Capstone Project- Choose Your Own

Steven Wild

09 March, 2022

Horse Colic Dataset

The report is organised into the following sections:

1. Introduction
    1. Variable description
2. Analysis
    1. Data download and preparation
    2. Training and test set preparation
    3. Training / modelling methods
        1. Random guess
        2. Linear regression
        3. Logistic regression
        4. K Nearest Neighbour
        5. Classification tree
        6. Random forest
        7. Ensemble
3. Results
4. Conclusion.

1 Introduction

This report forms a part of the project submission for the HarvardX PH125.9x course : Data Science: Capstone. The aim of the project is to develop a model, using machine learning techniques, to predict the survival of horses with colic, based on a number of medical observations.

The data "Horse Colic Data Set" is based on the data from the UCI Machine Learning Repository at this address link

The version of the data used comes from a sub-set of original and can be found here link

The data has 299 sets of observations using 28 attributes, a mix of continuous, discrete and nominal, with approximately 30% of the values missing. A copy of the data, in "CSV" format has been included in the Git repository link together with the ".R" script file, the ".RMD" report files and this ".PDF" copy of the report.

1.1 Variable description

the variables described in the table below are from a DATADICT file included with the dataset

Variable name	Description
Surgery	1 = Yes, it had surgery 2 = It was treated without surgery
Age	1 = Adult horse 2 = Young (< 6 months)
Hospital Number	numeric id or the case number assigned to the horse (may not be unique if the horse is treated > 1 time)
Rectal temperature	- linear - in degrees celsius. - An elevated temp may occur due to infection. - temperature may be reduced when the animal is in late shock - normal temp is 37.8 - this parameter will usually change as the problem progresses, eg. may start out normal, then become elevated because of the lesion, passing back through the normal range as the horse goes into shock
pulse	linear - the heart rate in beats per minute - is a reflection of the heart condition: 30 -40 is normal for adults - rare to have a lower than normal rate although athletic horses may have a rate of 20-25 - animals with painful lesions or suffering from circulatory shock may have an elevated heart rate
Respiratory rate	linear - normal rate is 8 to 10 - usefulness is doubtful due to the great fluctuations
Temperature of extremities	a subjective indication of peripheral circulation - possible values: 1 = Normal 2 = Warm 3 = Cool 4 = Cold - cool to cold extremities indicate possible shock - hot extremities should correlate with an elevated rectal temp.
Peripheral pulse	subjective - possible values are: 1 = normal 2 = increased 3 = reduced 4 = absent - normal or increased p.p. are indicative of adequate circulation while reduced or absent indicate poor perfusion
Mucous membranes	a subjective measurement of colour - possible values are: 1 = normal pink 2 = bright pink 3 = pale pink 4 = pale cyanotic 5 = bright red / injected 6 = dark cyanotic - 1 and 2 probably indicate a normal or slightly increased circulation - 3 may occur in early shock - 4 and 6 are indicative of serious circulatory compromise - 5 is more indicative of a septicemia
Capillary refill time	a clinical judgment. The longer the refill, the poorer the circulation - possible values 1 = < 3 seconds 2 = >= 3 seconds
Pain	a subjective judgment of the horse's pain level - possible values: 1 = alert, no pain 2 = depressed 3 = intermittent mild pain 4 = intermittent severe pain 5 = continuous severe pain - should NOT be treated as a ordered or discrete variable! - In general, the more painful, the more likely it is to require surgery - prior treatment of pain may mask the pain level to some extent
Peristalsis	an indication of the activity in the horse's gut. As the gut becomes more distended or the horse becomes more toxic, the activity decreases - possible values: 1 = hypermotile 2 = normal 3 = hypomotile 4 = absent
Abdominal distension	An IMPORTANT parameter. - possible values 1 = none 2 = slight 3 = moderate 4 = severe - an animal with abdominal distension is likely to be painful and have reduced gut motility. - a horse with severe abdominal distension is likely to require surgery just tio relieve the pressure

Variable name	Description
Nasogastric tube	this refers to any gas coming out of the tube - possible values: 1 = none 2 = slight 3 = significant - a large gas cap in the stomach is likely to give the horse discomfort
Nasogastric reflux	possible values 1 = none 2 = > 1 liter 3 = < 1 liter - the greater amount of reflux, the more likelihood that there is some serious obstruction to the fluid passage from the rest of the intestine
Nasogastric reflux PH	linear - scale is from 0 to 14 with 7 being neutral - normal values are in the 3 to 4 range
Rectal examination - feces	possible values 1 = normal 2 = increased 3 = decreased 4 = absent - absent feces probably indicates an obstruction
Abdomen	possible values 1 = normal 2 = other 3 = firm feces in the large intestine 4 = distended small intestine 5 = distended large intestine - 3 is probably an obstruction caused by a mechanical impaction and is normally treated medically - 4 and 5 indicate a surgical lesion
Packed cell volume	linear - the # of red cells by volume in the blood - normal range is 30 to 50. The level rises as the circulation becomes compromised or as the animal becomes dehydrated.
Total protein	linear - normal values lie in the 6-7.5 (gms/dL) range - the higher the value the greater the dehydration
Abdominocentesis appearance	a needle is put in the horse's abdomen and fluid is obtained from the abdominal cavity - possible values: 1 = clear 2 = cloudy 3 = serosanguinous - normal fluid is clear while cloudy or serosanguinous indicates a compromised gut
Abdomcentesis total protein	linear - the higher the level of protein the more likely it is to have a compromised gut. Values are in gms/dL
Outcome	what eventually happened to the horse? - possible values: 1 = lived 2 = died 3 = was euthanized
Surgical lesion?	retrospectively, was the problem (lesion) surgical? - all cases are either operated upon or autopsied so that this value and the lesion type are always known - possible values: 1 = Yes 2 = No
Type of lesion	Attribute 25, 26 and 27 - first number is site of lesion 1 = gastric 2 = sm intestine 3 = lg colon 4 = lg colon and cecum 5 = cecum 6 = transverse colon 7 = retum/descending colon 8 = uterus 9 = bladder 11 = all intestinal sites 00 = none - second number is type 1 = simple 2 = strangulation 3 = inflammation 4 = other - third number is subtype 1 = mechanical 2 = paralytic 0 = n/a - fourth number is specific code 1 = obturation 2 = intrinsic 3 = extrinsic 4 = adynamic 5 = volvulus/torsion 6 = intussuption 7 = thromboembolic 8 = hernia 9 = lipoma/slenic incarceration 10 = displacement 0 = n/a
Cp_data	is pathology data present for this case? 1 = Yes 2 = No - this variable is of no significance since pathology data is not included or collected for these cases

