Machine Learning
Project

Vaduva Alexandra - Mihaela
CEN 4.2SB

Dataset: Wine quality - [UCI Machine Learning Repository: Wine Quality Data Set]

Load libraries

```
library(ggplot2)
library(mlbench)
library(caret)
library(lattice)
library(e1071)
library(corrplot)
library(correlation)
library(randomForest)
library(rpart)
library(rpart.plot)
```

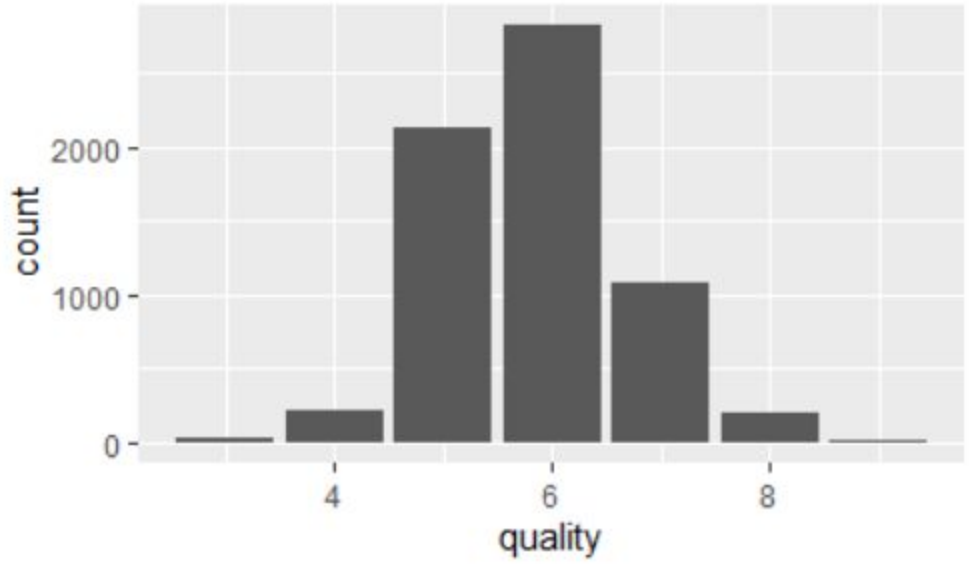
Load dataset

```
dataset <- wines
```

Summarize data

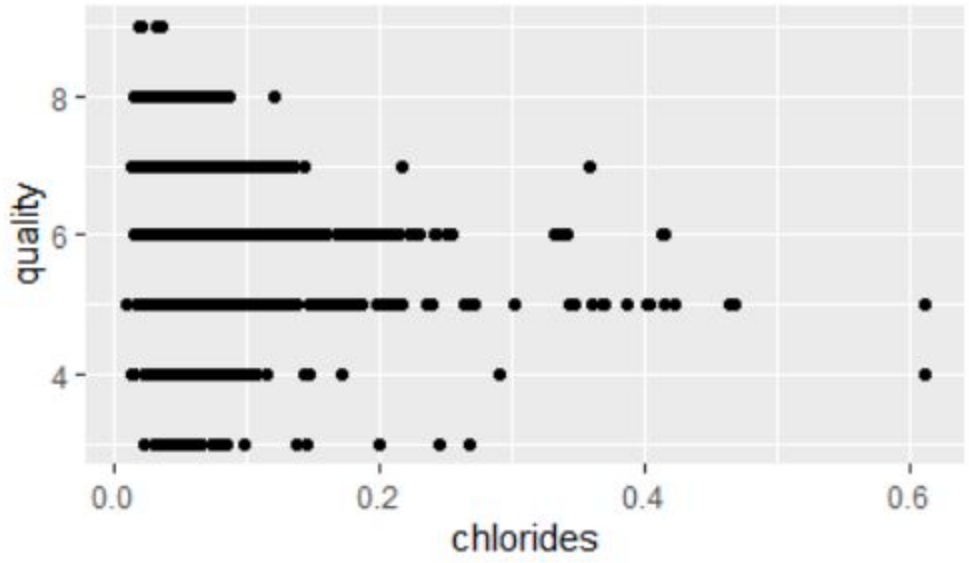
```
str(dataset)
summary(dataset)

#check if missing values exist in dataset
is.na(dataset)
anyNA(dataset)

#omit NA values
na.omit(dataset)
dataset <- na.omit(dataset)

#dimension of dataset
dim(dataset)                - [1] 6468   14
```

