UNIVERSITÀ DEGLI STUDI
DI MILANO

Probabilistic Modelling Final Exam

May, 2020 – 2021

Nazli Begum CIRPANLI

942345

Table of Contents

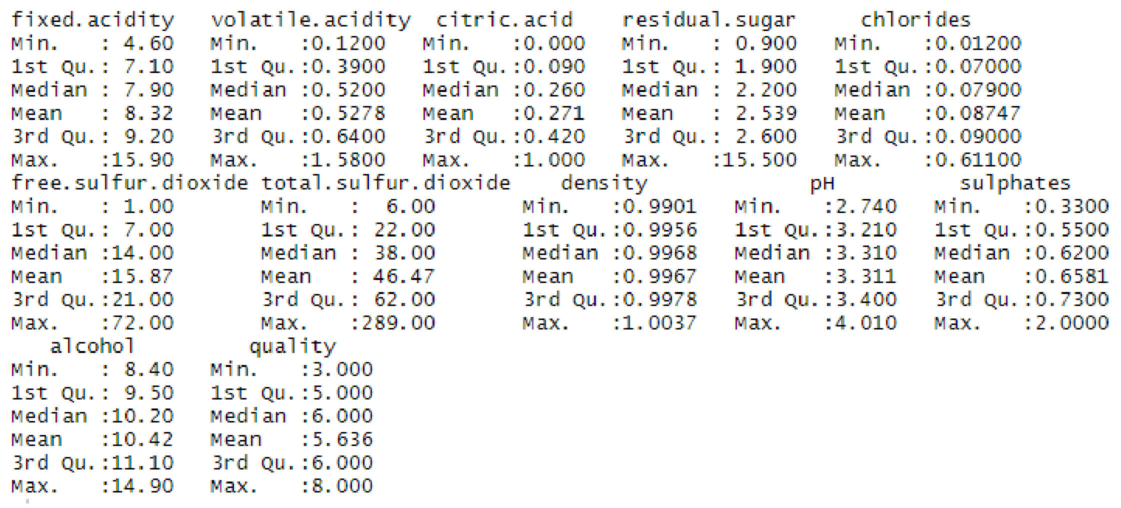
AIM OF THE ANALYSIS 1
DATASET DESCRIPTION 1
METHODOLOGY 2
RESULTS 4
  Inference 7
CONCLUSION 8
REFERENCES 9
APPENDIX: R CODE 10

AIM OF THE ANALYSIS

The aim of this analysis is to construct a Bayesian network with the variables in order to build a model to learn which affect the red wine quality the most and to predict the level of quality. Moreover, in addition to examining the potential dependencies and independencies among variables, priorly known relations between variables are also checked to see if they hold in this dataset.

DATASET DESCRIPTION

The dataset is a sample of red "vinho verde" of Portugal. It is produced in the northwest region, Minho, of Portugal and popular for its freshness. This dataset contains 1599 observations and 12 attributes without any missing values. These attributes are several physicochemical tests which are done as part of the wine certification process and quality sensory scores given by the assessors. Quality scores represent the quality of wine and range between 0 and 10. All attributes, except quality, are continuous.



```
 fixed.acidity   volatile.acidity  citric.acid    residual.sugar     chlorides
 Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900   Min.   :0.01200
 1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900   1st Qu.:0.07000
 Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200   Median :0.07900
 Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539   Mean   :0.08747
 3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600   3rd Qu.:0.09000
 Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500   Max.   :0.61100
 free.sulfur.dioxide total.sulfur.dioxide    density             pH          sulphates
 Min.   : 1.00       Min.   :  6.00       Min.   :0.9901   Min.   :2.740   Min.   :0.3300
 1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956   1st Qu.:3.210   1st Qu.:0.5500
 Median :14.00       Median : 38.00       Median :0.9968   Median :3.310   Median :0.6200
 Mean   :15.87       Mean   : 46.47       Mean   :0.9967   Mean   :3.311   Mean   :0.6581
 3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978   3rd Qu.:3.400   3rd Qu.:0.7300
 Max.   :72.00       Max.   :289.00       Max.   :1.0037   Max.   :4.010   Max.   :2.0000
    alcohol          quality
 Min.   : 8.40   Min.   :3.000
 1st Qu.: 9.50   1st Qu.:5.000
 Median :10.20   Median :6.000
 Mean   :10.42   Mean   :5.636
 3rd Qu.:11.10   3rd Qu.:6.000
 Max.   :14.90   Max.   :8.000
```

Brief descriptions for each variable as follows:

Fixed acidity: Mainly, tartaric, malic, and citric acids.

Volatile acidity: Acetic acid, which is commonly found in vinegar. Excessive amounts can lead to wine fault.