2 Analysis

2.1 Data download and preparation

The following code installs the required packages if required and loads the CSV file from the working directory.

```
#Install and attach required packages
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(rpart)
library(randomForest)

#read csv file from working directory into a data frame df
df <- read_csv("horse.csv")
```

We can take an initial look at the data by viewing the first 10 rows. The column or attribute we will be trying to predict is OUTCOME by using some or all of the other attributes.

```
#view data
head(df)
```

```
## # A tibble: 6 x 28
##   surgery age   hospital_number rectal_temp pulse respiratory_rate
##   <chr>   <chr>           <dbl>       <dbl> <dbl>            <dbl>
## 1 no      adult          530101        38.5    66               28
## 2 yes     adult          534817        39.2    88               20
## 3 no      adult          530334        38.3    40               24
## 4 yes     young         5290409        39.1   164               84
## 5 no      adult          530255        37.3   104               35
## 6 no      adult          528355        NA      NA               NA
## # ... with 22 more variables: temp_of_extremities <chr>,
## #   peripheral_pulse <chr>, mucous_membrane <chr>, capillary_refill_time <chr>,
## #   pain <chr>, peristalsis <chr>, abdominal_distention <chr>,
## #   nasogastric_tube <chr>, nasogastric_reflux <chr>,
## #   nasogastric_reflux_ph <dbl>, rectal_exam_feces <chr>, abdomen <chr>,
## #   packed_cell_volume <dbl>, total_protein <dbl>, abdomo_appearance <chr>,
## #   abdomo_protein <dbl>, outcome <chr>, surgical_lesion <chr>, ...
```

There are 299 rows in the dataset and 28 attributes. The class of the variables is a mix of character and numeric data.

```
dim(df)
```

```
## [1] 299  28
```

```
sapply(df,class)
```

```
##               surgery                   age       hospital_number
##           "character"           "character"             "numeric"
##           rectal_temp                 pulse      respiratory_rate
##             "numeric"             "numeric"             "numeric"
##   temp_of_extremities      peripheral_pulse       mucous_membrane
##           "character"           "character"           "character"
## capillary_refill_time                 pain           peristalsis
##           "character"           "character"           "character"
##  abdominal_distention      nasogastric_tube    nasogastric_reflux
##           "character"           "character"           "character"
## nasogastric_reflux_ph     rectal_exam_feces              abdomen
##             "numeric"           "character"           "character"
##    packed_cell_volume         total_protein     abdomo_appearance
##             "numeric"             "numeric"           "character"
##        abdomo_protein               outcome       surgical_lesion
##             "numeric"           "character"           "character"
##              lesion_1              lesion_2              lesion_3
##             "numeric"             "numeric"             "numeric"
##               cp_data
##           "character"
```

Let us first examine the data for missing values. The website where the data was sourced noted that approximately 30% of the values were absent. We will have to decide how to remove or replace these missing values.

```
#save data frame with number of NAs per column
col_nas <- as.data.frame(colSums(is.na(df)))
col_nas <- col_nas %>%
  mutate(p = row.names(col_nas)) %>%
  setNames(c("parameters", "count_NAs"))

#replace row names with numbers
row.names(col_nas) <- 1:nrow(col_nas)

#order to see where the most NAs are
col_nas <- col_nas %>%
  arrange(desc(parameters))

#view top 20
head(col_nas, 20)
```

```
##    parameters             count_NAs
## 1         246 nasogastric_reflux_ph
## 2         198        abdomo_protein
## 3         165     abdomo_appearance
## 4         118              abdomen
## 5         106    nasogastric_reflux
## 6         104      nasogastric_tube
## 7         102     rectal_exam_feces
## 8          69      peripheral_pulse
```

```
## 9          60           rectal_temp
## 10         58      respiratory_rate
## 11         56   temp_of_extremities
## 12         56  abdominal_distention
## 13         55                  pain
## 14         47       mucous_membrane
## 15         44           peristalsis
## 16         33         total_protein
## 17         32 capillary_refill_time
## 18         29    packed_cell_volume
## 19         24                 pulse
## 20          0               surgery
```

We have listed the attributes with the most missing values, from the largest to smallest. As seen in the table above there are in some columns, more than 80% of the values missing. In most cases we will replace them with the mean of the non NA values for each variable.

Let us first examine the data for missing values. The website where the data was sourced noted that approximately 30% of the values were absent. We will have to decide how to remove or replace these missing values.

First, we will save a copy of the dataframe in case we need to go back to our original unchanged data. We remove a few columns that are not relevant, using the attribute description file as our guide.