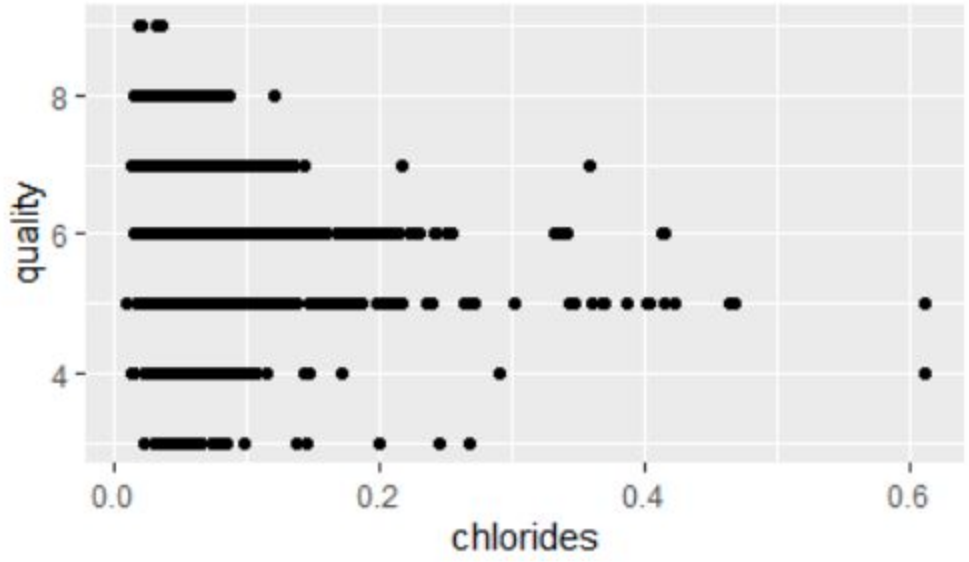
```
#list types for each attribute
sapply(dataset,class)

head(dataset, n=20)

   type        fixed.acidity     volatile.acidity
        "character"           "numeric"           "numeric"
        citric.acid       residual.sugar            chlorides
          "numeric"           "numeric"           "numeric"
 free.sulfur.dioxide total.sulfur.dioxide             density
          "numeric"           "numeric"           "numeric"
                 pH           sulphates             alcohol
          "numeric"           "numeric"           "numeric"
            quality               taste
          "numeric"            "factor"

#class distribution
y <- dataset$quality
cbind(freq=table(y),percentage=prop.table(table(y))*100)


freq  percentage
3   30  0.46382189
4  214  3.30859617
5 2128 32.90043290
6 2824 43.66110080
7 1075 16.62028448
8  192  2.96846011
9    5  0.07730365


#Distribution of quality
ggplot(data=dataset) + geom_bar(mapping = aes(x=quality))
```




```
#Univariate plots
names(dataset) <-make.names(names(dataset))
ggplot(data = dataset, aes(x =fixed.acidity,y=quality)) + geom_point()
ggplot(data = dataset, aes(x = volatile.acidity, y = quality)) +
  geom_point()
ggplot(data = dataset, aes(x = citric.acid, y = quality)) +
  geom_point()
ggplot(data = dataset, aes(x = residual.sugar, y = quality)) +
  geom_point()
ggplot(data = dataset, aes(x = chlorides, y = quality)) +
  geom_point()
ggplot(data = dataset, aes(x = free.sulfur.dioxide, y = quality)) +
  geom_point()
ggplot(data = dataset, aes(x = total.sulfur.dioxide, y = quality)) +
  geom_point()
ggplot(data = dataset, aes(x = density, y = quality)) +
  geom_point()
ggplot(data = dataset, aes(x = pH, y = quality)) +
  geom_point()
ggplot(data = dataset, aes(x = sulphates, y = quality)) +
```