Citric acid: Usually used as an acid supplement during the fermentation process to increase the total acidity of the wine**.**

Residual sugar: The amount of sugar left in the wine after the fermentation process.

Chlorides: The amount of salt in the wine.

Free sulfur dioxide: Sulfur dioxide available in the active, molecular form to help protect the wine from oxidation and spoilage.[1]

Total sulfur dioxide: The total amount of sulfur dioxide which consists of the free sulfur dioxide and the sulfur dioxide bound to other chemicals in the wine.[1]

Density: Mass divided by volume. Density is measured by the weight in grams (g) of each milliliter (mL) of liquid.

pH: Measure of acidity strength. Most wines have a pH between 3 and 4.Lower pH values indicate higher acidity. However, it is also possible to find wines both with a high pH value and high acidity.[2]

Sulphates: Additive used in order to protect wine from oxidation.

Alcohol: Percentage of alcohol amount in the wine.

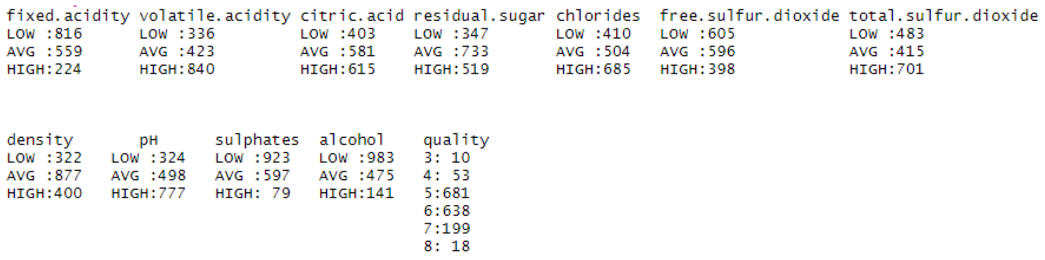
Quality : Integer variable ranges from 0 – very bad- to 10 –excellent – . It represents the scores given by the wine experts. It is the target variable. In this sample quality levels of 1,2,9 and 10 do not exist.

METHODOLOGY

The methodology applied in this analysis is Bayesian network for discrete data. Since all the variables except quality are continuous, discretizing the variables is the first step. There are several different approaches to discretization. In this analysis Hartemink method is employed for it preserves pairwise dependencies between the variables and tries to keep the information loss as low as possible. [3] Moreover, it is not sensitive to outliers.

Each of the eleven variables was discretized into three different categories (“HIGH”, “AVG”,”LOW”) by taking their quantiles into consideration. The target variable “quality” was only factorized and left with six different levels.

Summary of the discretized data is below:



```
fixed.acidity volatile.acidity citric.acid residual.sugar chlorides  free.sulfur.dioxide total.sulfur.dioxide
LOW :816      LOW :336         LOW :403    LOW :347       LOW :410   LOW :605            LOW :483
AVG :559      AVG :423         AVG :581    AVG :733       AVG :504   AVG :596            AVG :415
HIGH:224      HIGH:840         HIGH:615    HIGH:519       HIGH:685   HIGH:398            HIGH:701


density      pH       sulphates  alcohol    quality
LOW :322  LOW :324  LOW :923   LOW :983   3: 10
AVG :877  AVG :498  AVG :597   AVG :475   4: 53
HIGH:400  HIGH:777  HIGH: 79   HIGH:141   5:681
                                          6:638
                                          7:199
                                          8: 18
```

After discretizing and a quick examination of the variables, next step is to learn the structure.

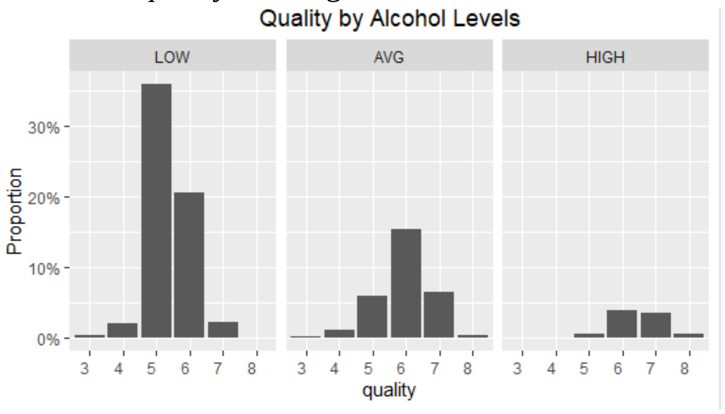
Three different structure learning algorithms were used in this study: Grow-shrink, Hill-climbing and Max-Min Hill Climbing. They belong to different categories according to the way they learn the structure. These categories are Constraint-based Learning Algorithms, Score-based Learning Algorithms, Hybrid Learning Algorithms, respectively.