We will changed all the columns to numeric so that they can be readily used as predictors. The code below addresses each column at a time, replacing the character entries with their replacement numeric value from the text files. we then delete the original version of the column. This was only done for the character columns.

```
#remove columns / parameters that will be of no use to us
#Change all columns to numeric values
#Replace NAs with appropriate values

#save to new dataframe
df2 <- df %>%
#cp_data indicates if pathology data is present (yes/no) / not relevant because no pathology data is in
#remove column
  select(-cp_data) %>%
#replace lesion 1, 2 and 3 with one column, there were no NAs
  mutate(lesion = ifelse(lesion_1 >0 | lesion_2 > 0 | lesion_3 > 0, 1 ,0)) %>%
  select(-lesion_1, -lesion_2, -lesion_3) %>%
#Changed as per column description file, no NAs
  mutate(surgery2 = ifelse(surgery == "yes" ,1, 0)) %>%
  select(-surgery) %>%
#Changed as per column description file, no NAs
  mutate(age2 = ifelse(age == "adult" ,1, 2)) %>%
  select(-age) %>%
#Changed as per column description file, NAs replaced below
  mutate(temp_extrem = ifelse(temp_of_extremities == "normal" ,1 ,
                              ifelse(temp_of_extremities == "warm" ,2,
                                     ifelse(temp_of_extremities == "cool" ,3,
                                            ifelse(temp_of_extremities == "cold" ,4,0))))) %>%
  select(-temp_of_extremities) %>%
#Changed as per column description file, NAs replaced below
  mutate(periph_pulse = ifelse(peripheral_pulse == "normal" ,1 ,
```

```
                              ifelse(peripheral_pulse == "increased" ,2,
                                     ifelse(peripheral_pulse == "reduced" ,3,
                                            ifelse(peripheral_pulse == "absent" ,4,0)))) %>%
  select(-peripheral_pulse) %>%
#Changed as per column description file, NAs replaced below
  mutate(mucous_membrane2 = ifelse(mucous_membrane == "normal_pink" ,1 ,
                                ifelse(mucous_membrane == "bright_pink" ,2,
                                       ifelse(mucous_membrane == "pale_pink" ,3,
                                              ifelse(mucous_membrane == "pale_cyanotic" ,4,
                                                     ifelse(mucous_membrane == "bright_red" ,5,
                                                            ifelse(mucous_membrane == "dark_cyanotic" ,6
  select(-mucous_membrane) %>%
#Changed as per column description file, NAs replaced below
  mutate(cap_refill_time = ifelse(capillary_refill_time == "less_3_sec" ,1, 2)) %>%
  select(-capillary_refill_time) %>%
#Changed as per column description file, NAs replaced below
  mutate(pain2 = ifelse(pain == "alert" ,1 ,
                                    ifelse(pain == "depressed" ,2,
                                           ifelse(pain == "mild_pain" ,3,
                                                  ifelse(pain == "severe_pain" ,4,
                                                         ifelse(pain == "extreme_pain" ,5, 0)))))) %>%
  select(-pain) %>%
#Changed as per column description file, NAs replaced below
  mutate(peristalsis2 = ifelse(peristalsis == "hypermotile" ,1 ,
                                ifelse(peristalsis == "normal" ,2,
                                       ifelse(peristalsis == "hypomotile" ,3,
                                              ifelse(peristalsis == "absent" ,4,0))))) %>%
  select(-peristalsis) %>%
#Changed as per column description file, NAs replaced below
  mutate(abdom_distention = ifelse(abdominal_distention == "none" ,1 ,
                                ifelse(abdominal_distention == "slight" ,2,
                                       ifelse(abdominal_distention == "moderate" ,3,
                                              ifelse(abdominal_distention == "severe" ,4,0))))) %>%
  select(-abdominal_distention) %>%
#Changed as per column description file, NAs replaced below
  mutate(nasogast_tube = ifelse(nasogastric_tube == "none" ,1 ,
                                   ifelse(nasogastric_tube == "slight" ,2,
                                          ifelse(nasogastric_tube == "significant" ,3,0)))) %>%
  select(-nasogastric_tube) %>%
#Changed as per column description file, NAs replaced below
  mutate(nasogast_reflux = ifelse(nasogastric_reflux == "none" ,1 ,
                                ifelse(nasogastric_reflux == "less_1_liter" ,2,
                                       ifelse(nasogastric_reflux == "more_1_liter" ,3,0)))) %>%
  select(-nasogastric_reflux) %>%
#Changed as per column description file, NAs replaced below
  mutate(rectal_exam_feces2 = ifelse(rectal_exam_feces == "normal" ,1 ,
                                ifelse(rectal_exam_feces == "increased" ,2,
                                       ifelse(rectal_exam_feces == "decreased" ,3,
                                              ifelse(rectal_exam_feces == "absent" ,4,0))))) %>%
  select(-rectal_exam_feces) %>%
#Changed as per column description file, NAs replaced below
  mutate(abdomen2 = ifelse(abdomen == "normal" ,1 ,
                           ifelse(abdomen == "other" ,2,
```

```
                                ifelse(abdomen == "firm" ,3,
                                       ifelse(abdomen == "distend_small" ,4,
                                              ifelse(abdomen == "distend_large" ,5, 0)))))) %>%
  select(-abdomen) %>%
#Changed as per column description file, NAs replaced below
  mutate(abdomo_appear = ifelse(abdomo_appearance == "clear" ,1 ,
                                  ifelse(abdomo_appearance == "cloudy" ,2,
                                         ifelse(abdomo_appearance == "serosanguious" ,3,0)))) %>%
  select(-abdomo_appearance) %>%
#Changed to include only 2 outcomes 1 = lived or 0 = euthanised or died
  mutate(outcome2 = ifelse(outcome == "lived" ,1 ,
                                ifelse(outcome == "died" ,0,
                                       ifelse(outcome == "euthanized" ,0,0)))) %>%
  select(-outcome) %>%
  mutate(surg_lesion = ifelse(surgical_lesion == "yes" ,1, 0)) %>%
  select(-surgical_lesion)

dim(df2)
```

```
## [1] 299  25
```

```
sapply(df2,class)
```

```
##       hospital_number           rectal_temp                 pulse
##             "numeric"             "numeric"             "numeric"
##      respiratory_rate nasogastric_reflux_ph    packed_cell_volume
##             "numeric"             "numeric"             "numeric"
##         total_protein        abdomo_protein                lesion
##             "numeric"             "numeric"             "numeric"
##              surgery2                  age2           temp_extrem
##             "numeric"             "numeric"             "numeric"
##          periph_pulse      mucous_membrane2       cap_refill_time
##             "numeric"             "numeric"             "numeric"
##                 pain2          peristalsis2      abdom_distention
##             "numeric"             "numeric"             "numeric"
##         nasogast_tube       nasogast_reflux    rectal_exam_feces2
##             "numeric"             "numeric"             "numeric"
##              abdomen2         abdomo_appear              outcome2
##             "numeric"             "numeric"             "numeric"
##           surg_lesion
##             "numeric"
```

