CS 346 Final Project: Horse Colic

Eliot Stanton

Spring 2020

Loading required libraries:

```r
library(mosaic)
library(rpart)
library(rpart.plot)
library(mlbench)
library(caret)
library(randomForest)
library(caTools)
library(e1071)
library(neuralnet)
library(tidyverse)
```

DATA CLEANING

Loading data set, consolidating it, adding column names:

```r
horse.colic <- read.table("~/Desktop/college/Spring 2020/DataMining/data/horse-colic.data", quote="\"",
#View(horse.colic)
horse.colic.test <- read.table("~/Desktop/college/Spring 2020/DataMining/data/horse-colic.test", quote=
#View(horse.colic.test)
horse <- rbind(horse.colic,horse.colic.test)
names(horse) <- c("surgery","age","horseID","rectaltemp","pulse","resprate","extrtemp","periphpulse","mu
horse[horse=="?"]=NA
#View(horse)
```

Fixing variable types and levels:

```r
horse$surgery <- factor(horse$surgery,levels=c(1,2),labels=c("yes","no"))

horse$age <- factor(horse$age,levels=c(9,1),labels=c("<6mos","adult"))

horse$horseID <- as.character(horse$horseID)

horse$rectaltemp <- as.numeric(as.character(horse$rectaltemp))

horse$pulse <- as.numeric(as.character(horse$pulse))

horse$resprate <- as.numeric(as.character(horse$resprate))

horse$extrtemp <- factor(horse$extrtemp,levels=c(4,3,1,2),labels=c("cold","cool","normal","warm"))
```

```
horse$periphpulse <- factor(horse$periphpulse,levels=c(4,3,1,2),labels=c("absent","reduced","normal","in

horse$mucous <- factor(horse$mucous,levels=c(1,2,3,4,6,5),labels=c("normal pink","bright pink","pale pin

horse$capref <- factor(horse$capref,levels=c(1,2),labels=c("<3secs",">=3secs"))

horse$pain <- as.numeric(as.character(horse$pain))

horse$peristalsis <- factor(horse$peristalsis,levels=c(4,3,2,1),labels=c("absent","hypomotile","normal"

horse$abdist <- factor(horse$abdist,levels=c(1,2,3,4),labels=c("none","slight","moderate","severe"))

horse$nasogastube <- factor(horse$nasogastube,levels=c(1,2,3),labels=c("none","slight","significant"))

horse$nasogasref <- factor(horse$nasogasref,levels=c(1,3,2),labels=c("none","<1L",">1L"))

horse$nasogasrefph <- as.numeric(as.character(horse$nasogasrefph))

horse$feces <- factor(horse$feces,levels=c(4,3,1,2),labels=c("absent","decreased","normal","increased")

horse$abdomen <- factor(horse$abdomen,levels=c(1,2,3,4,5),labels=c("normal","other","firminlarge","dist

horse$cellvol <- as.numeric(as.character(horse$cellvol))

horse$protein <- as.numeric(as.character(horse$protein))

horse$abcentap <- factor(horse$abcentap,levels=c(1,2,3),labels=c("clear","cloudy","serosanguinous"))

horse$abcenpro <- as.numeric(as.character(horse$abcenpro))

horse$outcome <- factor(horse$outcome,levels=c(1,2,3),labels=c("lived","died","euthanized"))

horse$surgles <- factor(horse$surgles,labels=c("yes","no"))

horse$pathdata <- factor(horse$pathdata,labels=c("yes","no"))

horse <- mutate(horse,lived=ifelse(outcome=="lived","yes","no"))

horse <- horse[!is.na(horse$lived),]
```

Code to print all unique values of each variable (not run here):

Creating subsets/mutated versions of the data (making variables numeric, replacing missing values):

```
#all the categorical variables including lived
cat <- select(horse,c("surgery","age","extrtemp","periphpulse","mucous","capref","peristalsis","abdist"

#copy of categorical variables to regroup them and get rid of missing values
categorical <- cat

categorical$abcentap <- fct_collapse(categorical$abcentap,normal=c("clear"),abnormal=c("cloudy","serosan
for(j in 1:length(categorical$abcentap)){
    if(is.na(categorical$abcentap[j])){
      categorical$abcentap[j]="normal"
    }
}

categorical$extrtemp <- fct_collapse(categorical$extrtemp,cold=c("cool","cold"))
for(j in 1:length(categorical$extrtemp)){
    if(is.na(categorical$extrtemp[j])){
      categorical$extrtemp[j]="normal"
    }
}

categorical$periphpulse <- fct_collapse(categorical$periphpulse,normal=c("normal","increased"),poor=c("
for(j in 1:length(categorical$periphpulse)){
    if(is.na(categorical$periphpulse[j])){
      categorical$periphpulse[j]="normal"
    }
}

categorical$mucous <- fct_collapse(categorical$mucous,normal=c("normal pink","bright pink"),bad=c("pale
for(j in 1:length(categorical$mucous)){
    if(is.na(categorical$mucous[j])){
      categorical$mucous[j]="normal"
    }
}

categorical$peristalsis <- fct_collapse(categorical$peristalsis,normal=c("normal","hypermotile"),slow=c
for(j in 1:length(categorical$peristalsis)){
    if(is.na(categorical$peristalsis[j])){
      categorical$peristalsis[j]="normal"
    }
}

categorical$abdist <- fct_collapse(categorical$abdist,low=c("normal","slight"),high=c("moderate","severe
```

```
## Warning: Unknown levels in `f`: normal
```

```
for(j in 1:length(categorical$abdist)){
  if(is.na(categorical$abdist[j])){
    categorical$abdist[j]="low"
  }
}

categorical$nasogastube <- fct_collapse(categorical$nasogastube,low=c("none","slight"),high=c("significa
for(j in 1:length(categorical$nasogastube)){
  if(is.na(categorical$nasogastube[j])){
    categorical$nasogastube[j]="low"
  }
}

categorical$nasogasref<- fct_collapse(categorical$nasogasref,gas=c("<1L",">1L"))
for(j in 1:length(categorical$nasogasref)){
  if(is.na(categorical$nasogasref[j])){
    categorical$nasogasref[j]="none"
  }
}

categorical$feces <- fct_collapse(categorical$feces,normal=c("normal","increased"),less=c("decreased","a
for(j in 1:length(categorical$feces)){
  if(is.na(categorical$feces[j])){
    categorical$feces[j]="normal"
  }
}

categorical$abdomen <- fct_collapse(categorical$abdomen,normal=c("normal","other","firminlarge"),distend
for(j in 1:length(categorical$abdomen)){
  if(is.na(categorical$abdomen[j])){
    categorical$abdomen[j]="normal"
  }
}

for(j in 1:length(categorical$capref)){
  if(is.na(categorical$capref[j])){
    categorical$capref[j]="<3secs"
  }
}

#all the numeric variables
numeric <- select(horse,c("rectaltemp","pulse","resprate","pain","nasogasrefph","cellvol","protein","abd

#all numeric variables with no missing values
numnona <- numeric
for (i in 1:ncol(numnona)){
  average <- mean(numnona[,i],na.rm=TRUE)
  for(j in 1:nrow(numnona)){
    if(is.na(numnona[j,i])){
      numnona[j,i]=average
    }
  }
}

#all the categorical variables turned numeric except lived
cattonum <- categorical
n=1
while(n < ncol(categorical)){
  cattonum[,n] <- as.numeric(categorical[[n]])
  n=n+1
}

#all variables in numeric form except lived, no missing values
allnum <- cbind(numnona,cattonum)

#all originally numeric variables with no missing values, plus lived
num <- mutate(numnona,lived=horse$lived)

#no missing values at all,both numeric and categorical variables
ht <- data.frame(categorical,numnona)
```

```
ht$lived <- as.factor(ht$lived)
```

