col = everywhere) %>%

# Title

set\_caption("Model Performance Results") %>%

set\_position(value = "center")

### Visualize the confusion matrix tables for white and red wine per model

#set up target and predict classes

TargetClass <- factor(c("0", "0", "1", "1"))

PredictClass <- factor(c("0", "1", "0", "1"))

#get the TN, FP, FN, TP values for each model from the output of the function.

#white wine

white\_svm <- c(246, 85, 167, 476)

white\_rf <- c(270, 61, 115, 528)

white\_knn <- c(243, 88, 161, 482)

white\_rpart <- c(239, 92, 185, 458)

#red wine

red\_svm <- c(121, 27, 48, 123)

red\_rf <- c(131, 17, 45, 126)

red\_knn <- c(103, 45, 39, 132)

red\_rpart <- c(119, 29, 56, 115)

#setup the data frames

df\_white\_rpart <- data.frame(TargetClass, PredictClass, white\_rpart)

df\_white\_knn <- data.frame(TargetClass, PredictClass, white\_knn)

df\_white\_rf <- data.frame(TargetClass, PredictClass, white\_rf)

df\_white\_svm <- data.frame(TargetClass, PredictClass, white\_svm)

df\_red\_rpart <- data.frame(TargetClass, PredictClass, red\_rpart)

df\_red\_knn <- data.frame(TargetClass, PredictClass, red\_knn)

df\_red\_rf <- data.frame(TargetClass, PredictClass, red\_rf)

df\_red\_svm <- data.frame(TargetClass, PredictClass, red\_svm)

#plot the confusion matrix using ggplot

#white wine DT model

a <- ggplot(data = df\_white\_rpart, mapping = aes(x = TargetClass, y = PredictClass)) +

geom\_tile(aes(fill = white\_rpart), colour = "white") +

geom\_text(aes(label = sprintf("%1.0f", white\_rpart)), vjust = 1, size = 8 ) +

scale\_fill\_gradient(low = "white", high = "steelblue", name = "Freq" ) +

theme\_bw() +

ggtitle("DT Model") +

theme(text = element\_text(size = 20), plot.title = element\_text(hjust = 0.5))

#white wine KNN model

b <- ggplot(data = df\_white\_knn, mapping = aes(x = TargetClass, y = PredictClass)) +

geom\_tile(aes(fill = white\_knn), colour = "white") +

geom\_text(aes(label = sprintf("%1.0f", white\_knn)), vjust = 1, size = 8 ) +

scale\_fill\_gradient(low = "white", high = "steelblue", "Freq") +

theme\_bw() +

ggtitle("KNN Model") +

theme(text = element\_text(size = 20), plot.title = element\_text(hjust = 0.5))

#white wine RF model

c <- ggplot(data = df\_white\_rf, mapping = aes(x = TargetClass, y = PredictClass)) +

geom\_tile(aes(fill = white\_rf), colour = "white") +

geom\_text(aes(label = sprintf("%1.0f", white\_rf)), vjust = 1, size = 8 ) +

scale\_fill\_gradient(low = "white", high = "steelblue", name = "Freq") +

theme\_bw() +

ggtitle("RF Model") +

theme(text = element\_text(size = 20), plot.title = element\_text(hjust = 0.5))

#white wine SVM model

d <- ggplot(data = df\_white\_svm, mapping = aes(x = TargetClass, y = PredictClass)) +

geom\_tile(aes(fill = white\_svm), colour = "white") +

geom\_text(aes(label = sprintf("%1.0f", white\_svm)), vjust = 1, size = 8 ) +

scale\_fill\_gradient(low = "white", high = "steelblue", name = "Freq") +

theme\_bw() +

ggtitle("SVM Model") +

theme(text = element\_text(size = 20), plot.title = element\_text(hjust = 0.5))

#put all plots into one grid for white wine

grid.arrange(a, b, c, d,

ncol = 2, nrow = 2, top = textGrob("Confusion Matrix for White Wine Models",gp=gpar(fontsize=30)))

#red wine DT model

e <- ggplot(data = df\_red\_rpart, mapping = aes(x = TargetClass, y = PredictClass)) +

geom\_tile(aes(fill = red\_rpart), colour = "white") +

geom\_text(aes(label = sprintf("%1.0f", red\_rpart)), vjust = 1, size = 8 ) +

scale\_fill\_gradient(low = "white", high = "steelblue", name = "Freq") +

theme\_bw() +

ggtitle("DT Model") +

theme(text = element\_text(size = 20), plot.title = element\_text(hjust = 0.5))

#red wine KNN model

f <- ggplot(data = df\_red\_knn, mapping = aes(x = TargetClass, y = PredictClass)) +

geom\_tile(aes(fill = red\_knn), colour = "white") +

geom\_text(aes(label = sprintf("%1.0f", red\_knn)), vjust = 1, size = 8 ) +

scale\_fill\_gradient(low = "white", high = "steelblue", name = "Freq") +

theme\_bw() +

ggtitle("KNN Model") +

theme(text = element\_text(size = 20), plot.title = element\_text(hjust = 0.5))

#red wine RF model

g <- ggplot(data = df\_red\_rf, mapping = aes(x = TargetClass, y = PredictClass)) +

geom\_tile(aes(fill = red\_rf), colour = "white") +

geom\_text(aes(label = sprintf("%1.0f", red\_rf)), vjust = 1, size = 8 ) +

scale\_fill\_gradient(low = "white", high = "steelblue", name = "Freq") +

theme\_bw() +

ggtitle("RF Model") +

theme(text = element\_text(size = 20), plot.title = element\_text(hjust = 0.5))

#red wine SVM model

h <- ggplot(data = df\_red\_svm, mapping = aes(x = TargetClass, y = PredictClass)) +

geom\_tile(aes(fill = red\_svm), colour = "white") +

geom\_text(aes(label = sprintf("%1.0f", red\_svm)), vjust = 1, size = 8 ) +

scale\_fill\_gradient(low = "white", high = "steelblue", name = "Freq") +

theme\_bw() +

ggtitle("SVM Model") +

theme(text = element\_text(size = 20), plot.title = element\_text(hjust = 0.5))

#put all plots into one grid for red wine

grid.arrange(e, f, g, h,

ncol = 2, nrow = 2, top = textGrob("Confusion Matrix for Red Wine Models",gp=gpar(fontsize=30)))