CS 699 Assignment 3

Katherine Rein

```
# Import libraries
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.3.3
```

```
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 4.3.3
```

```
## Registered S3 method overwritten by 'future':
##   method               from
##   all.equal.connection parallelly
```

```
##
## Attaching package: 'rsample'
```

```
## The following object is masked from 'package:e1071':
##
##     permutations
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 4.3.3
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.3.3
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.3
```

Problem 1

Suppose we have a new observation X = (A1 = Medium, A2 = Cool, A3 = East). Use the formulas presented in the Blackboard Module 2 reading (and reinforced during live class) to predict the class label of X using Naïve Bayes classification. You may use code to help you count how many you have of each level of each feature and do the math you need to do, but you may not use a Naïve Bayes function in any library, package, or other software. Reminder: your calculation must include probability information obtained from each of the three attributes. Be sure to submit your code with the assignment.

```r
# Read in data
data = read.csv('HW3_P1.csv')
print(data)
```

```
##    ID     A1   A2   A3 Class
## 1   1 Medium Mild East     Y
## 2   2    Low Mild East     Y
## 3   3   High Mild East     N
## 4   4    Low Mild West     N
## 5   5    Low Cool East     Y
## 6   6 Medium  Hot West     N
## 7   7   High  Hot East     Y
## 8   8    Low Cool West     N
## 9   9 Medium  Hot East     Y
## 10 10   High Cool East     Y
## 11 11 Medium Mild East     Y
## 12 12    Low Cool West     N
```

```r
## Calculate probability of yes given the data
# Formula: P(X | yes) * P(yes)
    # P(X | yes) = P(col = data | yes) * ...

# Filter for yes data
total_count = nrow(data)
yes_data = subset(data, Class == 'Y')
yes_data_count = nrow(yes_data)

# Count each class
yes_medium_count = table(yes_data$A1)["Medium"]
yes_cool_count = table(yes_data$A2)["Cool"]
yes_east_count = table(yes_data$A3)["East"]

# Calculate probabilities
P_yes = yes_data_count / total_count
P_A1_yes = yes_medium_count / yes_data_count
P_A2_yes = yes_cool_count / yes_data_count
P_A3_yes = yes_east_count / yes_data_count

# Multiply together to get P(yes | data)
P_yes_data = P_yes * P_A1_yes * P_A2_yes * P_A3_yes
print('Probability of yes given data:')
```

```
## [1] "Probability of yes given data:"
```

```r
print(P_yes_data)
```

```
##     Medium
## 0.07142857
```

```r
## Calculate probability of no

# Filter for no data
total_count = nrow(data)
no_data = subset(data, Class == 'N')
no_data_count = nrow(no_data)
```

```r
# Count each class
no_medium_count = table(no_data$A1)["Medium"]
no_cool_count = table(no_data$A2)["Cool"]
no_east_count = table(no_data$A3)["East"]

# Calculate probabilities
P_no = no_data_count / total_count
P_A1_no = no_medium_count / no_data_count
P_A2_no = no_cool_count / no_data_count
P_A3_no = no_east_count / no_data_count

# Multiply together to get P(yes | data)
P_no_data = P_no * P_A1_no * P_A2_no * P_A3_no
print('Probability of no given data:')
```

```
## [1] "Probability of no given data:"
```

```r
print(P_no_data)
```

```
##      Medium
## 0.006666667
```

After all calculations are done, the probability that the new observation is a yes is 0.0714 and the probability that it is a no is 0.00667. Because the probability of yes given data is higher, the naive bayes model would classify it as yes.