Creating new variable for number of morbidity factors a horse has:

```
ht$indicators <- 0
for(i in 1:nrow(ht)){
  n = 0
  a <- ht[i,]
  if(a$extrtemp=="cold"){n=n+1}
  if(a$mucous=="bad"){n=n+1}
  if(a$capref==">=3secs"){n=n+1}
  if(a$periphpulse=="poor"){n=n+1}
  if(a$peristalsis=="slow"){n=n+1}
  if(a$abdist=="high"){n=n+1}
  if(a$abdomen=="distended"){n=n+1}
  if(a$abcentap=="abnormal"){n=n+1}
  if(a$feces=="less"){n=n+1}
  if(a$cellvol>=50){n=n+1}
  if(a$pulse>=60){n=n+1}
  if(a$pain>=3){n=n+1}
  if(a$protein>10){n=n+1}
  if(a$abcenpro>3){n=n+1}
  ht$indicators[i]=n
}
#adding to the proper datasets besides ht
num<- mutate(num,indicators=ht$indicators)
allnum<- mutate(allnum,indicators=ht$indicators)
numnona<- mutate(numnona,indicators=ht$indicators)
```

FEATURE SELECTION/IMPORTANCE

Variable importance of categorical variables only: Broken down by each level separately, so less useful, but mucous and abdist are at the top.

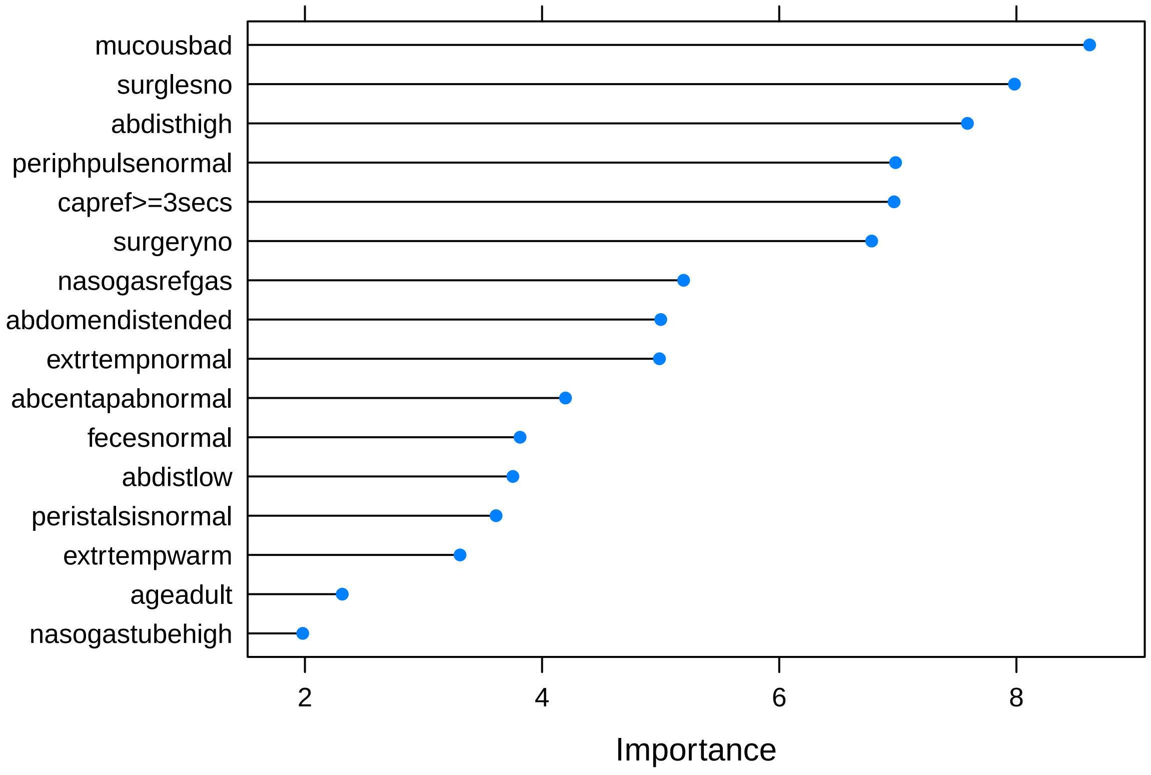
```
control <- trainControl(method="repeatedcv", number=10, repeats=3)
model <- train(lived~., data=categorical, method="rf", preProcess="scale", trControl=control,na.action=n
print(model)
```

```
## Random Forest
##
## 366 samples
##  14 predictor
##   2 classes: 'no', 'yes'
##
## Pre-processing: scaled (16)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 330, 329, 329, 330, 329, 330, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7109952  0.3807616
##    9    0.6834400  0.3331209
##   16    0.6887717  0.3462874
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
# estimate variable importance
importance <- varImp(model, scale=FALSE)
# summarize importance
print(importance)
```

```
## rf variable importance
##
##                    Overall
## mucousbad            8.618
## surglesno            7.984
## abdisthigh           7.587
## periphpulsenormal    6.981
## capref>=3secs        6.968
## surgeryno            6.780
## nasogasrefgas        5.194
## abdomendistended     5.001
## extrtempnormal       4.990
## abcentapabnormal     4.198
## fecesnormal          3.814
## abdistlow            3.754
## peristalsisnormal    3.612
## extrtempwarm         3.308
## ageadult             2.315
## nasogastubehigh      1.982
```

```r
# plot importance
plot(importance)
```



