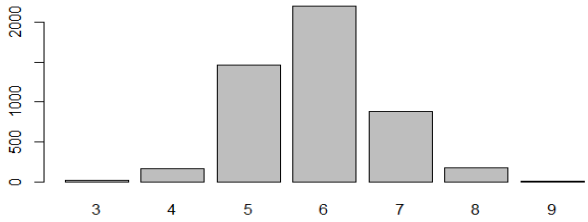
Task 2: Wine Rating Prediction

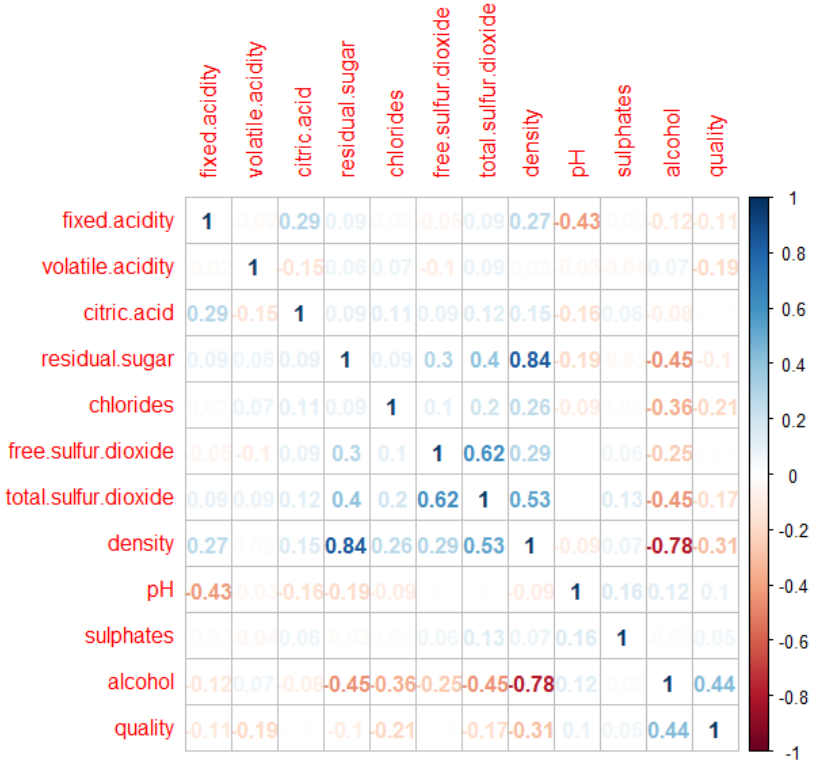
Data & Pre-Processing

The dataset used is the Wine Quality Ratings and Chemicals-White dataset, which contains information regarding the various features of the wine, and its overall quality. We have 11 independent variables, and one dependent variable, which is the quality, which ranges from 0 (very bad) and 10 (very excellent). To know the range of quality values, we have plotted the quality variable in a bar chart.



As we can see, the majority values lie in the intervals 5-7. We have used feature engineering to enhance the dataset, by grouping the quality variables into a new column called quality_1.

