STAT40750 Data Analysis Project

Matthew Gannon 16343261

4/9/2020

Title

An investigation of supervised classification methods on the backpain dataset.

Abstract

With this report, I will be discussing various supervised classification methods. Those used in the analysis are random forests, class trees, bagging, roc curves and multinominal logistic regression respectively. I will discuss each methods merits and how they can best used effectively. I will also be discussing how to optimise the ROC curve threshold, in order to maximise classification accuracy.

Introduction

Firstly we must load in our dataset.

```
getwd()
```

```
## [1] "C:/Users/Jellyflabman/Documents"
```

```
load( "data_project_backpain.RData" )
```

Using str(dat) and print(codeVariables), we can see a dataset of 32 variables and 380 observations of those variables on the lower back pain of patients. 13 are numerical variables and 19 are categorical. The variable we are most interested in, is that of PainDiagnosis. Our aim is to create methods which can successfully predict by training, which patients suffer from Nociceptive pain (Pain created by damage to non neural tissue) and neuropathic pain (pain caused caused by dysfunction in the nervous system)

Methods

As a variety of our classification methods necessitate the use of categorical variables, we must not a wide variety of continuous variables, for example the Component summary scores. There is a variety of ways to deal with this. We could remove the data all together, which I think would be wasting valuable predictive data. Or we could split the numbers into high and low categories using the mean. While this may seem like a good solution it should be noted that the less categories presented to the prediction the less accurate it will be. This will also be skewed alot more by outliers as the mean is highly succeptable to change from them.

The method I will be using is by seperating the variables into groups of equal intervals in terms of observations. So first I will use the summary function to discover the quantiles and subsequently use the cut function to separate them.

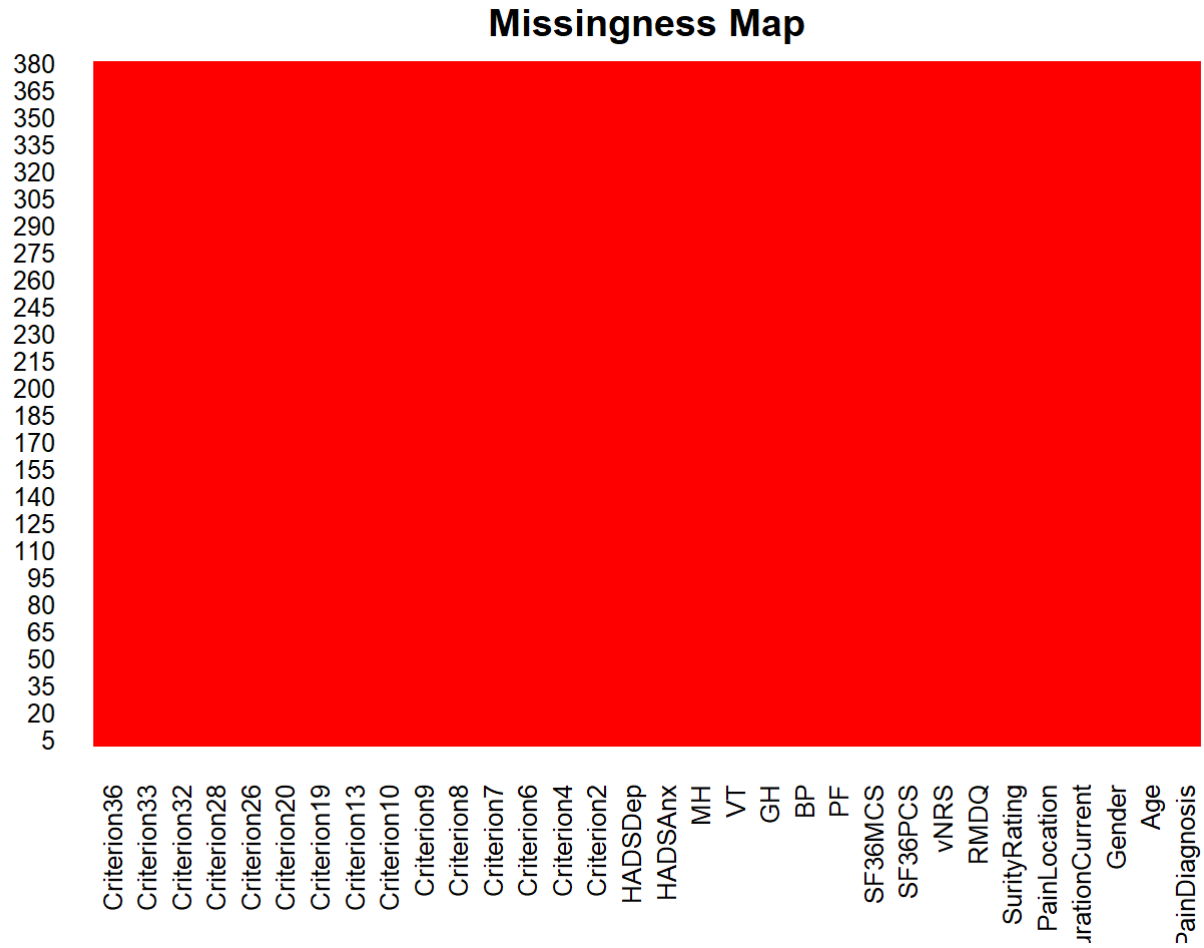
summary(dat)

```
dat$Age <- cut(dat$Age, breaks = c(0,33,53,84), labels = c("Low","Mod","High"), right = FALSE
)
dat$SurityRating <- cut(dat$SurityRating, breaks = c(2,8,9,11), labels = c("Low","Mod","High"
), right = FALSE)
dat$RMDQ <- cut(dat$RMDQ, breaks = c(-1,6,15,25), labels = c("Low","Mod","High"), right = FAL
SE)
dat$vNRS <- cut(dat$vNRS, breaks = c(-1,4,7,11), labels = c("Low","Mod","High"), right = FALS
E)
dat$SF36PCS  <- cut(dat$SF36PCS, breaks = c(13,27.2,41.12,59), labels = c("Low","Mod","High"
), right = FALSE)
dat$SF36MCS <- cut(dat$SF36MCS, breaks = c(11,33,52.7,71), labels = c("Low","Mod","High"), ri
ght = FALSE)
dat$PF <- cut(dat$PF, breaks = c(14,33.9,43,58), labels = c("Low","Mod","High"), right = FALS
E)
dat$BP <- cut(dat$BP, breaks = c(19,24.9,37.2,63), labels = c("Low","Mod","High"), right = FA
LSE)
dat$GH <- cut(dat$GH, breaks = c(16,33.65,50.6,64), labels = c("Low","Mod","High"), right = F
ALSE)
dat$VT <- cut(dat$VT, breaks = c(20,33.4,49,71), labels = c("Low","Mod","High"), right = FALS
E)
dat$MH  <- cut(dat$MH, breaks = c(10,35.9,52.8,65), labels = c("Low","Mod","High"), right = F
ALSE)
dat$HADSAnx  <- cut(dat$HADSAnx, breaks = c(-1,4,11,22), labels = c("Low","Mod","High"), righ
t = FALSE)
dat$HADSDep <- cut(dat$HADSDep, breaks = c(-1,3,10,21), labels = c("Low","Mod","High"), right
= FALSE)
```