- Grow-shrink is also called Grow-Shrink Markov Blanket. It constructs the Markov Blankets of the variables with conditional independence tests.

- Hill-climbing is one of the greedy search algorithms. It is based on optimizing AIC or BIC score in the structure building process.
- Max-Min Hill Climbing is a hybrid algorithm which combines Hill-Climbing and Max-Min Parents and Children algorithms.

The structure of the network was learned first from the data without any prior knowledge and then with a priori knowledge by each of these three different learning algorithms with the purpose of finding the most suitable one to the data. A priori knowledge was obtained by some preliminary research about the ingredients of red wines. Based on this research following dependencies were established:

- **Alcohol → Quality:** Alcohol level affects the quality level. Higher quality levels tend to have higher levels of alcohol. It can be seen from the below chart that there are more of 7 and 8 scores of quality with high levels of alcohol.



- **Total sulfur dioxide → Free Sulfur Dioxide:** Fixed sulfur dioxide usually keeps decreasing over the lifetime of the wine. [1]Therefore, measuring total sulfur dioxide gives insights about fixed sulfur dioxide level.
- **Sulphates → Quality:** Inputs such as sulfur dioxide and processing enzymes contribute to the quality of wine [4]
- **Volatile acidity → Quality:** Low levels of quality associated with high levels of volatile acidity. Volatile acidity is considered a fault at higher levels and can smell sharp but at lower levels, it can add fruity-smelling raspberry, passion fruit, or cherry-like flavors.[5]
- **pH→Sulphates:** Wines with higher pH values require more sulphates to protect them from oxidation because it lowers the sulphates' effectiveness. [6]
- **Residual Sugar→Density:** Residual sugar is one of the determinants of density of wine.
- **Alcohol→Residual sugar:** Alcohol level can be seen as an indication of different levels of sugar amount in wine.[7]

During the learning phase, three different learning algorithms mentioned above were employed on the data both with knowledge and without knowledge. Plotting the learned networks, it can be seen that some of the dependencies constructed based on a priori knowledge also exist in the networks completely learned from the data. Dependencies between alcohol, sulphates and quality were also created by HC and MMHC algorithms run on the data without forcing any dependency. (Figures 1,2)

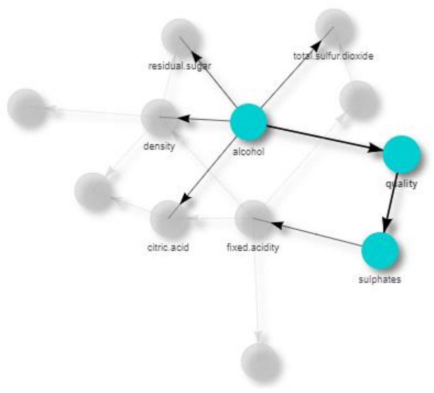


Figure 2. Network learned by Hill-Climbing algorithm

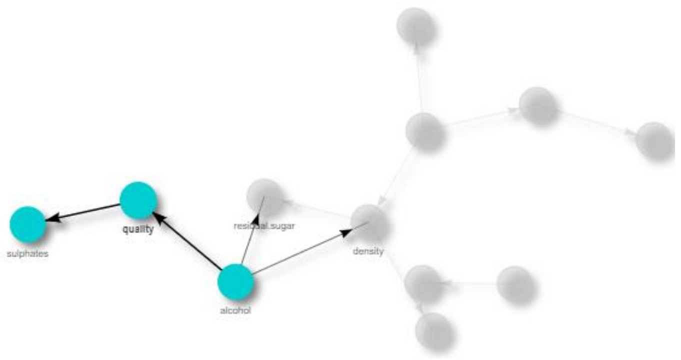


Figure 3. Network learned by MMHC algorithm

RESULTS

After the learning phase, parameters were estimated. To validate the models, firstly, the dataset was split into training and test sets in 80:20 and models were fitted in the training set. Fitted models were predicted and confusion matrices were built.

	w/o knowledge	w/ knowledge
Grow-Shrink	0.4236760	0.6199377
Hill-Climbing	0.5451713	0.6261682
MMHC	0.5451713	0.6261682

Table 1. Accuracy ratios of the models

Accuracy results of the models show that models constructed with knowledge have better performance in terms of prediction. However, we see that none of the classifiers shows high performance. Although the accuracy numbers are above 50%, further inspection of confusion matrices reveals that they do not provide the same prediction performance for all the quality levels.