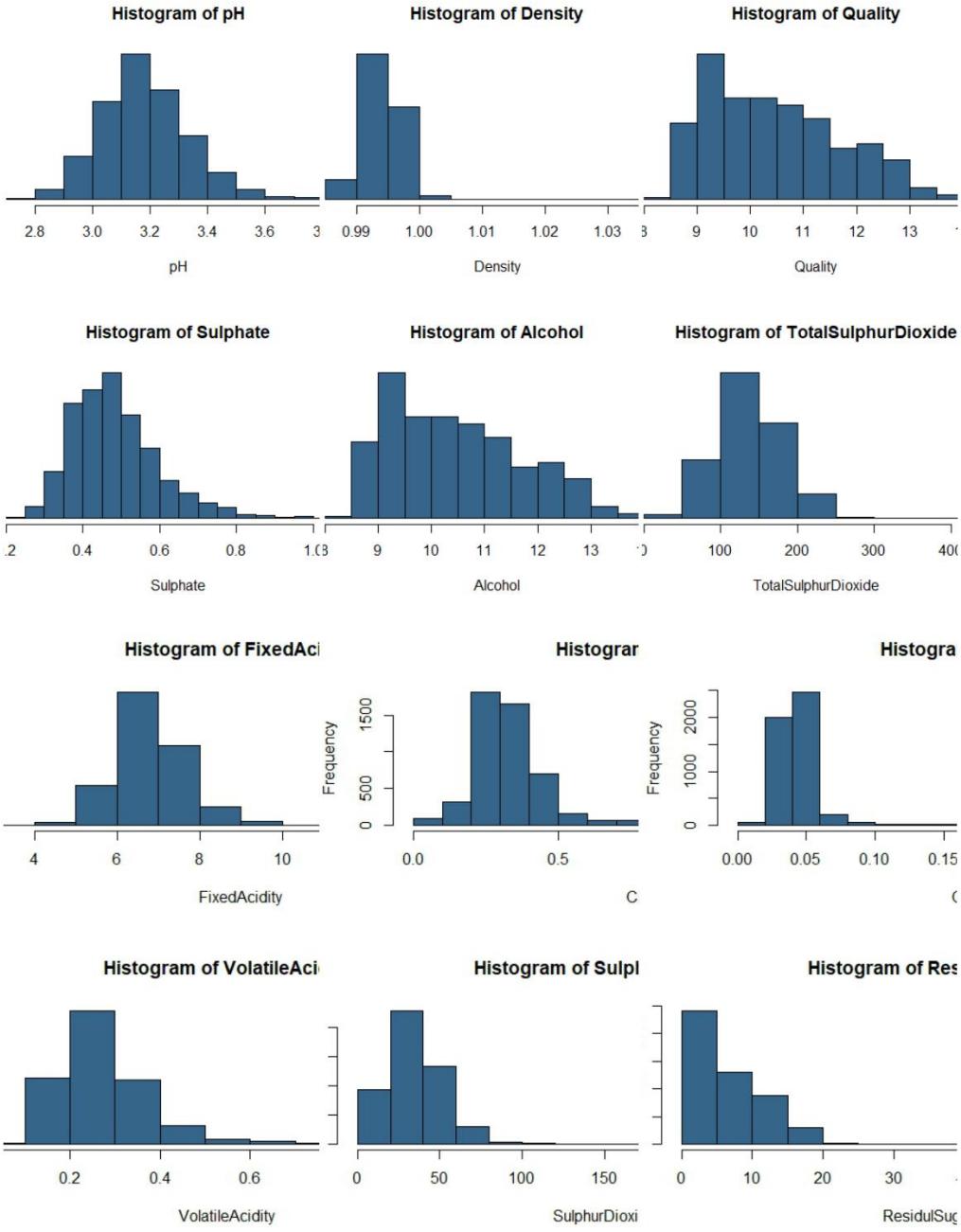
- For values with quality under 6, we assigned value: Bad
- For values with quality equal to 6, we assigned: Average
- For values with quality equal above 6, we assigned: Excellent



Above we have plotted a heat map representing the correlation value of all the features. Concerning our target variable – 'quality', alcohol is having the highest degree of correlation and then the density while citric acid and free Sulphur dioxide have the least correlation and we will also we build models by removing the least correlated feature.

Algorithm & Feature Selection

Below are the histograms showing the distribution of values for our features. Most of the features do not have a normal distribution which makes hard to develop correct prediction models. Instance, features like density (Fig. 2) have most the values centred around medium, and residual sugar (Fig. 12) have values centred at 1st quartile. Similarly, from the are below graph we can predict that most of the data is having non-normalized distribution. For this, we have done normalised scaling to maintain a certain degree of level among in our dataset.



Algorithm Selection -
We are using Random forest algorithm and Stochastic Gradient Boosting for classification. Both Random forest and Gradient Boosting are a supervised learning algorithm.
Random Forest creates a forest with some trees. It is selected as it deals with
Over fitting on its own.
Gradient boosting model is an ensemble for regression and classification problems, it produces a prediction model in the form of weak prediction models of ensembles, typically decision trees. We have selected gradient boosting because it builds one tree at a time and each new tree helps to correct errors made by the previously trained tree. With each tree added, the model becomes even more expressive.

Evaluation:

On training the model and predicting it on the test data, we get the following result:

From the confusion matrix, we can see the comparison between actual and predicted values, how many the model fit correctly, and how many it classified incorrectly. Random Forest model did well, with an accuracy of 72.55% while Gradient Boosting produced an accuracy of 64.80%. The accuracy of both the models is increased because of the inclusion of feature engineering which allowed for more precision in classification, as opposed to running it without feature engineering which then gave an accuracy of 53% and 55%.