Variable importance of numeric variables only: Pulse, cellvol, and indicators are most important

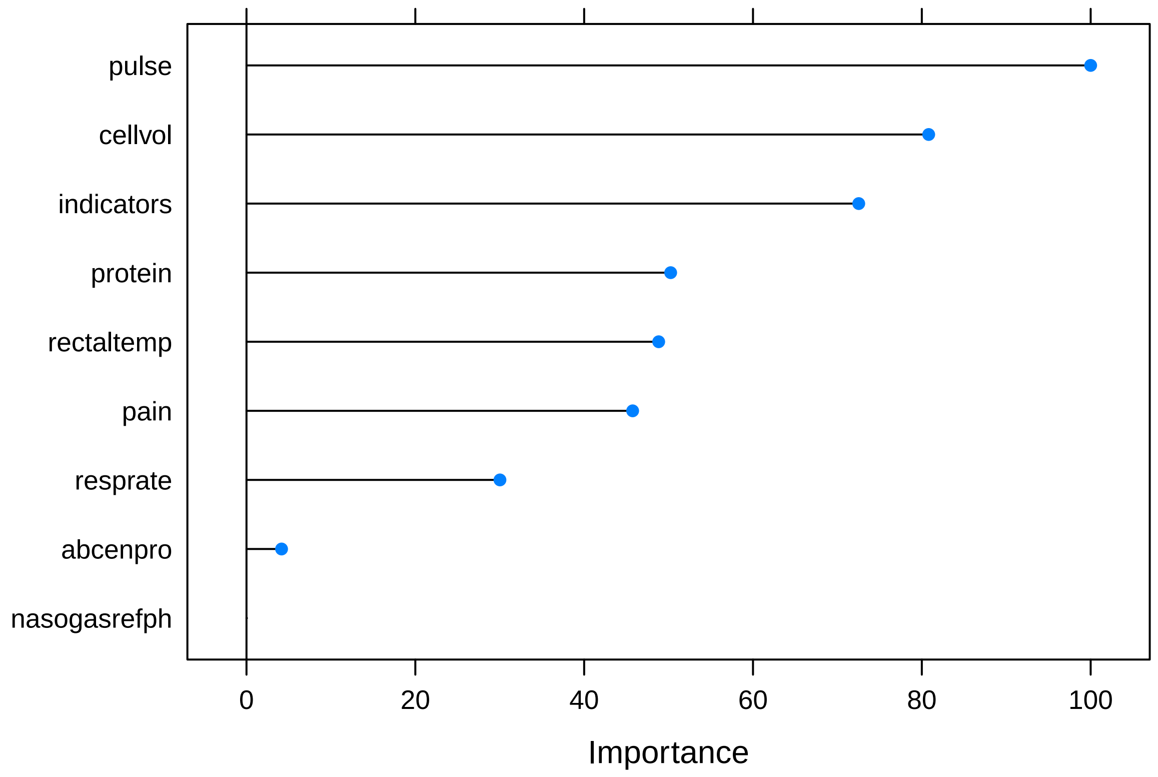
```
model <- train(lived~., data=num, method="rf", preProcess="scale",na.action=na.omit)
print(model)
```

```
## Random Forest
##
## 366 samples
##   9 predictor
##   2 classes: 'no', 'yes'
##
## Pre-processing: scaled (9)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 366, 366, 366, 366, 366, 366, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7233706  0.4059135
##   5     0.7148669  0.3924753
##   9     0.7128969  0.3887530
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
# estimate variable importance
importance <- varImp(model, scale=TRUE)
# summarize importance
print(importance)
```

```
## rf variable importance
##
##               Overall
## pulse         100.000
## cellvol        80.815
## indicators     72.529
## protein        50.249
## rectaltemp     48.828
## pain           45.746
## resprate       30.026
## abcenpro        4.155
## nasogasrefph    0.000
```

```r
# plot importance
plot(importance)
```