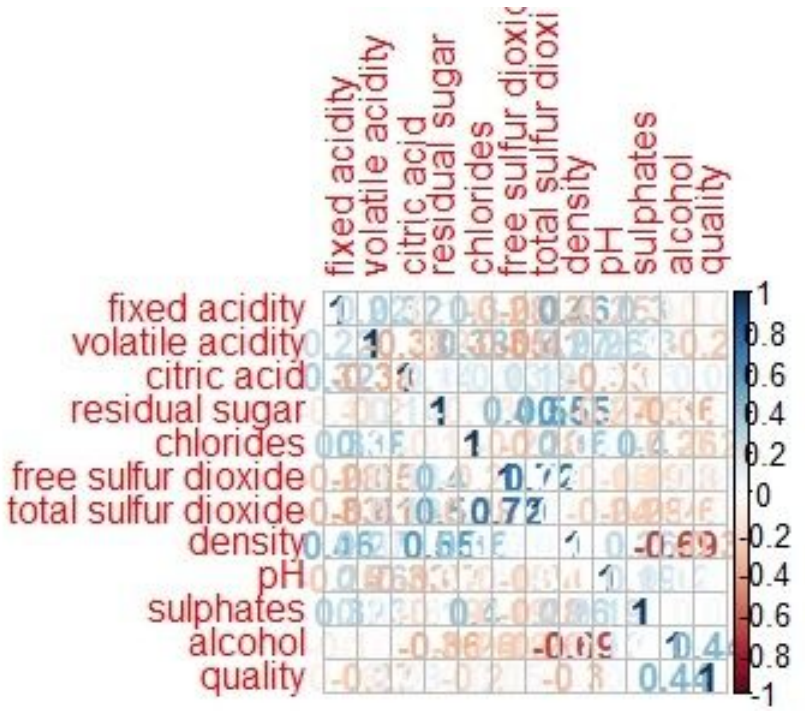
```
  geom_point()
ggplot(data = dataset, aes(x = alcohol, y = quality)) +
  geom_point()
```



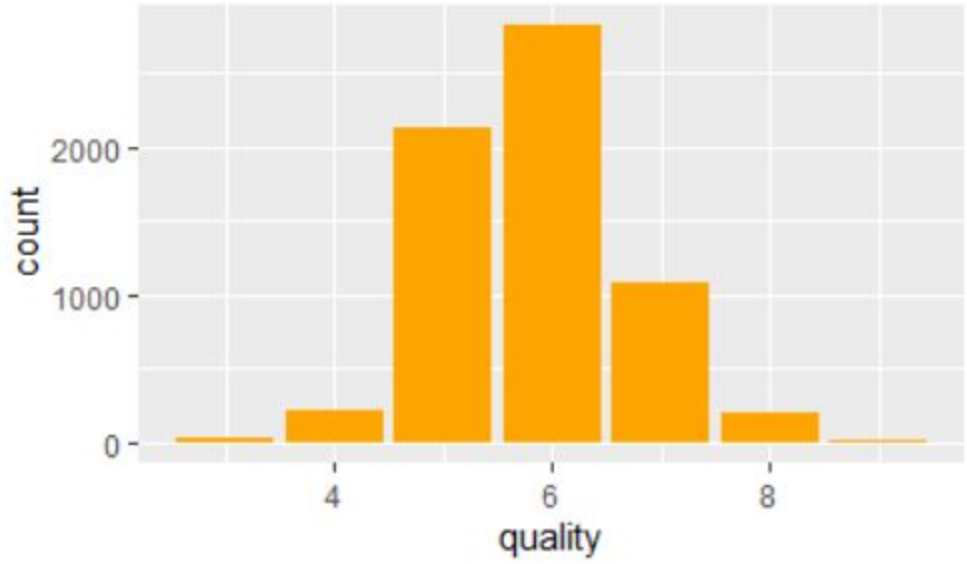



```
# correlation plot
data1 <- dataset[, c( "fixed acidity",
                      "volatile acidity",
                      "citric acid",
```