Random Forest Confusion Matrix :

Prediction	Average	Bad	Excellent
Average	497	138	133
Bad	97	349	6
Excellent	47	10	193

Average Accuracy: 72.55%

Stochastic Gradient Boosting Confusion Matrix :

Prediction	Average	Bad	Excellent
Average	608	230	182
Bad	151	420	13
Excellent	123	17	216

Average Accuracy: 63.47%

Random Forest Statistics:

Class	Average	Bad	Excellent
Sensitivity	0.7754	0.7022	0.5813
Specificity	0.6731	0.8941	0.9499

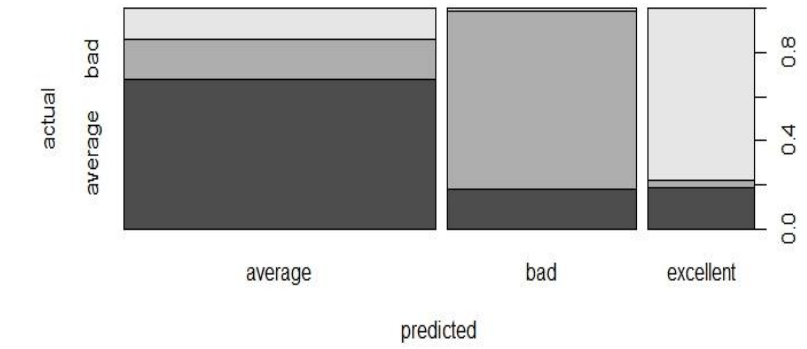
Gradient Boosting Statistics:

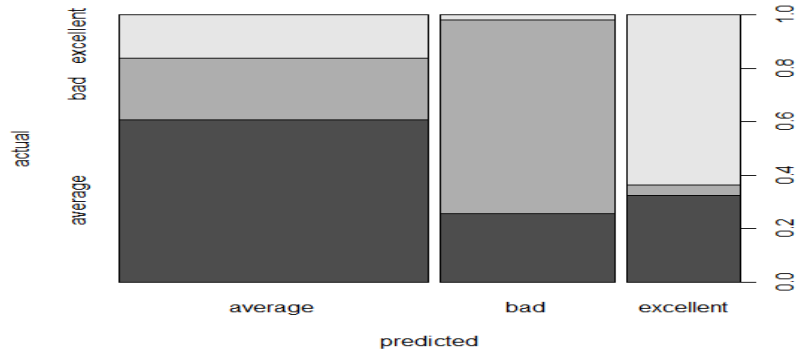
Class	Average	Bad	Excellent
Sensitivity	0.6961	0.6282	0.5266
Specificity	0.6317	0.8786	0.9122

The above two table show sensitivity is given by the number of correct positive predictions divided by the total number of positives, with one being the highest and 0 being the lowest. For Random Forest we have a good rate of sensitivity for Average (0. 7754) and Bad (0. 7022) wines, and a moderate rate for Excellent (0.5813). While using Gradient Boost we have received a moderate rate of sensitivity for Average (0.6961) and Bad (0.6282), and for Excellent it is (0.5266).

Specificity is calculated as the number of correct negative predictions divided by the total number of negatives. Again, with random forest, we have a decent rate of specificity for our Average wines (0. 6731), and excellent specificity for our Bad (0.8941) and Excellent (0.9499) wines. While with gradient boost as well we have excellent specificity for Bad (0.8786) and Excellent (0.9122), and for Average (0.6317) we have decent value. From this, we can see that both the models did an excellent job at calculating if the wine was Average or Bad, and it did an excellent job at calculating if the wine was not Bad, or not Excellent where the performance of random forest model superseded gradient boost model.

The box-plot of actual vs predicted values for Random forest and Gradient Boosting:





We can see that the model predicts a large amount of data correctly, and has some outliers, especially in the prediction regarding Average wine. This was a result of the data being centred heavily on the Average quality (6) as we can see from the first chart. This makes the classification a little biased since most of the training data were centred around one point.

