```
# Execute the lines below to prepare data
library(class)
library(MASS)
library(rpart)
library(randomForest)
wine <- read.csv("winequality-red.csv")
wine$quality <- as.factor(wine$quality)
set.seed(2023)
train <- sample(1:nrow(wine), 0.7*nrow(wine))
valid <- setdiff(1:nrow(wine), train)
input_vars <- colnames(wine)[1:11]

# Scratch area: Put your R code for exploratory analysis below. If you don't know what to
do on your own, follow the tutorial PDF on Canvas and try some commands there. The scratch
code does not need to be well organized or annotated.
typeof(train)
head(train)

typeof(wine)

length(train)
length(valid)

unique(wine$quality)
class(wine$quality)
table(wine$quality)

dim(wine)
colnames(wine)

wine_corr <- wine
wine_corr$quality <- as.numeric(wine_corr$quality)
wine_corr1 <- wine_corr[, c(1,2,3,4,5,6,7,8,9,10,11)]
cor(wine_corr1)



hist(wine$density)
barplot(table(wine$quality), xlab = 'quality', ylab = 'obs.')
with(wine, boxplot(fixed.acidity ~ quality))


(formula.string <- paste0(colnames(wine), " ~ quality"))

pdf("wine.numeric_vs_quality.pdf")
for(i in 1:(length(formula.string)-1)){
  with(wine, boxplot(as.formula(formula.string[i])))
}
dev.off()

input_vars

quality_distribution <- prop.table(table(wine[train, 'quality']))

wine_formula <- formula(wine$quality ~ volatile.acidity + chlorides)
rf_wine <- randomForest(wine_formula, ntree=100, data = wine, subset = train, method =
'class')



decision_tree_wine <- rpart(wine_formula, data = wine, subset = train)
```

```
# Write a function to predict wine quality using any subset of the input variables
# The function should return the predicted quality value for each row in newdata
# Do not alter the function declaration, i.e., function name and the argument list
myPredictFunc <- function(model = NULL, newdata = wine[valid, input_vars]){
  ## write your function body here
  ## model can be any object, but it should not encode data from the valid set
  ## Do not reference any information beyond what is packed in the model object
  pred <- c(0)
  qd <- model$qd
  for(i in 1:nrow(newdata)){
    pred[i] <- sample(names(quality_distribution), size = 1, prob = quality_distribution)
  }


  return(pred)  # placeholder, replace with something meaningful
}

myPredictFunc_randomForest <- function(model = NULL, newdata = wine[valid, input_vars]){
  ## write your function body here
  ## model can be any object, but it should not encode data from the valid set
  ## Do not reference any information beyond what is packed in the model object
  pred_rf <- predict(model$rf_wine, newdata = newdata)


  pred <- rf_wine$predicted


  return(pred)  # placeholder, replace with something meaningful
}

myPredictFunc_decisionTree <- function(model = NULL, newdata = wine[valid, input_vars]){
  ## write your function body here
  ## model can be any object, but it should not encode data from the valid set
  ## Do not reference any information beyond what is packed in the model object
  pred_dt <- predict(model$dt, newdata, type = 'class')
  pred <- pred_dt


  return(pred)  # placeholder, replace with something meaningful
}

# Pack your model in an R object called myModel
myModel <- list(qd = quality_distribution, rf_wine = rf_wine, dt = decision_tree_wine)  #
This is a placeholder, change it

# Test the prediction function on the valid set
pred <- myPredictFunc(myModel)
pred_rf <- myPredictFunc_randomForest(myModel)
pred_dt <- myPredictFunc_decisionTree(myModel)

# Report the accuracy
(accuracy <- sum(pred == wine[valid,'quality'])/length(valid))
(accuracy_rf <- sum(pred_rf == wine[valid,'quality'])/length(valid))
(accuracy_dt <- sum(pred_dt == wine[valid,'quality'])/length(valid))

print(accuracy)
print(accuracy_rf)
print(accuracy_dt)
```