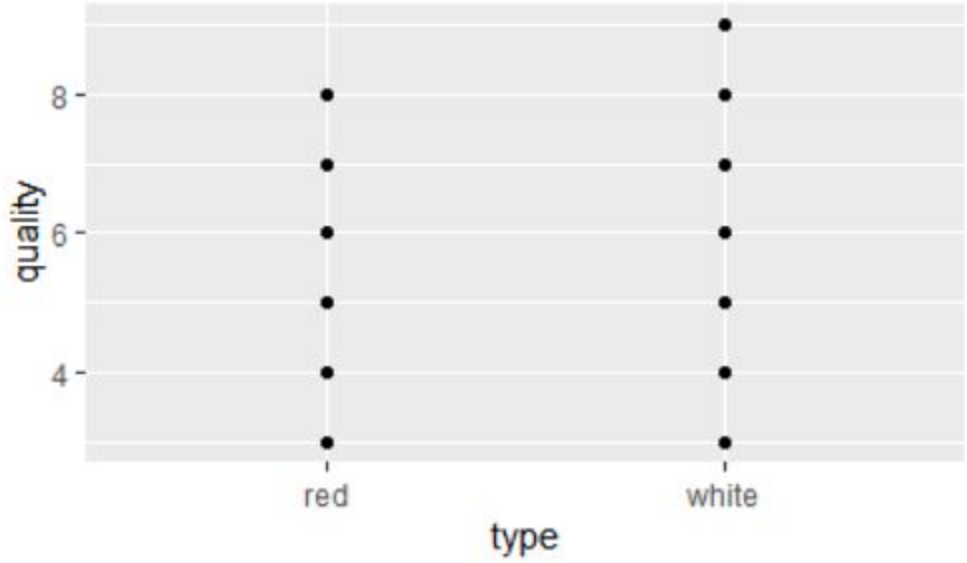
```
                "residual sugar",
                "chlorides",
                "free sulfur dioxide",
                "total sulfur dioxide","density","pH","sulphates",
                "alcohol","quality") ]
data1 <- na.omit(data1)
cor(data1)
corrplot(cor(data1), method="number")
```




```
#bar plot
par(mfrow=c(2,4))
for(i in 2:9) {
  counts <- table(dataset[,i])
  name <- names(dataset)[i]
  barplot(counts, main=name)
}

ggplot(dataset, aes(quality)) + geom_bar(fill="orange")
```

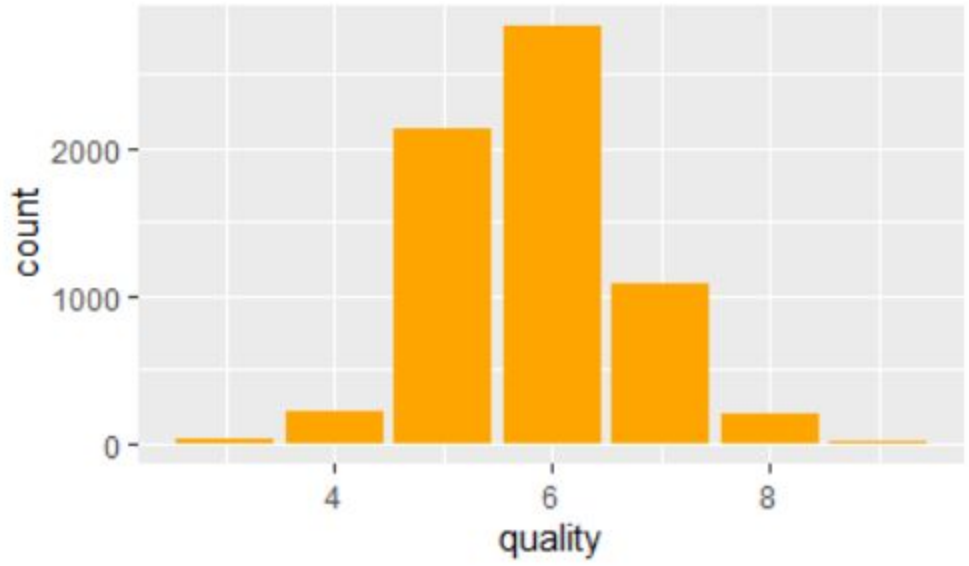



```
#point plot
ggplot(dataset , aes(x = type, y =quality)) + geom_point()
```




```
# bar plot switch the x and y axes
ggplot(dataset, aes(x = type, y = quality)) +
  geom_col() +
  coord_flip()

ggplot(dataset, aes(quality)) +
```

```
geom_bar(fill="orange")
```