Secondly, 10-fold cross validation was run for 10 times for each model and log-likelihood loss values were compared. Bn.cv function from bnlearn package is used for this process. 10 subsets were selected randomly from the data and the model was validated in each subset after fitted on the rest. This process repeated ten times, and the losses were averaged over the folds. Below, models and associated log-likelihood losses are depicted. There is not much difference between the models learned from data and the models learned with prior knowledge, since the models with knowledge give higher prediction accuracy, selection was made by considering only those models.

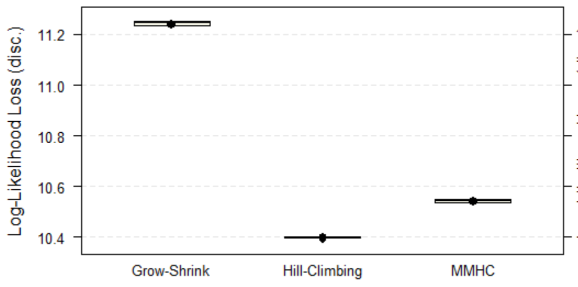


Figure 4. Log-Likelihood Loss of Models (Structure learned from the data)

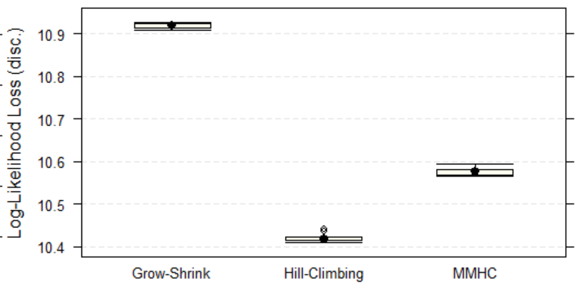


Figure 5. Log-Likelihood Loss of Models (Structure learned with apriori knowledge)

	w/o knowledge	w/ knowledge
Grow-Shrink	11.24041	10.92354
Hill-Climbing	10.39611	10.41737
MMHC	10.53960	10.57186

Table 2. Log-Likelihood Losses

Finally, Hill-Climbing with prior knowledge gives the highest accuracy and the lowest loss hence, was selected as the final model. In order to have a robust model, model averaging was applied to the selected algorithm and run with two different scores: AIC and BIC. Resulting accuracy rates are 0.5919 and 0.5950 respectively. There is not any significant difference between the two models. In this analysis, Hill-Climbing with AIC was selected as the final model.

	F1	Sensitivity	Presicion	Recall	Balanced_acc
Class: 3	NA	0.0000000	NA	0.0000000	0.5000000
Class: 4	NA	0.0000000	NA	0.0000000	0.5000000
Class: 5	0.6912752	0.7573529	0.6358025	0.7573529	0.7192170
Class: 6	0.5440000	0.5312500	0.5573770	0.5312500	0.6257286
Class: 7	0.5000000	0.4750000	0.5277778	0.4750000	0.7072509
Class: 8	NaN	0.0000000	0.0000000	0.0000000	0.4984227

Table 3. Prediction Performance Statistics of the Final Model

It is obvious that final model performs better at predicting moderate levels of quality. Recall ratio is above 50% for the levels 5 and 6. However, for lower and higher quality levels, model fails to predict accurately. This is not surprising considering the high number of class 5 and class 6 observations in the sample. Lower and higher quality levels are less in number and dominated by the moderate quality levels. This can mislead the classifier and cause misclassifications.

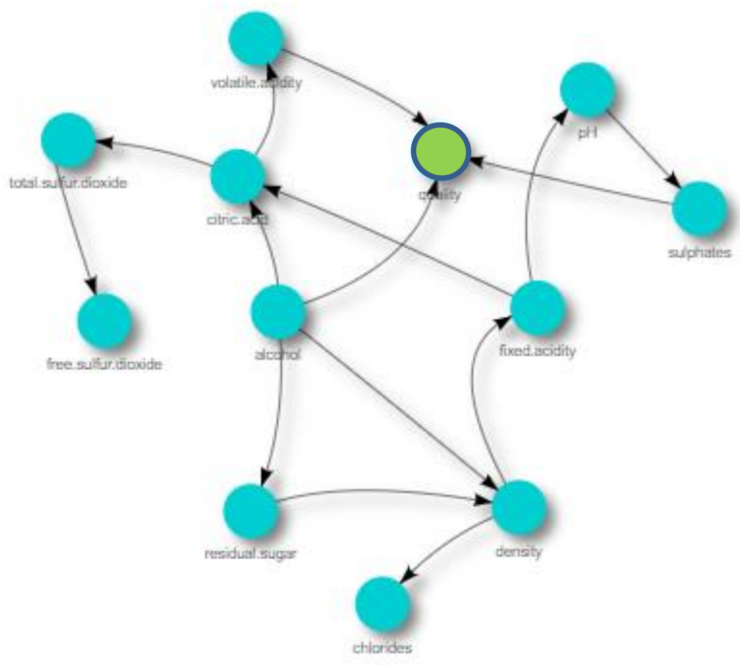


Figure 6. Bayesian Network Structure of the Final Model
(Hill-Climbing with AIC score and prior knowledge)