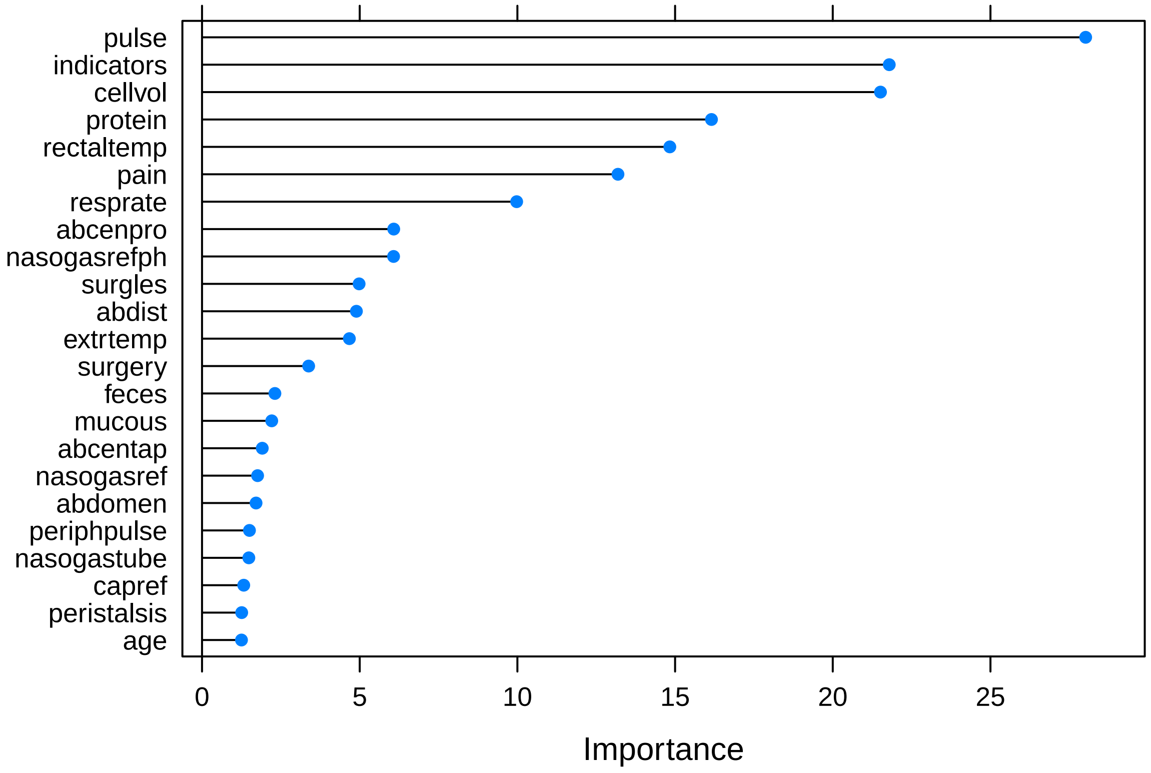
Variable importance of both categorical and numeric variables, treating categorical variables as numeric: Consistently important variables are pulse, indicators, cellvol, protein.

```
control <- trainControl(method="repeatedcv", number=10, repeats=3)

#View(num)
model <- train(lived~., data=allnum, method="rf", preProcess="scale", trControl=control,na.action=na.omi
# estimate variable importance
importance <- varImp(model, scale=FALSE)
# summarize importance
print(importance)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 23)
##
##              Overall
## pulse         28.020
## indicators    21.791
## cellvol       21.513
## protein       16.154
## rectaltemp    14.834
## pain          13.189
## resprate       9.977
## abcenpro       6.079
## nasogasrefph   6.073
## surgles        4.981
## abdist         4.895
## extrtemp       4.672
## surgery        3.381
## feces          2.310
## mucous         2.210
## abcentap       1.910
## nasogasref     1.762
## abdomen        1.713
## periphpulse    1.504
## nasogastube    1.484
```

```
# plot importance
plot(importance)
```



Correlation of numeric variables only: None of them are so highly-correlated that any variables should be avoided.

```
correlationMatrix <- cor(numnona)
# summarize the correlation matrix
print(correlationMatrix)
```

```
##              rectaltemp       pulse    resprate        pain nasogasrefph
## rectaltemp   1.00000000  0.19284897  0.21704657 -0.05251035   0.10500217
## pulse        0.19284897  1.00000000  0.41163435  0.29606015   0.01875541
## resprate     0.21704657  0.41163435  1.00000000  0.11689747   0.03690279
## pain        -0.05251035  0.29606015  0.11689747  1.00000000  -0.03189290
## nasogasrefph 0.10500217  0.01875541  0.03690279 -0.03189290   1.00000000
## cellvol      0.06393217  0.37360792  0.06455495  0.17940011  -0.04241745
## protein     -0.01269051 -0.06662805 -0.07030331 -0.08110423  -0.27629548
## abcenpro    -0.01170339  0.04282889 -0.02720787 -0.10174086   0.07184466
## indicators   0.04878583  0.51952946  0.12223760  0.50174100  -0.02742615
##                 cellvol     protein    abcenpro  indicators
## rectaltemp   0.06393217 -0.01269051 -0.01170339  0.04878583
## pulse        0.37360792 -0.06662805  0.04282889  0.51952946
## resprate     0.06455495 -0.07030331 -0.02720787  0.12223760
## pain         0.17940011 -0.08110423 -0.10174086  0.50174100
## nasogasrefph -0.04241745 -0.27629548  0.07184466 -0.02742615
## cellvol      1.00000000 -0.08419665  0.08523633  0.50456080
## protein     -0.08419665  1.00000000 -0.29294699  0.03695827
## abcenpro     0.08523633 -0.29294699  1.00000000  0.05535058
## indicators    0.50456080  0.03695827  0.05535058  1.00000000
```

```
# find attributes that are highly corrected (ideally >0.75, but here 0.5
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.25,  verbose = TRUE, names = TRUE)
```

```
## Compare row 2  and column  9 with corr  0.52
##   Means:  0.24 vs 0.136 so flagging column 2
## Compare row 9  and column  6 with corr  0.505
##   Means:  0.185 vs 0.115 so flagging column 9
## Compare row 7  and column  8 with corr  0.293
##   Means:  0.136 vs 0.096 so flagging column 7
## All correlations <= 0.25
```

```
# print indexes of highly correlated attributes
print(highlyCorrelated)
```

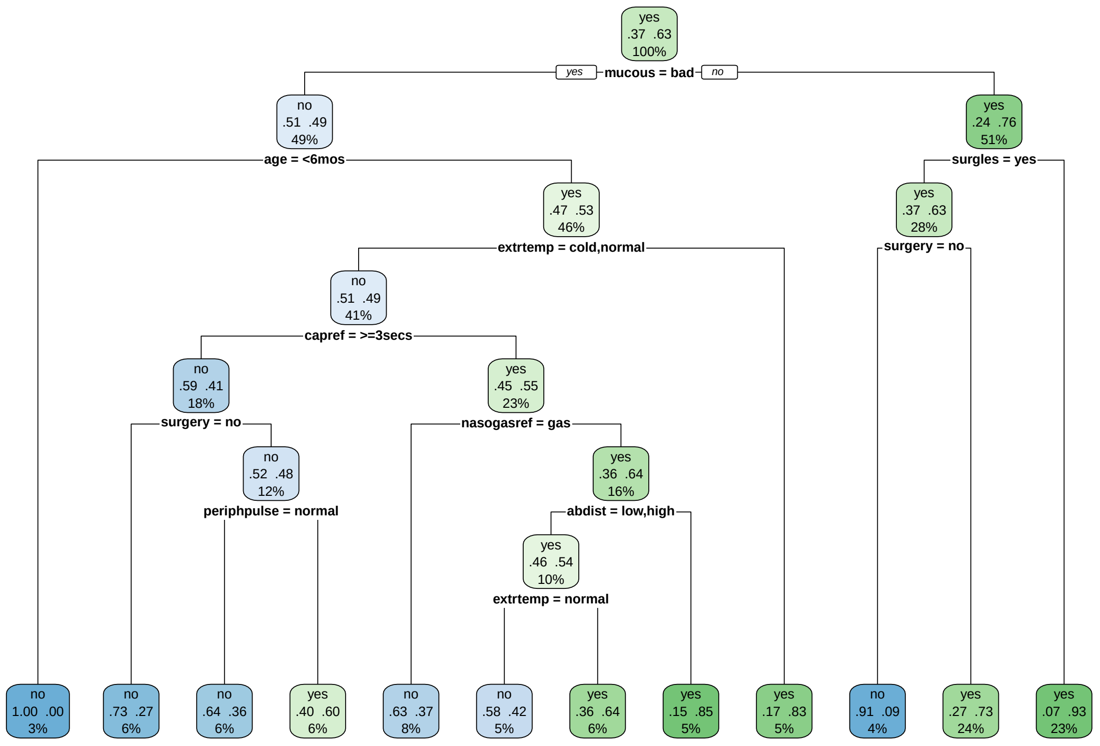
```
## [1] "pulse"      "indicators" "protein"
```