Problem 2

Use the same dataset as in Problem 1. Calculate the information gain (based on change of entropy) for A2 and A3 and determine which of these two is better as the test attribute at the root. You may use code to help you with this task, but you may not use a decision tree algorithm in any library, package, or other software. Be sure to submit your code with the assignment.

```r
# Use same data as problem 1

## Calculate entropy for initial data

# We already have yes_data_count, no_data_count, and total_count
# We also have the proportions: P_yes and P_no
ent_class = -(P_yes*log2(P_yes) + P_no*log2(P_no))

## Calculate entropy of A2

# Find all splits of A2
A2_splits = table(data$A2)

# Calculate proportion of each
A2_Cool = sum(data$A2 == 'Cool') / total_count
A2_Hot = sum(data$A2 == 'Hot') / total_count
A2_Mild = sum(data$A2 == 'Mild') / total_count

# Find the yes's and no's of each split (proportion)
A2_Cool_yes = sum(data$A2 == 'Cool' & data$Class == 'Y') / sum(data$A2 == 'Cool')
A2_Cool_no = sum(data$A2 == 'Cool' & data$Class == 'N') / sum(data$A2 == 'Cool')
```

```r
A2_Hot_yes = sum(data$A2 == 'Hot' & data$Class == 'Y') / sum(data$A2 == 'Hot')
A2_Hot_no = sum(data$A2 == 'Hot' & data$Class == 'N') / sum(data$A2 == 'Hot')
A2_Mild_yes = sum(data$A2 == 'Mild' & data$Class == 'Y') / sum(data$A2 == 'Mild')
A2_Mild_no = sum(data$A2 == 'Mild' & data$Class == 'N') / sum(data$A2 == 'Mild')

# Entropy calculation
ent_A2_Cool = -(A2_Cool_yes*log2(A2_Cool_yes) + A2_Cool_no*log2(A2_Cool_no))
ent_A2_Hot = -(A2_Hot_yes*log2(A2_Hot_yes) + A2_Hot_no*log2(A2_Hot_no))
ent_A2_Mild = -(A2_Mild_yes*log2(A2_Mild_yes) + A2_Mild_no*log2(A2_Mild_no))
ent_A2 = A2_Cool * ent_A2_Cool + A2_Hot * ent_A2_Hot + A2_Mild * ent_A2_Mild

# Entropy gain calculation
ent_A2_gain = ent_class - ent_A2
print('A2 entropy gain:')
```

```
## [1] "A2 entropy gain:"
```

```r
print(ent_A2_gain)
```

```
## [1] 0.01239872
```

```r
## Calculate entropy of A3

# Find all splits of A3
A3_splits = table(data$A3)

# Calculate proportion of each
A3_East = sum(data$A3 == 'East') / total_count
A3_West = sum(data$A3 == 'West') / total_count

# Find the yes's and no's of each split (proportion)
A3_East_yes = sum(data$A3 == 'East' & data$Class == 'Y') / sum(data$A3 == 'East')
A3_East_no = sum(data$A3 == 'East' & data$Class == 'N') / sum(data$A3 == 'East')
A3_West_yes = sum(data$A3 == 'West' & data$Class == 'Y') / sum(data$A3 == 'West')
A3_West_no = sum(data$A3 == 'West' & data$Class == 'N') / sum(data$A3 == 'West')

# Entropy calculation
ent_A3_East = -(A3_East_yes*log2(A3_East_yes) + A3_East_no*log2(A3_East_no))
ent_A3_West = -(A3_West_yes*log2(A3_West_yes) + A3_West_no*log2(A3_West_no))
ent_A3_West_skip = -(0 + A3_West_no*log2(A3_West_no)) ## prevent R from turning into NAN
ent_A3 = A3_East * ent_A3_East + A3_West * ent_A3_West_skip

# Entropy gain calculation
ent_A3_gain = ent_class - ent_A3
print('A3 entropy gain:')
```

```
## [1] "A3 entropy gain:"
```

```r
print(ent_A3_gain)
```

```
## [1] 0.6174925
```

The entropy gain by A2 is 0.0124 and by A3 is 0.6175. Thus A3 is better to split on because the gain is higher.

Problem 3

(1). Generate training and holdout partitions on the data set. Use 1/3 of the data in the holdout. (2). Fit a Naïve Bayes model. Use the model to make predictions on the holdout data. Generate a confusion matrix and measure the model's performance using one or more appropriate metrics of your choice. (3). Does this seem to be a good model? Discuss why or why not.

```
# Load data
autism_data = read.csv('autism-adult.csv')

# Set label as factor
autism_data$Class.ASD <- factor(autism_data$Class.ASD)

# Split 2/3 for train and 1/3 for test
set.seed(42)
split <- initial_split(autism_data, prop = 2/3, strata = Class.ASD)
train <- training(split)
test <- testing(split)

# Create naive bayes model
model_nb <- naiveBayes(Class.ASD ~ ., data = autism_data)

# Predict on test data
pred <- predict(model_nb, newdata = test, type = "class")

# Create confusion matrix
performance_measures  <- confusionMatrix(data=pred,
                                         reference = test$Class.ASD,
                                         positive = 'YES')
performance_measures
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  NO YES
##        NO  170   7
##        YES   2  56
##
##                Accuracy : 0.9617
##                  95% CI : (0.9285, 0.9823)
##     No Information Rate : 0.7319
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8999
##
##  Mcnemar's Test P-Value : 0.1824
##
##             Sensitivity : 0.8889
##             Specificity : 0.9884
##          Pos Pred Value : 0.9655
##          Neg Pred Value : 0.9605
##              Prevalence : 0.2681
##          Detection Rate : 0.2383
##    Detection Prevalence : 0.2468
##       Balanced Accuracy : 0.9386
```

```
##
##        'Positive' Class : YES
##
```