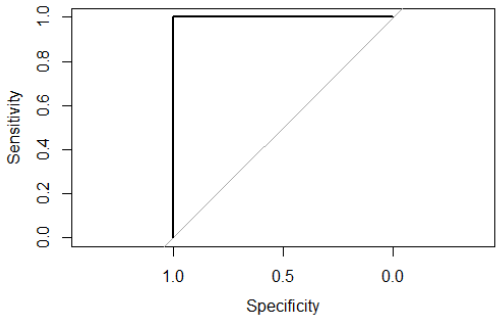
Conclusion:

We use 10-fold validation for both the algorithms to get the best accuracy. All the independent variables are selected for the analysis, and we have one output variable quality_1. When we removed few of the less correlated features like citric acid and free sulphur dioxide, there was a slight decrease in model accuracies.
From our overall analysis, we can conclude that the Random Forest algorithm does an excellent job of classifying the wine quality, with an accuracy of 75%.

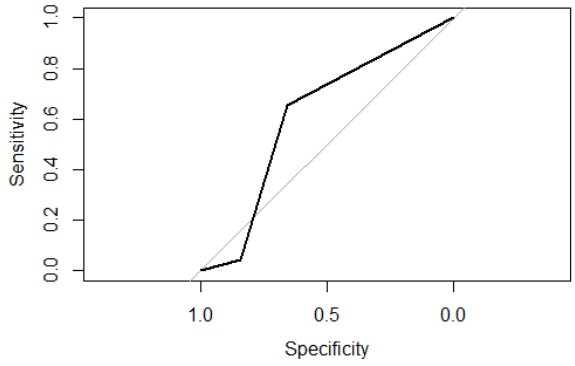
ROC Curve :

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate(FPR) at various threshold settings. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

Random Forest –



Gradient Boosting –



Limitations :

The amount of computational time increases as the number of trees in the random forest model increased.

Appendix :

1. Random Forest model implementation and reporting part was done by Siddharth Sheshadri (17310763).
2. Gradient Boost model and reporting was done by Sujay Talesara (17306775).
3. 3rd team member Kanishka Venishetty (17311875)
4. Total hours consumed for completing the assignment are 30 hours.

Source Code :

```
library(randomForest)

library(mlbench)

library(caret)

library(caretEnsemble)

library(lattice)

library(ggplot2)

library(corrplot)

library(dplyr)

library(ggplot2)

library(gbm)

library(ROCR)

library(pROC)

data.frame <- data.frame(winequality.white)

sum(is.na(data.frame))

# Distribution plot ----------------------------------------------------------

barplot(table(data.frame$quality))

cor(data.frame$density,data.frame$quality)

# Heatplot-------------------------------------------------------------

cor.white <- cor(data.frame, use='pairwise')

corrplot(cor.white, method = 'number')

# Feature Engineering ---------------------------------------

data.frame <- data.frame[-3]

data.frame <- data.frame[-5]

data.frame$taste <- ifelse(data.frame$quality < 6, 'bad', 'excellent')

data.frame$taste[data.frame$quality == 6] <- 'average'

data.frame$taste <- as.factor(data.frame$taste)
```

```
# Training & Test Data ------------------------------------------------------------
set.seed(123)
samp <- sample(nrow(data.frame), 0.6 * nrow(data.frame))
train <- data.frame[samp, ]
test <- data.frame[-samp, ]
# Random Forest Model -------------------------------------------------------------
fit.rf <- randomForest(taste ~ ., data = train)
plot(fit.rf$importance)
# Stochastic Gradient Boosting Model ----------------------------------------------------------------
metric <- "Accuracy"
control <- trainControl(method="repeatedcv", number=10, repeats=3)
fit.gbm <- train(taste ~ .- quality, data=train, method="gbm",
            metric=metric, trControl=control, verbose=FALSE)
# Prediction ----------------------------------------------------------------
pred.rf <- predict(fit.rf, newdata = test)
pred.gbm <- predict(fit.gbm , newdata = test)
# Accuracy ------------------------------------------------------------------
confusionMatrix(pred.rf,test$taste)
confusionMatrix(pred.gbm,test$taste)
# Plotting ------------------------------------------------------------------
barplot(pred.rf, test$taste,xlab="predicted",ylab="actual")
table(pred.rf, test$quality)
plot(pred.gbm, test$taste,xlab="predicted",ylab="actual")
table(pred.gbm, test$taste)
# ROC Curve -----------------------------------------------------------------
plot(roc(test$taste,as.numeric(pred.rf)))
plot(roc(test$taste,as.numeric(pred.gbm)))
```