Takeaways from feature selection- When looking at numeric and categorical variables together, the originally numeric variables show up as more important than the categorical-turned-numeric variables. In models, still choose some of the top categorical variables from looking at just categorical variables, in case there is some bias against them in the combined importance calculation. Looking at the information on the variables provided along with the dataset can also help with feature selection. For example, it says abdist is a very important variable. And, it says that pain should not be considered as quantitative, so I'd avoid that variable even if it shows up as important in the calculation. Also looked at tally tables (in percent form) for the categorical variables to see the different distributions among horses that lived versus those that died. No one variable was wildly awesome. Best case scenario had 25% of horses that lived exhibiting a morbidity factor with 50% of those that died exhibiting it too. Adding the indicators variable for number of morbidity factors was a good idea! This helps since horses have many different combinations of morbidity factors (no one obvious predictor) but overall some have far more than others.

MODELS

Tree of categorical variables only: Did this as more of a test than for evaluating performance.

```
s <- sample(368,250)
train <- categorical[s,]
test <- categorical[-s,]
#c <- rpart.control(minsplit=10,cp=.0001,maxsurrogate=0,maxdepth=5)
nmm <- rpart(lived~.,data=train)#,control=c)
rpart.plot(nmm,extra=104)#,box.palette="Blues")
```

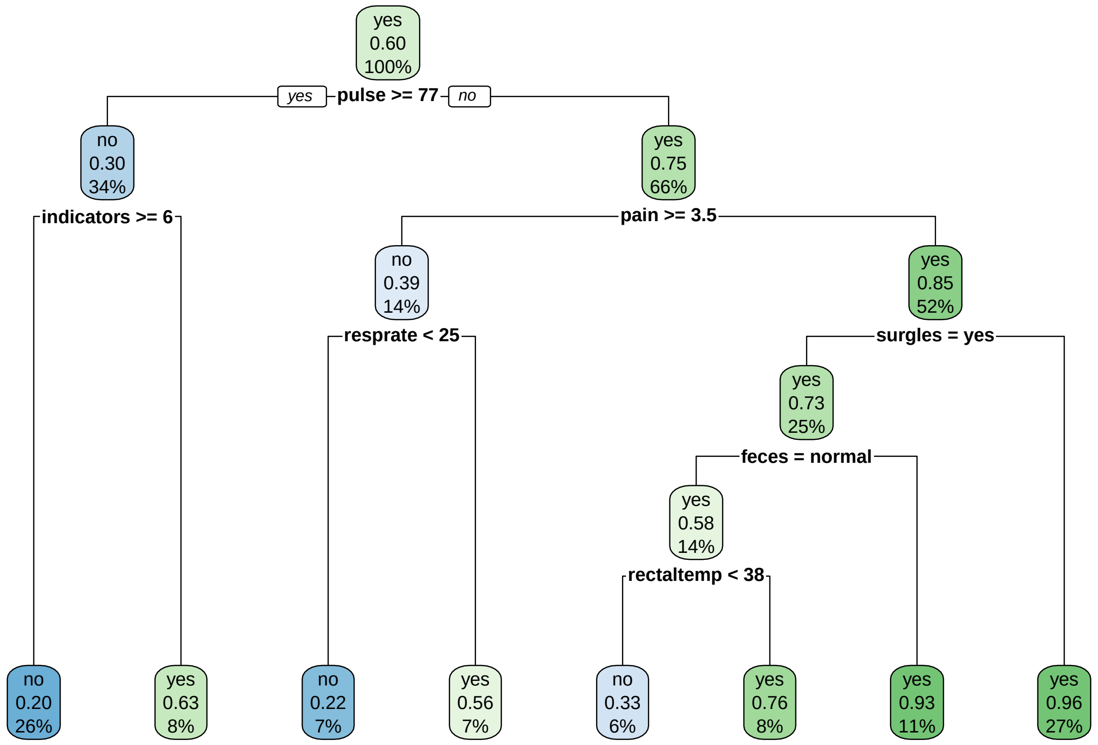


```
p<- predict(nmm,test,type="class")
table(test[,14],p)
```

```
##      p
##        no yes
##   yes 24  52
##   no   8  34
```

Tree with all variables: Accuracy most commonly around 67%, but ranging 55-75%. Fiddling with parameters honestly didn't get wildly different results for me…

```
s <- sample(366,250)
train <- ht[s,]
test <- ht[-s,]
#c <- rpart.control(minsplit=1,cp=.05,maxsurrogate=3,maxdepth=5)
htcart <- rpart(lived~.,data=train,method="class")#,control=c)
rpart.plot(htcart)
```



```
p<- predict(htcart,test,type="class")
tablemat <- table(test[,15],p)
tablemat
```

```
##      p
##        no yes
##    no  24  16
##    yes 18  58
```