I would argue that this is a good model. We have an accuracy of 0.9617 which is very high. While accuracy isn't always the best measure this is still very good. Other good metrics are Sensitivity and Specificity which classify the true positive rate and the true negative rate respectively. These are at 0.8889 and 0.9884 which are both very high and indicate we are correctly classifying most datapoints.

Problem 4

(1). Generate training and holdout partitions on the data set. Use 1/3 of the data in the holdout. (2). Fit a regression tree model with Complexity Parameter of 0. Use the tree to make predictions on the holdout data. Measure the tree's performance using one or more appropriate metrics of your choice. (3). Prune the tree by implementing the minimum cross-validation error method. Use the pruned tree to make predictions on the holdout data. Measure the tree's performance using one or more appropriate metrics of your choice. (4). Compare the pruned tree to the full tree. Which, if any, seems to perform better? Discuss. (5.) Calculate variable importance for the pruned tree. What are the most important considerations if you want to start a restaurant that generates a large amount of revenue?

```
# Load data (predictive column = Revenue)
rest_data = read.csv('restaurantdata-small.csv')

# Train test split
set.seed(42)
split <- initial_split(rest_data, prop = 2/3, strata = Revenue)
train <- training(split)
test <- testing(split)

# Fit regression tree model with cp = 0
rpart.tree <- rpart(Revenue ~ ., data = train, method = "anova", cp = 0,
                    parms = list(split = "information"))

# Measure performance
pred <- predict(rpart.tree, newdata = test)
caret::postResample(pred, test$Revenue)
```

```
##         RMSE     Rsquared          MAE
## 9.145392e+04 8.846747e-01 7.311721e+04
```

```
# Prune Tree
best <- rpart.tree$cptable[which.min(rpart.tree$cptable[,"xerror"]), "CP"]
pruned.tree <- prune(rpart.tree, cp=best)

# Measure performance
pred <- predict(pruned.tree, newdata = test)
caret::postResample(pred, test$Revenue)
```

```
##         RMSE     Rsquared          MAE
## 8.081399e+04 9.097155e-01 6.591496e+04
```

```
# Calculate variable importance
imp_var = pruned.tree$variable.importance
print(imp_var)
```

```
##               Cuisine              Location Service.Quality.Score
##          1.831587e+14          1.830494e+14          5.088146e+12
##        Ambience.Score                Rating
##          2.928534e+12          1.591521e+12
```

(2). Fit a regression tree model with Complexity Parameter of 0. Use the tree to make predictions on the holdout data. Measure the tree's performance using one or more appropriate metrics of your choice.

An R squared of 0.8846 is typically very good as this means that 88% of the variance of the data is explained by the model.

(3). Prune the tree by implementing the minimum cross-validation error method. Use the pruned tree to make predictions on the holdout data. Measure the tree's performance using one or more appropriate metrics of your choice.

The pruned tree has an R squared of 0.9097 which means that 90.9% of the the variance in the data is explained by the model. This indicates a very good model.

(4). Compare the pruned tree to the full tree. Which, if any, seems to perform better? Discuss.

The R squared for the pruned tree is higher which indicates that the pruned model is performing better. This makes sense as we aren't finding every split of the data and rather looking at overall patterns.

(5.) Calculate variable importance for the pruned tree. What are the most important considerations if you want to start a restaurant that generates a large amount of revenue?

The top 3 considerations are Cuisine, Location, and Service Quality. These are the factors that a resturant should focus on (according to the model) to have the highest revenue.