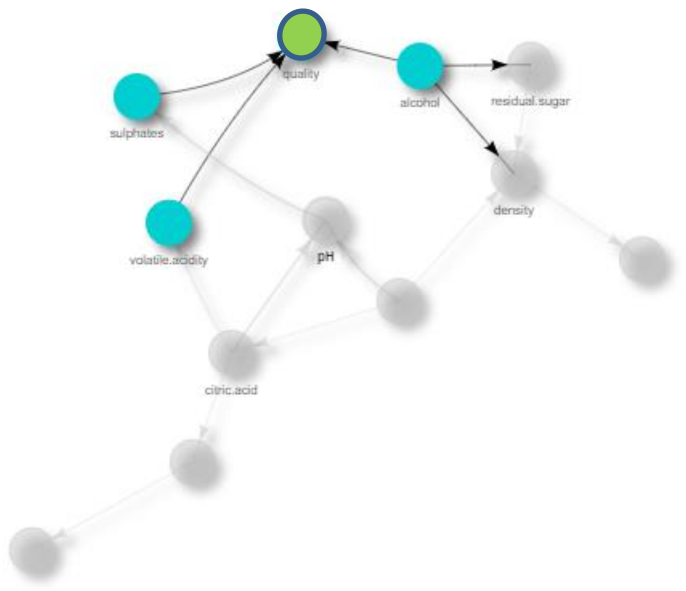


Figure 7. Dependencies of the Target Variable Highlighted

Variables that quality has dependencies with are alcohol, sulphates and volatile acidity. These dependencies were forced while learning the structure and none of the rest of the variables was found relative by the algorithm. Conditional independencies of the target variable are as follows:

$$\text{Quality} \perp\!\!\!\perp \text{pH} \mid \text{Sulphates}$$

$$\text{Quality} \perp\!\!\!\perp (\text{Residual Sugar, Citric Acid, Density}) \mid \text{Alcohol}$$

$$\text{Quality} \perp\!\!\!\perp \text{Citric Acid} \mid \text{Volatile Acidity}$$

As for the marginal independencies, quality is marginally independent on fixed acidity, chlorides, total sulfur dioxide and free sulfur dioxide.

Inference

Marginal probabilities of the quality levels are as follows:

Quality levels

3	**4**	**5**	**6**	**7**	**8**
%0.6	3.4%	4.2%	4.2%	10%	0.9%

Quality levels 7 and 8 are considered as high, while 3 and 4 are considered as low levels of quality.

Quality levels given alcohol

Conditional probability of having a higher level quality increases with higher levels of alcohol.When compared to marginal probabilities, it can be seen that when low levels of alcohol is observed, probability of getting higher quality scores decreases from 10% to 4% for score 7.

Probability of getting quality score 7 increases from 10% to 34% , getting score 8 increases from 0% to 6% when high levels of alcohol have been observed.

Quality levels given sulphates

Sulphates is one of the variables that affects the target variable, examination of conditional probabilities of quality shows that probability of having quality score 7 given high levels of sulphates 12% for level 7, 2% for level 8. These probabilities are higher than the marginal probabilities of quality scores 7 and 8. Conditional probabilities of having higher levels of quality given low levels of sulphates are 6% and 0% for levels 7 and 8 respectively.

Quality levels given volatile acidity

Another variable that affects the quality level of red wine is volatile acidity. Low levels of volatile acidity cause an increase in the probability of having a high quality wine. While marginal probability of having a red wine with quality score 7 is %10, getting the same quality when low level of volatile acidity has been observed is 19%, when high level of volatile acidity has been observed it drops to 7%.

Residual sugar given alcohol

Marginal probabilities of residual sugar for low, average and high levels are 21%, 46% and 32%, respectively. Given the high levels of alcohol, conditional probability of high residual sugar is 39%. Increase in the probability of high levels of residual sugar from 32% to 39% proves that high levels of alcohol could be an indicator of the high sugar level in the wine. Similarly, given the low levels of alcohol, conditional probability of high residual sugar drops to 30%.

Density given alcohol

Marginal probabilities of density for low, average and high levels are 19%, 57% and 23%, respectively. Conditional probability of high level of density given low levels of alcohol is 31%. Increase in the conditional probability of low levels of density given high levels of alcohol is more dramatic. Probability of seeing low levels of density jumps from 19% to 72% density given high levels of alcohol.