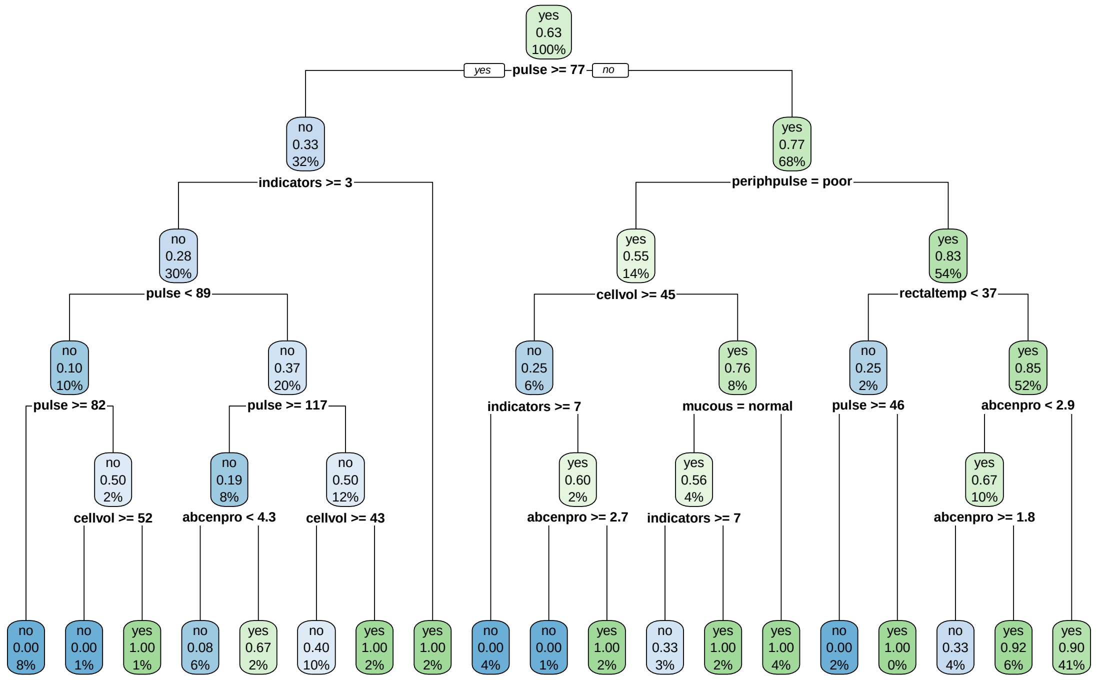
```
accuracy <- sum(diag(tablemat))/sum(tablemat)
accuracy
```

```
## [1] 0.7068966
```

Tree with most important variables: Tried just with indicators and cellvol… accuracy around 68%. Accuracy isn't much better with the top 9 or so variables… similar mean accuracy (65%) but higher range in both good and bad directions (60-75%)

```
#ht <- data.frame(categorical,numnona)
s <- sample(366,200)
train <- ht[s,]
test <- ht[-s,]
c <- rpart.control(minsplit=1,cp=.01,maxsurrogate=3,maxdepth=5)
htcart <- rpart(lived~indicators+mucous+pulse+periphpulse+abdist+abdomen+cellvol+rectaltemp+abcenpro,da
```

```r
#htcart <- rpart(lived~indicators+cellvol,data=train,method="class",control=c)
rpart.plot(htcart)
```



```r
p<- predict(htcart,test,type="class")
tablemat <- table(test[,15],p)
tablemat
```

```
##      p
##        no yes
##   no   47  20
##   yes 25  74
```

```r
accuracy <- sum(diag(tablemat))/sum(tablemat)
accuracy
```

```
## [1] 0.7289157
```

Random Forest: Produces widely variable results (65% to 90%), but mostly 75-80%. Interesting that range of results is wider than it is for a single tree… Using all variables here.

```r
s <- sample(366,250)
train <- ht[s,]
test <- ht[-s,]
```

```
rf <- randomForest(lived~.,data=train)
#take a look
rf
```

```
##
## Call:
##  randomForest(formula = lived ~ ., data = train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 24%
## Confusion matrix:
##     no yes class.error
## no  66  32   0.3265306
## yes 28 124   0.1842105
```

```
#pred and evaluate
pred = predict(rf, newdata=test[-15])
cm = table(test[,15], pred)
cm
```

```
##      pred
##       no yes
##   no  28  15
##   yes 20  53
```

```
accuracy_Test <- sum(diag(cm)) / sum(cm)
accuracy_Test
```

```
## [1] 0.6982759
```

Support Vector Machine: Accuracy in the low 60s, around 63%.

```
s <- sample(366,250)
train <- ht[s,]
test <- ht[-s,]

model <- svm(lived~indicators+mucous+pulse+periphpulse+abdist+abdomen+cellvol+rectaltemp+abcenpro,data=
print(model)
```

```
##
## Call:
## svm(formula = lived ~ indicators + mucous + pulse + periphpulse +
##     abdist + abdomen + cellvol + rectaltemp + abcenpro, data = train,
##     probability = TRUE, cost = 100, gamma = 1, kernel = "sigmoid")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  sigmoid
##        cost:  100
##      coef.0:  0
##
## Number of Support Vectors:  105
```

```r
summary(model)
```

```
##
## Call:
## svm(formula = lived ~ indicators + mucous + pulse + periphpulse +
##     abdist + abdomen + cellvol + rectaltemp + abcenpro, data = train,
##     probability = TRUE, cost = 100, gamma = 1, kernel = "sigmoid")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  sigmoid
##        cost:  100
##      coef.0:  0
##
## Number of Support Vectors:  105
##
##  ( 53 52 )
##
##
## Number of Classes:  2
##
## Levels:
##  no yes
```