We are now down to 25 attributes having deleted 1 and consolidating 3 into 1. All the character attributes have been replaced with numeric ones, as shown in the table above.

The "Outcome" attribute which we will be trying to predict has been changed to only indicate 2 separate outcomes, "lived" == 1, and "dies" == 0. The "euthanized" option was changed to "died" as it's eventual outcome is the same.

Now we can replace all the missing entries column-by-column, the ones with the NAs, with the means of the rest of the column.

```
#replace NAs/Os from process above

#Temp of extremities, replace NAs with mean
  df2$temp_extrem[is.na(df2$temp_extrem)] <- mean(df2$temp_extrem,na.rm=TRUE)
#Peripheral pulse, replace NAs with mean
  df2$periph_pulse[is.na(df2$periph_pulse)] <- mean(df2$periph_pulse,na.rm=TRUE)
#Mucous membranes, replace NAs with mean
  df2$mucous_membrane2[is.na(df2$mucous_membrane2)] <- mean(df2$mucous_membrane2,na.rm=TRUE)
#Capillary refill time, replace NAs with mean
  df2$cap_refill_time[is.na(df2$cap_refill_time)] <- mean(df2$cap_refill_time,na.rm=TRUE)
#Pain level, replace NAs with mean
  df2$pain2[is.na(df2$pain2)] <- mean(df2$pain2,na.rm=TRUE)
#Peristalsis activity in gut, replace NAs with mean
  df2$peristalsis2[is.na(df2$peristalsis2)] <- mean(df2$peristalsis2,na.rm=TRUE)
#Abdominal distension, replace NAs with mean
  df2$abdom_distention[is.na(df2$abdom_distention)] <- mean(df2$abdom_distention,na.rm=TRUE)
#Gas coming out of nasogastric tube, replace NAs with mean
  df2$nasogast_tube[is.na(df2$nasogast_tube)] <- mean(df2$nasogast_tube,na.rm=TRUE)
#Nasogastric reflux, replace NAs with mean
  df2$nasogast_reflux[is.na(df2$nasogast_reflux)] <- mean(df2$nasogast_reflux,na.rm=TRUE)
#rectal examination - feces, replace NAs with mean
  df2$rectal_exam_feces2[is.na(df2$rectal_exam_feces2)] <- mean(df2$rectal_exam_feces2,na.rm=TRUE)
#abdomen, replace NAs with mean
  df2$abdomen2[is.na(df2$abdomen2)] <- mean(df2$abdomen2,na.rm=TRUE)
#abdomen, replace NAs with mean
  df2$abdomo_appear[is.na(df2$abdomo_appear)] <- mean(df2$abdomo_appear,na.rm=TRUE)
#PH of nasogastric reflux, replace NAs with mean
  df2$nasogastric_reflux_ph[is.na(df2$nasogastric_reflux_ph)] <- mean(df2$nasogastric_reflux_ph,na.rm=TR
#abdomcentesis total protein, replace NAs with mean
  df2$abdomo_protein[is.na(df2$abdomo_protein)] <- mean(df2$abdomo_protein,na.rm=TRUE)
#rectal temperature, replace NAs with mean
  df2$rectal_temp[is.na(df2$rectal_temp)] <- mean(df2$rectal_temp,na.rm=TRUE)
#respiratory rate, replace NAs with mean
  df2$respiratory_rate[is.na(df2$respiratory_rate)] <- mean(df2$respiratory_rate,na.rm=TRUE)
#total protein, replace NAs with mean
  df2$total_protein[is.na(df2$total_protein)] <- mean(df2$total_protein,na.rm=TRUE)
#packed cell volume, replace NAs with mean
  df2$packed_cell_volume[is.na(df2$packed_cell_volume)] <- mean(df2$packed_cell_volume,na.rm=TRUE)
#Pulse, replace NAs with mean
  df2$pulse[is.na(df2$pulse)] <- mean(df2$pulse,na.rm=TRUE)
```

We can now check again to see if all the NAs have been resolved. With a descending sort we see that all data is complete.

```
#check again for NAs
#save data frame with number of NAs per column
col_nas2 <- as.data.frame(colSums(is.na(df2)))
col_nas2 <- col_nas2 %>%
  mutate(p = row.names(col_nas2)) %>%
  setNames(c("parameters", "count_NAs"))

#replace row names with numbers
row.names(col_nas2) <- 1:nrow(col_nas2)
```