CONCLUSION

In summary, Hill-Climbing with AIC score was selected as the best model for the red wine dataset. After learning the structure based on prior knowledge, Bayesian network was constructed and parameters were estimated. Variables affecting quality which are alcohol, volatile acidity and sulphates are among the variables found important in the article[8] which originally introduced the dataset used in this analysis. Although alcohol and sulphates were given to the algorithm as a priori knowledge, algorithms run on the data without knowledge also found dependencies between these variables and the target variable. In general, not seeing any dependency between the other variables and quality, does not mean that other variables are not important. There should be balance in the amounts of ingridients in order to have a good wine.

Final model does not have a strong classification performance and biased towards the majority, meaning that it has relatively better performance on predicting quality levels 5 and 6 but fails to predict levels 3,4,7 and 8. This is due to the fact that the data is imbalanced. Model can be improved with more observations of higher/lower levels of quality and/or establishing the network based purely on expert knowledge.

REFERENCES

1. https://www.extension.iastate.edu/wine/total-sulfur-dioxide-why-it-matters-too
2. https://en.wikipedia.org/wiki/Acids_in_wine#:~:text=The%20acids%20in%20wine%20are,the%20finished%20product%20of%20wine.&text=Generally%2C%20the%20lower%20the%20pH,for%20wine%20and%20high%20acidity)
3. Bayesian Network Approach for Modelling and Inference of Communication Networks, Master's thesis in Engineering Mathematics and Computational Science Mouliakos, Themis
4. https://www.jjbuckley.com/wine-knowledge/blog/the-4-factors-and-4-indicators-of-wine-quality/1009
5. https://winefolly.com/deep-dive/weird-wine-flavors-and-the-science-behind-them/
6. https://www.winemag.com/2019/06/19/what-is-acidity-in-wine/
7. https://www.wineinvestment.com/wine-blog/2019/04/what-ingredients-are-really-in-your-glass-of-wine/
8. 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. ISSN: 0167-9236.
9. https://archive.ics.uci.edu/ml/datasets/Wine+Quality