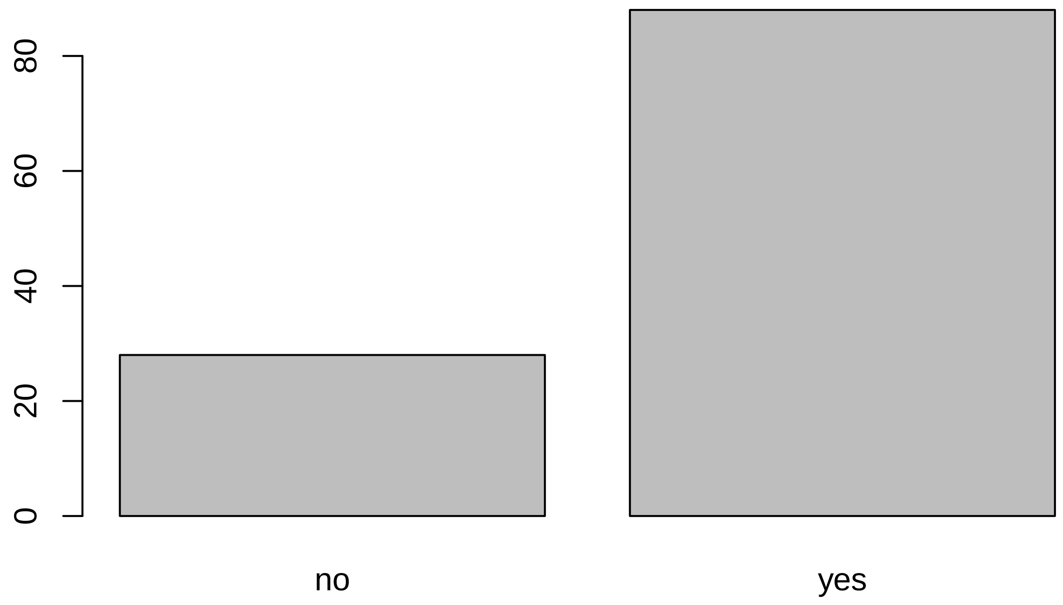
```r
# compute decision values and probabilites
pred <- predict(model, subset(test,select=-lived),decision.values = TRUE, probability = TRUE)
attr(pred, "decision.values")[c(-15),]
```

```
##            1            2            3            6            7           14
##   314.405068   482.590748  2024.544991    80.666018  2490.379170   140.286857
##           16           17           19           20           25           27
##  -410.221102    96.701543  2117.759077  1459.217380 -1051.571787     1.000023
##           29           37           42           45           46           50
##   740.378838   476.439703 -1119.547799  -431.736885   -94.764161  -972.544369
##           62           63           67           68           69           70
##   765.254943 -1680.518605 -1442.574328   935.575279   689.377286   955.840137
##           72           75           76           81           82           83
##  -235.383384   851.378889  1159.491578  -239.956890  1494.380523  -115.321055
##           86           88           92           95          104          108
##  1035.136624   586.125749  1798.408027  1655.950984  -370.269991  2027.590693
##          110          111          112          113          114          115
##    76.163996  -639.300581  1723.262039  2342.955414 -1216.264282  -670.836234
##          121          123          127          134          137          139
##  -855.338488   994.983862  -581.918951  2004.530367 -1182.583485  1826.253317
##          140          142          145          146          153          164
##   126.063088   159.325783 -1617.956899  -218.123622  -188.623633  -820.556608
##          165          167          168          173          174          180
##   425.971041  1930.397358  2063.241114   964.860479  -356.152568    95.407874
##          181          182          187          206          207          210
##   111.737836  1979.700851  -344.306344    95.407874   159.945282  1251.504323
##          212          213          223          225          227          229
## -1105.220722   311.111547   -84.255496   362.297682  1713.438168 -1925.109596
##          231          233          236          239          242          244
##   415.370462  1890.407005  -116.521110  -287.772056  1748.607336  -309.044148
##          245          247          253          254          255          257
##  -951.611471  -537.553569  1127.295116   112.306940 -1036.369384   548.109542
##          259          260          261          264          266          279
##  1393.555619   178.880710  2445.472481  -349.922462  1836.920064  1799.281499
##          283          287          294          295          297          298
##  -639.300581  -306.684202  -912.032098  1772.358513 -1057.556479 -1596.571420
##          299          302          306          319          320          323
##  -916.592348   348.532696  1438.795515   -41.474217   126.441947   561.807500
##          325          328          336          338          343          347
##  -549.105773  -234.812923  -863.470660  1822.884414  1598.672425  -535.253705
##          348          353          355          362          363          364
##  -953.094405  1748.565385  2454.597384  2058.205149  -313.674125  -288.230740
##          368
##   680.411701
```

```r
attr(pred, "probabilities")[c(-15),] #sometimes nice to have probabilities to put into another model
```

```
##           yes        no
## 1   0.6271985 0.3728015
## 2   0.6509577 0.3490423
## 3   0.8274823 0.1725177
## 6   0.5931549 0.4068451
## 7   0.8645104 0.1354896
## 14  0.6019384 0.3980616
## 16  0.5190643 0.4809357
## 17  0.5955234 0.4044766
## 19  0.8354826 0.1645174
## 20  0.7723371 0.2276629
## 25  0.4215017 0.5784983
## 27  0.5813257 0.4186743
## 29  0.6859340 0.3140660
## 37  0.6501010 0.3498990
## 42  0.4113820 0.5886180
## 45  0.5157731 0.4842269
## 46  0.5669826 0.4330174
## 50  0.4333496 0.5666504
## 62  0.6892077 0.3107923
## 63  0.3313869 0.6686131
## 67  0.3644390 0.6355610
## 68  0.7111057 0.2888943
## 69  0.6791642 0.3208358
## 70  0.7136494 0.2863506
## 72  0.5457229 0.4542771
## 75  0.7003950 0.2996050
## 76  0.7384516 0.2615484
## 81  0.5450282 0.4549718
## 82  0.7761026 0.2238974
## 83  0.5638881 0.4361119
## 86  0.7234728 0.2765272
## 88  0.6652278 0.3347722
## 92  0.8068002 0.1931998
## 95  0.7928316 0.2071684
## 104 0.5251710 0.4748290
## 108 0.8277485 0.1722515
## 110 0.5924892 0.4075108
## 111 0.4839942 0.5160058
## 112 0.7995229 0.2004771
## 113 0.8535792 0.1464208
## 114 0.3971137 0.6028863
## 115 0.4791710 0.5208290
## 121 0.4510585 0.5489415
## 123 0.7185246 0.2814754
## 127 0.4927776 0.5072224
## 134 0.8257249 0.1742751
## 137 0.4020640 0.5979360
## 139 0.8094453 0.1905547
## 140 0.5998486 0.4001514
## 142 0.6047297 0.3952703
## 145 0.3399331 0.6600669
## 146 0.5483429 0.4516571
## 153 0.5528148 0.4471852
## 164 0.4563398 0.5436602
## 165 0.6430358 0.3569642
## 167 0.8190924 0.1809076
## 168 0.8308402 0.1691598
## 173 0.7147773 0.2852227
## 174 0.5273272 0.4726728
## 180 0.5953325 0.4046675
## 181 0.5977403 0.4022597
## 182 0.8235251 0.1764749
## 187 0.5291358 0.4708642
## 206 0.5953325 0.4046675
## 207 0.6048204 0.3951796
## 210 0.7491916 0.2508084
## 212 0.4135089 0.5864911
## 213 0.6267266 0.3732734
## 223 0.5685624 0.4314376
## 225 0.6340328 0.3659672
## 227 0.7985565 0.2014435
## 229 0.2990636 0.7009364
## 231 0.6415437 0.3584563
## 233 0.8154338 0.1845662
## 236 0.5637073 0.4362927
## 239 0.5377554 0.4622446
## 242 0.8020001 0.1979999
## 244 0.5345144 0.4654856
## 245 0.4365013 0.5634987
## 247 0.4995719 0.5004281
## 253 0.7346242 0.2653758
## 254 0.5978241 0.4021759
## 255 0.4237743 0.5762257
## 257 0.6600214 0.3399786
## 259 0.7651867 0.2348133
## 260 0.6075896 0.3924104
## 261 0.8612556 0.1387444
## 264 0.5282785 0.4717215
## 266 0.8104512 0.1895488
## 279 0.8068836 0.1931164
## 283 0.4839942 0.5160058
## 287 0.5348741 0.4651259
## 294 0.4424742 0.5575258
## 295 0.8043005 0.1956995
## 297 0.4206080 0.5793920
## 298 0.3428789 0.6571211
## 299 0.4417852 0.5582148
## 302 0.6320739 0.3679261
## 306 0.7701298 0.2298702
## 319 0.5749795 0.4250205
## 320 0.5999043 0.4000957
## 323 0.6619019 0.3380981
## 325 0.4978026 0.5021974
## 328 0.5458095 0.4541905
## 336 0.4498253 0.5501747
## 338 0.8091267 0.1908733
## 343 0.7870088 0.2129912
## 347 0.4999241 0.5000759
## 348 0.4362778 0.5637222
## 353 0.8019960 0.1980040
## 355 0.8619222 0.1380778
## 362 0.8304062 0.1695938
## 363 0.5338086 0.4661914
## 364 0.5376855 0.4623145
## 368 0.6779662 0.3220338
```

```r
plot(pred)
```



```
table_mat <- table(test$lived, pred)
table_mat
```

```
##      pred
##        no yes
##   no   15  34
##   yes 13  54
```

```
accuracy_Test <- sum(diag(table_mat)) / sum(table_mat)
accuracy_Test
```

```
## [1] 0.5948276
```