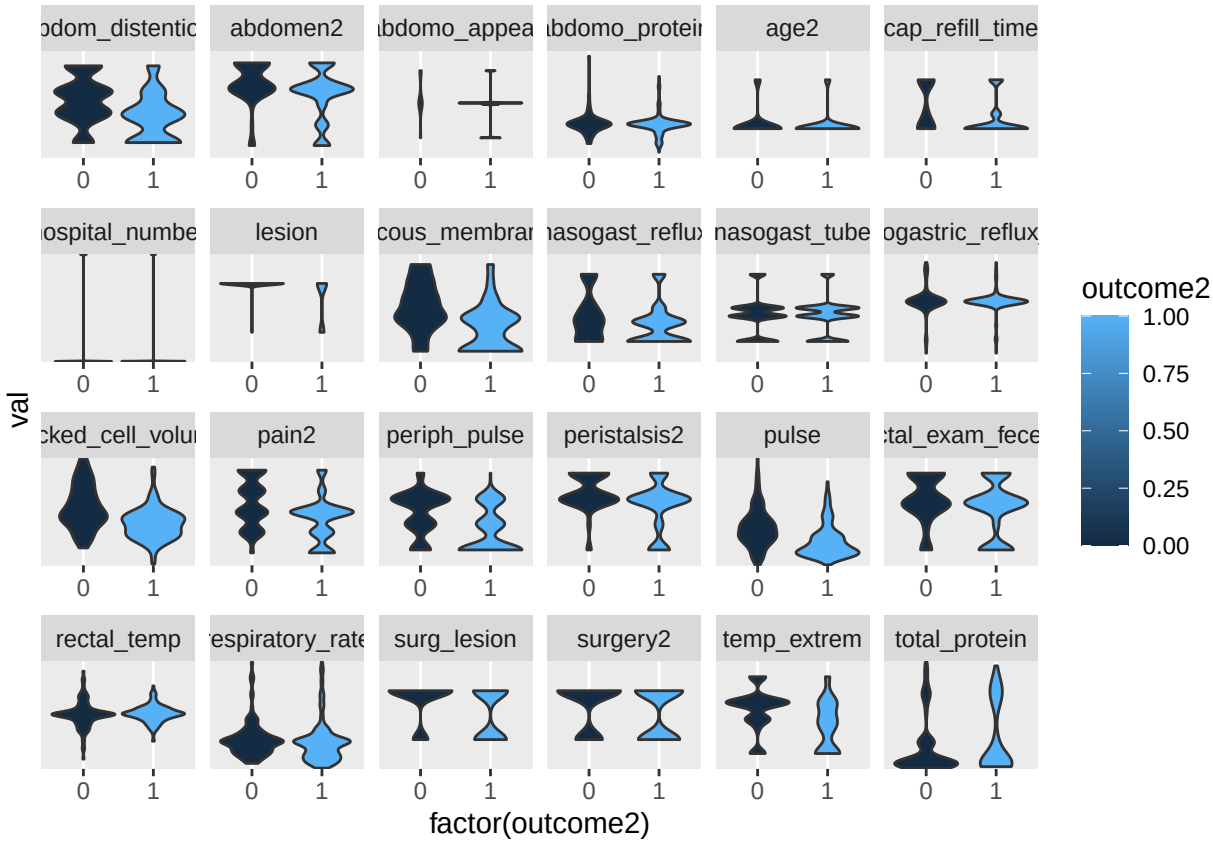
```
#order to see where the most NAs are
col_nas2 <- col_nas2 %>%
  arrange(desc(parameters))

#view top 20
head(col_nas2, 20)
```

```
##    parameters             count_NAs
## 1           0       hospital_number
## 2           0           rectal_temp
## 3           0                 pulse
## 4           0      respiratory_rate
## 5           0 nasogastric_reflux_ph
## 6           0    packed_cell_volume
## 7           0         total_protein
## 8           0        abdomo_protein
## 9           0                lesion
## 10          0              surgery2
## 11          0                  age2
## 12          0           temp_extrem
## 13          0          periph_pulse
## 14          0      mucous_membrane2
## 15          0       cap_refill_time
## 16          0                 pain2
## 17          0          peristalsis2
## 18          0      abdom_distention
## 19          0         nasogast_tube
## 20          0       nasogast_reflux
```

We should have a look at the various attributes to see if there are any that we can remove that wont help us with our predictions and will just extend processing time. Different types of plots were looked at, but the violin plot seems to highlight the attributes that will be most useful to assist with the predictions. We will be looking for those attributes where there are marked differences in the distributions of "died" and "lived" / 0 and 1.

```
#violin plot - use this one
df2 %>% gather(params, val, -outcome2) %>%
  ggplot(aes(factor(outcome2), val, fill = outcome2)) +
  geom_violin() +
  scale_y_discrete() +
  facet_wrap(~params, scales = "free", ncol = 6)
```



We will start with removing the "Hospital number" attribute as it does not seems as though it would be helpful to make a clear prediction. There is no clear distinction that can be made between the 2 desired outcomes from what we see in the data distribution.
Through observation and testing "Age", "Nasogastric_reflux_ph" and "abdomo_protein" were also removed. We now have only 20 attributes to use in our predictions.

```
#remove parameters that wont be useful
df2 <- df2 %>%
  select(-hospital_number, -age2, -nasogastric_reflux_ph, -abdomo_protein)
```

2.2 Training and test set preparation

Our dataset can now be partitioned into a training and test set. A 50% split was decided on to ensure an even split of all values into both sets. Train_set and Test_set were created from the cleaned copy of the dataframe. A validation set was not separated from the test set.

```
#create an index and data partition from the full dataframe to give us a training and a test dataset
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(df2$outcome2, times = 1, p = 0.5, list = FALSE)
test_set <- df2[test_index, ]
train_set <- df2[-test_index, ]
```

2.3 Training / modelling methods

In this section we will try to train different machine learning models on our training set and then check them against our testing set. We'll save our accuracy estimates for each method into a table so we can compare our modelling at the end.