Prepare data

```
#classification of the wines into good, bad and normal
dataset$taste <-ifelse(dataset$quality < 6, 'bad','good')
dataset$taste[dataset$quality == 6] <- 'normal'
dataset$taste <- as.factor(dataset$taste)

table(dataset$taste)
```

Split-out dataset

```
#splitting the dataset
set.seed(7)
index <- sample(nrow(dataset), 0.8*nrow(dataset))
```

```
train <- dataset[index, ]
test <- dataset[-index, ]

#omit NA values for train and test
na.omit(train)
na.omit(test)
train <- na.omit(train)

#normalize column names
names(train) <- make.names(names(train))
names(test) <- make.names(names(test))
```

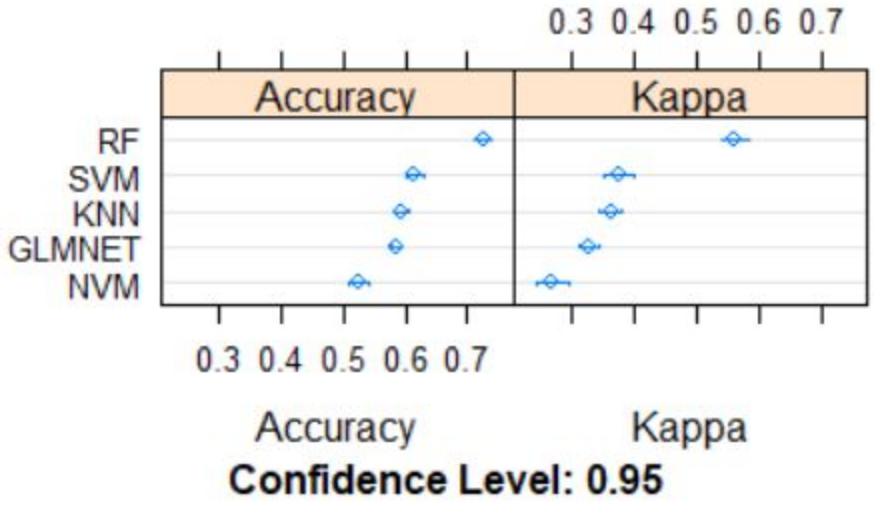
Evaluate algorithms and compare

```
#Run algorithms using 10-fold cross validation
control <- trainControl(method="repeatedcv", number=10)
metric <- "Accuracy"

# a) linear algorithms
#GLMNET
set.seed(7)
fit.glmnet <- train(taste~.-quality, data=train, method="glmnet", metric=metric,
trControl=control)
# SVM
set.seed(7)
fit.svm <- train(taste~.-quality, data=train, method="svmRadial", metric=metric,
trControl=control)
# b) nonlinear algorithms
#knn
set.seed(7)
fit.knn <- train(taste~.-quality, data=train, method="knn", metric=metric,
trControl=control)
#randomForest
set.seed(7)
fit.rf <- train(taste~.-quality, data=train, method="rf", metric=metric,
trControl=control)
```

```
#Naive Bayes
set.seed(7)
fit.nb <- train(taste~.-quality,data=train,method="nb",metric=metric,
trControl=control)

#Compare algorithms
transform_results <- resamples(list(GLMNET=fit.glmnet, SVM=fit.svm,
KNN=fit.knn,NVM=fit.nb, RF =fit.rf))
summary(transform_results)
dotplot(transform_results)

#Evaluate Algorithms : with Feature Selection step

#remove correlated attributes
#find attributes that are highly corrected
set.seed(7)
cutoff <- 0.70
correlations <- cor(train[,2:12])
highlyCorrelated <- findCorrelation(correlations, cutoff=cutoff)
for (value in highlyCorrelated) {
  print(names(train)[value])
}
#create a new dataset without highly corrected features
train_features <- train[,-highlyCorrelated]

#Run algorithms using 10-fold cross validation
control <- trainControl(method="repeatedcv", number=10)
metric <- "Accuracy"

# a) linear algorithms
#GLMNET
set.seed(7)
fit.glmnet <- train(taste~.-quality, data=train_features, method="glmnet",
metric=metric, trControl=control)
# SVM
set.seed(7)
```

```
fit.svm <- train(taste~.-quality, data=train_features, method="svmRadial",
metric=metric, trControl=control)
# b) nonlinear algorithms
#knn
set.seed(7)
fit.knn <- train(taste~.-quality, data=train_features, method="knn", metric=metric,
trControl=control)
#randomForest
set.seed(7)
fit.rf <- train(taste~.-quality, data=train_features, method="rf", metric=metric,
trControl=control)
#Naive Bayes
set.seed(7)
fit.nb <- train(taste~.-quality,data=train_features,method="nb",metric=metric,
trControl=control)

#Compare algorithms
transform_results <- resamples(list(GLMNET=fit.glmnet, SVM=fit.svm,
KNN=fit.knn,NVM=fit.nb, RF =fit.rf))
summary(transform_results)
dotplot(transform_results)

#Evaluate algorithms : with Box-Cox Transformation

#Run algorithms using 10-fold cross validation
control <- trainControl(method="repeatedcv", number=10)
metric <- "Accuracy"


# a) linear algorithms
#GLMNET
set.seed(7)
fit.glmnet <- train(taste~.-quality, data=train, method="glmnet",
metric=metric,preProc=c("center", "scale", "BoxCox"), trControl=control)
# SVM
set.seed(7)
```

```
fit.svm <- train(taste~.-quality, data=train, method="svmRadial",
metric=metric,preProc=c("center", "scale", "BoxCox"), trControl=control)
# b) nonlinear algorithms
#knn
set.seed(7)
fit.knn <- train(taste~.-quality, data=train, method="knn",
metric=metric,preProc=c("center", "scale", "BoxCox"), trControl=control)
#randomForest
set.seed(7)
fit.rf <- train(taste~.-quality, data=train, method="rf",
metric=metric,preProc=c("center", "scale", "BoxCox"), trControl=control)
#Naive Bayes
set.seed(7)
fit.nb <-
train(taste~.-quality,data=train,method="nb",metric=metric,preProc=c("center",
"scale", "BoxCox"), trControl=control)

#Compare algorithms
transform_results <- resamples(list(GLMNET=fit.glmnet, SVM=fit.svm,
KNN=fit.knn,NVM=fit.nb, RF =fit.rf))
summary(transform_results)
dotplot(transform_results)
```