APPENDIX: R CODE

```
library(gRbase)
library(gRim)
library(bnlearn)
library(caTools)
library(gRain)
library(caret)
library(visNetwork)
library(formattable)

#import the dataset
wine=read.csv("C:/Users/TOSHIBA/Desktop/PM_Project/wine_quality/winequality-red.csv", header
= TRUE,sep = ';')
attach(wine)
str(wine)
summary(wine)

#Discretize with Hartemink's Method (except "quality")
dwine = discretize(wine[1:11], method = "hartemink",
                breaks = 3,ibreaks = 20, idisc = "quantile") #

for (i in names(dwine))
  levels(dwine[, i]) = c("LOW", "AVG", "HIGH")
str(dwine)

#convert quality levels into factors
dwine$quality<-as.factor(wine$quality)
summary(dwine)

###Descriptives#####

#quality levels bar chart
ggplot(dwine, aes(x = quality)) +
  geom_bar() +
  xlab("quality") +
  theme(axis.text.x = element_text(hjust = 1))
# contingency charts


#quality levels vs alcohol
### quality tends to increase with alcohol
ggplot(dwine, aes(x = quality)) +
  geom_bar(aes(y = (..count..)/sum(..count..))) +
  xlab("quality") +
  ggtitle("Quality by Alcohol Levels")+
  scale_y_continuous(labels = scales::percent, name = "Proportion")+
  facet_grid(~ alcohol) +
  theme(axis.text.x = element_text(hjust = 1),plot.title = element_text(hjust = 0.5))


########## Plotting function#######

plot.network <- function(structure, ht = "700px"){
  nodes.uniq <- unique(c(structure$arcs[,1], structure$arcs[,2]))
  nodes <- data.frame(id = nodes.uniq,
                      label = nodes.uniq,
                      color = "darkturquoise",
                      shadow = TRUE)
  edges <- data.frame(from = structure$arcs[,1],
                      to = structure$arcs[,2],
                      arrows = "to",
                      smooth = TRUE,
                      shadow = TRUE,
                      color = "black")
  return(visEdges(visNetwork(nodes, edges, height = ht, width = "100%"),physics = FALSE, smooth =
FALSE)%>%
    visOptions(highlightNearest = TRUE, nodesIdSelection = TRUE) %>%
    visLayout(randomSeed = 123))
}

#LEARNING THE STRUCTURE ###

#creating whitelist and blacklist for learning with prior knowledge:
whitelist=matrix(c(
  "alcohol","quality",
  "volatile.acidity","quality",
  "residual.sugar","density",
  "alcohol","residual.sugar",
  "sulphates","quality",
  "pH","sulphates",
  "total.sulfur.dioxide","free.sulfur.dioxide"),,2,byrow = TRUE)
colnames(whitelist)=c("from","to")

#Grow-Shrink
learn_gs<-function(data,whitelist=NULL,blacklist=NULL){
  bn <- gs(data, test="mi",whitelist = whitelist,blacklist = blacklist)
  return(bn)}
```

```r
#Hill-Climbing
learn_hc<-function(data,whitelist=NULL,blacklist=NULL){
  bn <- hc(data,whitelist = whitelist,blacklist = blacklist)
  return(bn)}

###MMHC
#MMHC
#(mmhc() is simply rsmax2() with restrict set to mmpc and maximize set to hc.)

learn_mmhc<-function(data,whitelist=NULL,blacklist=NULL){
  bn <- rsmax2(data, restrict = "mmpc",maximize = "hc",whitelist = whitelist,blacklist = blacklist)
  return(bn)}

##structures learned from the data:
gs<-learn_gs(dwine)
hc<-learn_hc(dwine)
mmhc<-learn_mmhc(dwine)

##structures learned with knowledge:
gs_knowledge<-learn_gs(dwine,whitelist = whitelist)
hc_knowledge<-learn_hc(dwine,whitelist = whitelist)
mmhc_knowledge<-learn_mmhc(dwine,whitelist = whitelist)

## structure visualizations ##
plot.network(gs)
plot.network(gs_knowledge)

plot.network(hc)
plot.network(hc_knowledge)

plot.network(mmhc)
plot.network(mmhc_knowledge)

######## PARAMETER ESTIMATION #################

#splitting the data set
set.seed(123)
split= sample.split(dwine$quality, SplitRatio = 4/5)

train_set= subset(dwine, split== TRUE)
test_set= subset(dwine, split== FALSE)

#model fit, and confusion matrices
#with validation
fit_validate<-function(bn){
  model=bn.fit(bn,data=train_set,method='bayes')
  m_preds=bnlearn:::predict.bn.fit(model,"quality",test_set)
  cf=confusionMatrix(m_preds,test_set$quality, positive = "Yes")
  cf
}

#models estimated from structures learned from the data
#Grow-Shrink
#bn.gs is an undirected graph and must be extended into a DAG with cextend()
gs_ = cextend(gs)
fit_validate(gs_)
#Hill-Climbing
fit_validate(hc)
#MMHC
fit_validate(mmhc)

#models estimated from structures learned based on prior knowledge
#Grow-Shrink
#bn.gs is an undirected graph and must be extended into a DAG with cextend()
gs_k = cextend(gs_knowledge)
fit_validate(gs_k)
#Hill-Climbing
fit_validate(hc_knowledge)
#MMHC
fit_validate(mmhc_knowledge)

## Confusion Matrix Statistics ##
acc_scores1=c(fit_validate(gs_)$overall['Accuracy'],fit_validate(hc)$overall['Accuracy'],fit_validate(m
mhc)$overall['Accuracy'])
acc_scores2=c(fit_validate(gs_k)$overall['Accuracy'],fit_validate(hc_knowledge)$overall['Accuracy'],fi