Each method follows a similar pattern:

```
  * train our model on the training set
  * generate a set of predictions on our test set
  * validate the predictions against the real outcome
  * save results to table.
```

2.3.1 Random guess

Just to get a starting point and a feel for a random sampling estimate we will make a guess for each row in the test set and determine the accuracy, which we would expect to be around 50%.

```
#Guess an outcome and test the overall accuracy against the test set
set.seed(1, sample.kind="Rounding")
y_hat_guess <- sample(c(0, 1), length(test_index), replace = TRUE)
calc_acc <- mean(y_hat_guess == test_set$outcome2)
calc_acc
```

```
## [1] 0.5466667
```

```
accuracy_results <- tibble(method = "Guess", Accur = calc_acc, Sens = NA, Spec = NA)
```

2.3.2 Linear regression

Our first real test will be using linear regression with R's lm function.

```
#train
fit_lr <- train_set %>%
  lm(outcome2 ~ ., data = .)
#predict
p_hat_lr <- predict(fit_lr, newdata = test_set)
#change predictions from prob to discrete and factorise
y_hat_lr <- ifelse(p_hat_lr > 0.5, 1, 0) %>% factor()
#test
cm <- confusionMatrix(data = y_hat_lr, reference = factor(test_set$outcome2))
cm$overall["Accuracy"]
```

```
## Accuracy
##     0.72
```

```
cm$byClass[c("Sensitivity","Specificity", "Prevalence")]
```

```
## Sensitivity Specificity  Prevalence
##   0.6166667   0.7888889   0.4000000
```

```
#save result
accuracy_results <- accuracy_results %>% add_row(tibble_row(method = "Linear regression",
                                                            Accur = cm$overall["Accuracy"],
                                                            Sens = cm$byClass["Sensitivity"],
                                                            Spec = cm$byClass["Specificity"]))
```

We get an overall accuracy of 72%, much better than just guessing, but our sensitivity (true positive rate "TPR") is only 62%. The specificity, or true negative rate "TNR" is 79%.

2.3.3 Logistic regression

Next we'll apply Rs glm function to fit a logistic regression model.

```
#train
glm_fit <- train_set %>%
  glm(outcome2 ~ ., data = ., family = "binomial")
#predict
p_hat_glm <- predict(glm_fit, newdata = test_set, type = "response")
#change predictions from prob to discrete and factorise
y_hat_glm <- ifelse(p_hat_glm >0.5, 1, 0) %>% factor()
#test
cm <- confusionMatrix(data = y_hat_glm, reference = factor(test_set$outcome2))
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.7266667
```

```
cm$byClass[c("Sensitivity","Specificity", "Prevalence")]
```

```
## Sensitivity Specificity  Prevalence
##   0.6166667   0.8000000   0.4000000
```

```
#save result
accuracy_results <- accuracy_results %>% add_row(tibble_row(method = "Logistic regression",
                                                            Accur = cm$overall["Accuracy"],
                                                            Sens = cm$byClass["Sensitivity"],
                                                            Spec = cm$byClass["Specificity"]))
```

A negligible improvement on linear regression, the accuracy improves to 73%, sensitivity remains 62% and specificity increases to 80%.

2.3.4 K Nearest Neighbour

We do not know at this point which k parameter, the number of neighbours, would give the lowest error so we run a sequence of values, odd numbers from 1 to 25 in our training.

```
#train
train_knn <- train(outcome2 ~ ., data = train_set,
                   tunegrid = data.frame(k = seq(1,25,2)))
train_knn$bestTune
```

```
##   mtry
## 1    2
```

```
#predict
p_hat_knn <- predict(train_knn, test_set, type = "raw")
#change predictions from prob to discrete and factorise
y_hat_knn <- ifelse(p_hat_knn > 0.5,1 ,0) %>% factor()
#test
cm <- confusionMatrix(data = y_hat_knn, reference = factor(test_set$outcome2))
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.7866667
```

```
cm$byClass[c("Sensitivity","Specificity", "Prevalence")]
```

```
## Sensitivity Specificity  Prevalence
##   0.6666667   0.8666667   0.4000000
```