Finalize Model

```
print(fit.rf)


x <- test[,1:13]
y <- test[,14]

predictions <- predict(fit.rf, newdata=x)
print(predictions)


#calculate Accuracy
table(predictions, test$taste)
(373+163+431) / nrow(test)

#save the model to disk
saveRDS(fit.rf,"MyFinalModel.rds")

#use the model for prediction
print("load the model")
model <- readRDS("MyFinalModel.rds")

finalPredictions <- predict(model, x)
print(finalPredictions)
table(finalPredictions,test$taste)

finalPredictions bad good normal
        bad    374    7     89
        good     6  136     39
        normal 127   94    422

(519+233+652)/ nrow(test) - values of predicted values divided to the number of rows
```

```
#confusionMatrix
confusionMatrix(finalPredictions, test$taste)

Confusion Matrix and Statistics

          Reference
Prediction bad good normal
   bad    374    7     89
   good     6  136     39
   normal 127   94    422

Overall Statistics

               Accuracy : 0.7202
                 95% CI : (0.6949, 0.7446)
    No Information Rate : 0.425
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5494

 Mcnemar's Test P-Value : 1.753e-06

Statistics by Class:

                     Class: bad Class: good Class: normal
Sensitivity              0.7377      0.5738        0.7673
Specificity              0.8780      0.9574        0.7030
Pos Pred Value           0.7957      0.7514        0.6563
Neg Pred Value           0.8386      0.9093        0.8034
Prevalence               0.3918      0.1832        0.4250
Detection Rate           0.2890      0.1051        0.3261
Detection Prevalence     0.3632      0.1399        0.4969
Balanced Accuracy        0.8078      0.7656        0.7351
```