Neural Network: Wide range in accuracy. Most commonly around 68% but ranging up to 78%. Once again adjusting parameters didn't make much of an impact.

```
l <- as.numeric(as.factor(allnum$lived))

a <- allnum
a$lived <- l

max = apply(a , 2 , max)
min = apply(a, 2 , min)
scaled = as.data.frame(scale(a, center = min, scale = max - min))
```

```
s <- sample(366,250)
train <- scaled[s,]
test <- scaled[-s,]

nn <- neuralnet(lived~indicators+mucous+pulse+periphpulse+abdist+abdomen+cellvol+rectaltemp+abcenpro,da
plot(nn)

## Prediction using neural network
prediction=predict(nn,test[,-23])

prob <- prediction
pred <- ifelse(prob>0.5, 1, 0)
pred
```

```
##     [,1]
## 4      0
## 12     0
## 14     0
## 15     1
## 19     0
## 23     1
## 29     1
## 30     1
## 31     0
## 38     1
## 49     0
## 53     1
## 55     0
## 66     1
## 67     0
## 75     1
## 76     1
## 80     0
## 83     1
## 93     1
## 98     1
## 102    0
## 104    0
## 110    1
## 111    0
## 112    1
## 124    1
## 130    1
## 133    1
## 135    0
## 136    0
## 141    0
## 143    1
## 144    0
## 145    1
## 154    0
## 155    1
## 158    1
## 160     1
## 161     1
## 162     1
## 165     0
## 170     1
## 172     0
## 176     1
## 177     1
## 178     0
## 180     0
## 181     1
## 186     0
## 190     1
## 191     0
## 192     0
## 194     1
## 196     1
## 198     1
## 199     1
## 202     1
## 205     1
## 207     1
## 210     0
## 211     1
## 212     0
## 214     1
## 217     0
## 220     1
## 222     0
## 223     1
## 226     0
## 227     1
## 228     0
## 233     1
## 235     0
## 238     1
## 240     1
## 241     1
## 244     0
## 247     1
## 248     1
## 250     0
## 251     0
## 253     1
## 255     1
## 256     1
## 264     1
## 267     1
## 269     1
## 277     1
## 280     1
## 281     0
## 284     1
## 290     1
## 295    1
## 298    0
## 302    1
## 304    1
## 307    1
## 308    1
## 311    1
## 312    0
## 313    1
## 316    1
## 320    1
## 321    1
## 325    1
## 336    1
## 337    1
## 339    0
## 343    0
## 351    1
## 356    1
## 357    1
## 360    1
## 363    1
## 364    0
## 366    0
```

```
cm <- table(test$lived,pred)
cm
```

```
##    pred
##       0  1
##   0 24 23
##   1 17 52
```

```
accuracy <- sum(diag(cm)) / sum(cm)
accuracy
```

```
## [1] 0.6551724
```

REFLECTION Getting to apply a dataset to several different models was definitely helpful. I found it most frustrating that most of my models had similarly mediocre accuracy, even when I changed the features I was using or the values of a model's parameters. Maybe my dataset wasn't the greatest, or this is something that's just hard to predict. I would be interested to put each case before a large animal vet and see if they have an intuitive understanding of rules that work better/compare their accuracy with my models... Honestly? I'm most proud that I got through all the errors just to get these models working in the first place. There were several that were really tricky. The way that I fixed them all was by removing missing values. I replaced missing values for numeric variables with the mean, and missing values for categorical variables with the normal value rather than the positive display of a morbidity factor. In hindsight, this definitely could have impacted my results, especially since the missing values were distributed fairly evenly among horses that lived and horses that died. I wish the models had worked better with missing values (instead of just omitting all rows with anything missing) so I could have applied what we learned about dealing with them instead of just removing them all. I'm not surprised that my random forest works best, since ensemble models frequently do better than simpler models, but it was surprising that there was more variability in its accuracy than there was in a single tree's. From here if I were to continue exploring I would be more systematic about changing parameters and recording resulting accuracy, perhaps running models 100 times and looking at the distribution of accuracy results to see if it's normal, skewed, with high or low spread. I think I'd also try more models with just one or two variables, since annoyingly those work just about as well as a model with only the best predictors or a model with every single predictor included.