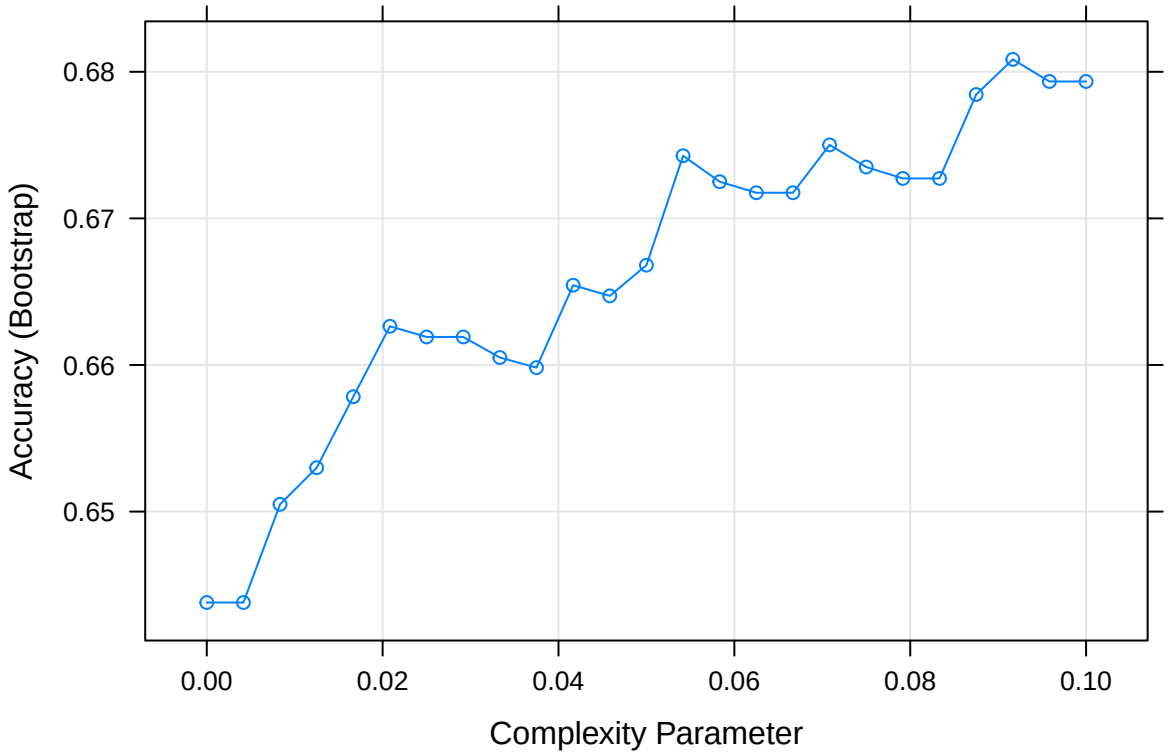
```
#save result
accuracy_results <- accuracy_results %>% add_row(tibble_row(method = "KNN",
                                                             Accur = cm$overall["Accuracy"],
                                                             Sens = cm$byClass["Sensitivity"],
                                                             Spec = cm$byClass["Specificity"]))
```

We have managed to improve our accuracy to 79% as well as our sensitivity to 67% with an increase in specificity to 87%.

2.3.5 Classification tree

With our decision tree we also try to select the best tuning parameter by running a sequence of values.

```
set.seed(1, sample.kind="Rounding")
#train
train_rpart <- train(factor(outcome2) ~ .,
                     method = "rpart",
                     tuneGrid = data.frame(cp = seq(0.0, 0.1, len = 25)),
                     data = train_set)
plot(train_rpart)
```




```
#predict
y_hat_rpart <- predict(train_rpart, test_set)
#test
cm <- confusionMatrix(data = y_hat_rpart, reference = factor(test_set$outcome2))
cm$overall["Accuracy"]
```

```
## Accuracy
##     0.68
```

```
cm$byClass[c("Sensitivity","Specificity", "Prevalence")]
```

```
## Sensitivity Specificity  Prevalence
##   0.3666667   0.8888889   0.4000000
```

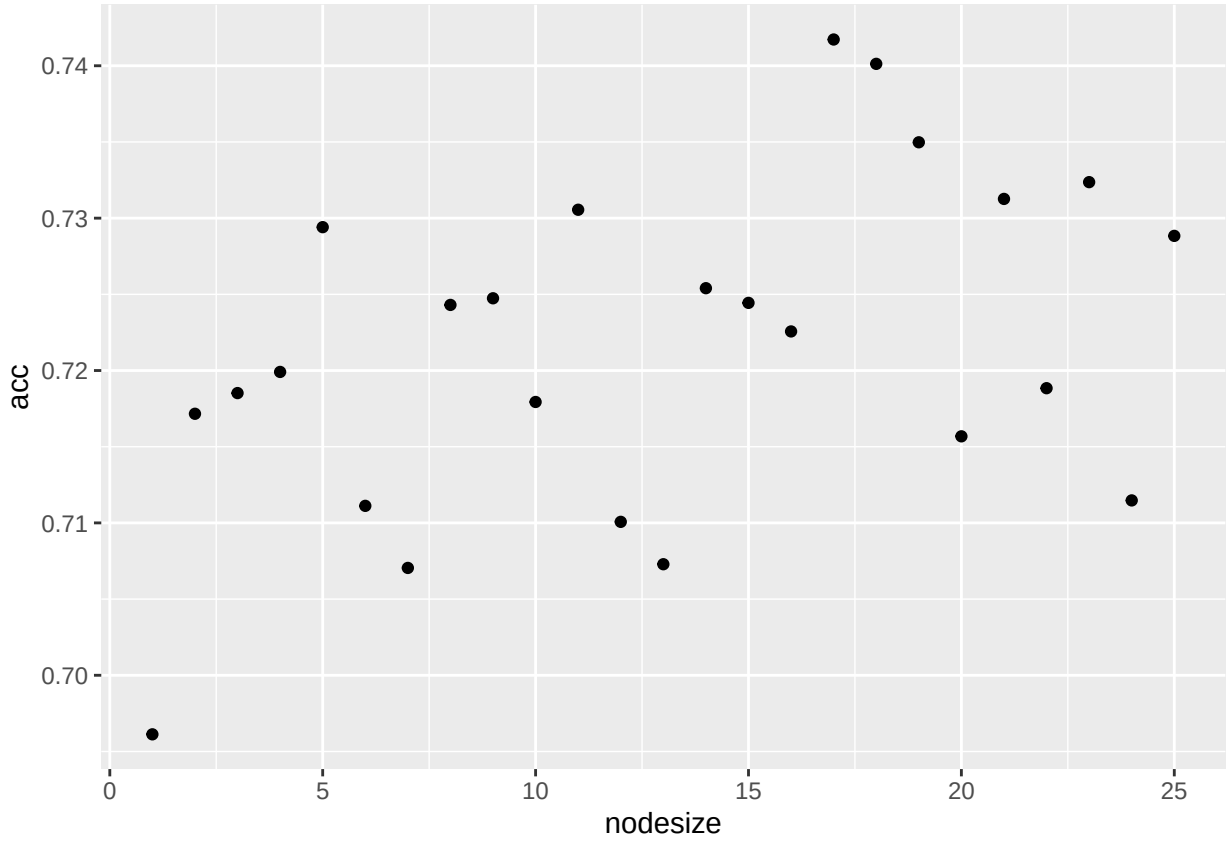
```
#save result
accuracy_results <- accuracy_results %>% add_row(tibble_row(method = "Classification tree",
                                                             Accur = cm$overall["Accuracy"],
                                                             Sens = cm$byClass["Sensitivity"],
                                                             Spec = cm$byClass["Specificity"]))
```