t_validate(mmhc_knowledge)$overall['Accuracy'])
accuracy_scores=data.frame(acc_scores1,acc_scores2)
colnames(accuracy_scores)=c('w/o knowledge',"w/ knowledge")
rownames(accuracy_scores)=c('Grow-Shrink','Hill-Climbing','MMHC')
accuracy_scores
formattable(accuracy_scores,
        list(`Indicator Name` = formatter("span", style = ~ style(color = "grey",font.weight = "bold"))))


#with cross-validation
fit_cv<-function(data,bn,runs){
  model=bn.cv(data=data, bn = bn ,fit = "bayes", runs=runs)
  }
#models estimated from structures learned from the data
#Grow-Shrink
```

```
gs_cv=fit_cv(dwine,gs,10)
#Hill-Climbing
hc_cv=fit_cv(dwine,hc,10)
###the MMHC
mmhc_cv=fit_cv(dwine,mmhc,10)

#models estimated from structures learned based on prior knowledge
#Grow-Shrink
#first adding directions to arcs
gs_cv_k=fit_cv(dwine,gs_knowledge,10)
#Hill-Climbing
hc_cv_k=fit_cv(dwine,hc_knowledge,10)
###MMHC
mmhc_cv_k=fit_cv(dwine,mmhc_knowledge,10)

#plotting losses
plot(gs_cv,hc_cv,mmhc_cv,xlab=c('Grow-Shrink','Hill-Climbing','MMHC'))
plot(gs_cv_k,hc_cv_k,mmhc_cv_k,xlab=c('Grow-Shrink','Hill-Climbing','MMHC'))

losses1=c(mean(loss(gs_cv)),mean(loss(hc_cv)),mean(loss(mmhc_cv)))
losses2=c(mean(loss(gs_cv_k)),mean(loss(hc_cv_k)),mean(loss(mmhc_cv_k)))
losses=data.frame(losses1,losses2)
colnames(losses)=c('w/o knowledge',"w/ knowledge")
rownames(losses)=c('Grow-Shrink','Hill-Climbing','MMHC')
losses

formattable(losses,
        list('Indicator Name' = formatter("span", style = ~ style(color = "grey",font.weight = "bold"))))

## MODEL AVERAGING FOR FINAL MODEL ##

#hc with bic

models_bic <- boot.strength(data = train_set,R=1000,algorithm = 'hc',algorithm.args =
list(score='bic',whitelist=whitelist))
plot(models_bic)# inspecting the threshold
best_model_bic <- averaged.network(models_bic,threshold = 0.85)
fitted_model_bic <- bn.fit(best_model_bic,data = train_set,method = 'bayes')
model_y_preds=bnlearn:::predict.bn.fit(fitted_model_bic,"quality",test_set)
Learned_Bayesian_network <- plot.network(best_model_bic)
conf_bic=confusionMatrix(model_y_preds,test_set$quality)
conf_bic
```

```
#hc with aic

models_aic <- boot.strength(data = train_set,R=1000,algorithm = 'hc',algorithm.args =
list(score='aic',whitelist=whitelist))
best_model_aic <- averaged.network(models_aic,threshold = 0.85)
fitted_model_aic <- bn.fit(best_model_aic,data = train_set,method = 'bayes')
avg_model_y_preds=bnlearn:::predict.bn.fit(fitted_model_bic,"quality",test_set)
Learned_Bayesian_network <- plot.network(best_model_aic)
conf_aic=confusionMatrix(avg_model_y_preds,test_set$quality)
conf_aic

accuracy=c(conf_bic$overall['Accuracy'],conf_aic$overall['Accuracy'])
losses2=c(mean(loss(gs_cv_k)),mean(loss(hc_cv_k)),mean(loss(mmhc_cv_k)))
accuracy=data.frame(accuracy)
colnames(accuracy)=c('Accuracy')
rownames(accuracy)=c('Hc_BIC','Hc_AIC')
formattable(accuracy)

# Confusion Matrix statistics of the final model
F1=conf_aic[["byClass"]][ , "F1"]
Sensitivity=conf_aic[["byClass"]][ , "Sensitivity"]
Presicion=conf_aic[["byClass"]][ , "Precision"]
Recall=conf_aic[["byClass"]][ , "Recall"]
Balanced_acc=conf_aic[["byClass"]][ , "Balanced Accuracy"]
final_stats=data.frame(F1,Sensitivity,Presicion,Recall,Balanced_acc)
formattable(final_stats)


#### INFERENCE ####

fitted = bn.fit(best_model_aic, dwine, method = "bayes")# fitting the model to the whole dataset
fitted_grain=as.grain(fitted ) #jtree

#quality_marginal probabilities
querygrain(fitted_grain,nodes ="quality",type='marginal')

#set the evidence
ev1 = setFinding(fitted_grain, nodes = "alcohol", states = "HIGH")
ev2 = setFinding(fitted_grain, nodes = "alcohol", states = "LOW")
#Conditional probabilities of quality levels given alcohol
querygrain(ev1,nodes = "quality",type = 'conditional')

#Conditional probabilities of quality levels given sulphates
ev3 = setFinding(fitted_grain, nodes = "sulphates", states = "LOW")
ev4 = setFinding(fitted_grain, nodes = "sulphates", states = "HIGH")
querygrain(ev4,nodes = "quality",type = 'conditional')
```

```
#Conditional probabilities of the quality levels given volatile acidity
ev5 = setFinding(fitted_grain, nodes = "volatile.acidity", states = "LOW")
ev6 = setFinding(fitted_grain, nodes = "volatile.acidity", states = "HIGH")
querygrain(ev6,nodes = "quality",type = 'conditional')

#Conditional probabilities of residual sugar given alcohol
querygrain(fitted_grain,nodes ="residual.sugar",type='marginal')
querygrain(ev1,nodes = "residual.sugar",type = 'conditional')

#Conditional probabilities of density given alcohol
querygrain(fitted_grain,nodes ="density",type='marginal')
querygrain(ev2,nodes = "density",type = 'conditional')
```