Accuracy drops with this model to 68% as well as the sensitivity to 37% but specificity is up to 89%. With our many predictors we suffer from the so-called *curse of dimesionality* making our neighborhood too large to deliver reasonable accuracy.

2.3.6 Random forest

We will now try to improve on the decision tree model by averaging multiple decision trees in a random forest model. First we need to pick the optimal node size as a tuning parameter running through a sequence from 1 to 25.

```
#determine best tuning parameter nodesize
set.seed(1, sample.kind="Rounding")
nodesize <- seq(1 ,25, 1)
acc <- sapply(nodesize, function(ns){
  train(factor(outcome2) ~ ., method = "rf", data = train_set,
        tuneGrid = data.frame(mtry = 2),
        nodesize = ns)$results$Accuracy
})
qplot(nodesize, acc)
```




```
which.max(acc)
```

```
## [1] 17
```

The best option seems to be a nodesize of 17

```
#use that nodesize in the RF prediction
#train using nodesize
fit_rf <- randomForest(outcome2 ~ ., data = train_set, nodesize = which.max(acc))
```

```
#predict
p_hat_rf <- predict(fit_rf, newdata = test_set)
#change predictions from prob to discrete and factorise
y_hat_rf <- ifelse(p_hat_rf > 0.5,1 ,0) %>% factor()
#test
cm <- confusionMatrix(data = y_hat_rf, reference = factor(test_set$outcome2))
cm$overall["Accuracy"]
```

```
## Accuracy
##     0.76
```

```
cm$byClass[c("Sensitivity","Specificity", "Prevalence")]
```

```
## Sensitivity Specificity  Prevalence
##   0.6166667   0.8555556   0.4000000
```

```
#save result
accuracy_results <- accuracy_results %>% add_row(tibble_row(method = "Random forest",
                                                             Accur = cm$overall["Accuracy"],
                                                             Sens = cm$byClass["Sensitivity"],
                                                             Spec = cm$byClass["Specificity"]))
```

We've improved our accuracy to 76%, with a sensitivity of 62% and a specificity of 86%.

2.3.7 Ensemble

As a last measure we will try and get the best from two of the better methods, the KNN model and the random forest model. We will create a simple mean of their probabilities and test against our test set.

```
#ensemble method
#create dataframe with KNN probs
p_hat_ens <- as.data.frame(p_hat_knn)
p_hat_ens["p_hat_rf"] <- p_hat_rf #add random forest probs
p_hat_ens["mean"] <- (p_hat_ens$p_hat_knn + p_hat_ens$p_hat_rf)/2 #mean out the probs
p_hat_ens$y_hat_ens <- ifelse(p_hat_ens$mean > 0.5,1 ,0) # convert to 1 / 0
y_hat_ens <- factor(p_hat_ens$y_hat_ens)  #factorise
#test
cm <- confusionMatrix(data = y_hat_ens, reference = factor(test_set$outcome2))
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.7733333
```

```
cm$byClass[c("Sensitivity","Specificity", "Prevalence")]
```

```
## Sensitivity Specificity  Prevalence
##   0.6500000   0.8555556   0.4000000
```

```
#save result
accuracy_results <- accuracy_results %>% add_row(tibble_row(method = "Ensemble",
                                                            Accur = cm$overall["Accuracy"],
                                                            Sens = cm$byClass["Sensitivity"],
                                                            Spec = cm$byClass["Specificity"]))
```

Our accuracy has now changed to 77% with a sensitivity of 65% and a specificity of 86%.

3 Results

Of all the methods used, the K Nearest Neighbour seems to deliver the best of all 3 measures in the confusion matrix. We have an accuracy of 79%, sensitivity of 67% and specificity of 87%. The other methods, and even the ensemble method do not seem to be able to improve on this.

```
accuracy_results
```

```
## # A tibble: 7 x 4
##   method              Accur   Sens   Spec
##   <chr>               <dbl>  <dbl>  <dbl>
## 1 Guess               0.547 NA     NA
## 2 Linear regression   0.72   0.617  0.789
## 3 Logistic regression 0.727  0.617  0.8
## 4 KNN                 0.787  0.667  0.867
## 5 Classification tree 0.68   0.367  0.889
## 6 Random forest       0.76   0.617  0.856
## 7 Ensemble            0.773  0.65   0.856
```

4 Conclusion

The model we have developed should be able to predict survival or death of a horse with colic, using the various medical parameters, to less than an 80% accuracy level.

Because we have very little data from the downloaded set, when compared to other massive datasets, the KNN method we used above seems to deliver the best result. We have considered cross validation to improve our models by using the tunegrid parameter. This has maximised our accuracy slightly over the other methods.

Future work should include more granularity on the selection of medical parameters. Many of them could possibly be excluded and deliver a better result because they are adding meaningless complexity. We could also try clustering the predictors to gain more meaningful insights especially with the classification tree method owing to the fact that most of the predictors are discrete.