Now if we have done this correctly, we should have no missing data, so let's check this.

```
library(mlbench)
library(Amelia)
```

```
missmap(dat, col=c("blue", "red"), legend=FALSE)
```



No missing data in our set. But note, this switch to a purely categorical approach will result in a lose of accuracy to our models that having continuous data can provide.

Now using our new dataset, let's compare classification using training validation and test sets.

I will be comparing the performatives of 3 seperate classification methods: A classification tree , a multinominal regression and lastly a random forest.

Before I begin, I will define the 3 methods being used in this comparative analysis.

The first of these is a classification tree of the data, with PainDiagnosis as our target variable. In this case, we are using our 31 other variables to predict whether one is Nociceptive or Neuropathic. The top of our decision tree, otherwise known as the root node is decided by calculating the Gini impurity of the predictor variables. The variable with the lowest Gini impurity is put at the opening root node, followed by the lowest Gini impurity calculated of the remaining variables, until we get to the final leaf nodes.

The second of these is a random forest of data, this involves creating whats known as a "bootstrapped" data set, by randomly selecting samples from the data. Here we are allowed pick the same sample more than once. Once this is done, we create a decision tree using the bootstrapped dataset and using a random subset of the variables at each node. We then repeat these two steps to create lots of tress (hence the name random forest). This ability to collect permutations tends to result in a much higher accuracy than class trees. The reason for our switch to purely categorical data is that random forests can not be made of numeric data.

The third of these is a multinominal logistic regression. This is the case where by we test to see if a variables effect on the target response variable is signifacantly different from zero. This method has the ability to provide probabilities and classify new samples using both continuous and discrete measurements as well as categorical ones. With logistic regression, the line is fitted using the maximum likelihood. In short, you pick the probability of observing a Nociceptive patient and use that to calculate the likelihood of observing a Neuropathic patient. Then multiply these likelihoods together and shift the line of this data until selecting the line of maximum likelihood.

Now to begin of cross validation we need to create a training dataset for our classification method by seperating a section of our data. Then we need to create a seperate validation and test set. In this case we should have none of our data overlapping as we are checking how the method works on data it hasn't included in it's training. This is so we can fully see how our model would work, had we uncovered an entirely fresh set of data.

```
D <- nrow(dat)
keep <- sample(1:D, 280)
test <- setdiff(1:D, keep)
dat_train <- dat[keep,]
dat_test <- dat[test,]
E<-nrow(dat_train)
keep2<- sample(1:E, 180)
val<- setdiff(1:E, keep2)
dat_train<-dat[keep2,]
dat_val<- dat[val,]
```

Results

So now we have 3 separate sets for training, validation and test at 180, 100 and 100 observations respectively. Now let's use the random forest package to fit our random forests.

```
library(randomForest)
```

```
fit.rf <- randomForest(PainDiagnosis~., data=dat_train)
```

Secondly we will fit our prediction model of our random forest, then create a table of the information in order to calculate the accuracy of this model. This methodology will be followed for our other models so keep this in mind.

```
pred_rf <- predict(fit.rf, type = "class", newdata = dat_val)
tab_rf <- table(dat_val$PainDiagnosis, pred_rf)
accuracy_rf <- sum(diag(tab_rf))/sum(tab_rf)
accuracy_rf
```

```
## [1] 0.96
```

Now we will use the nnet package in order to fit a multinominal regression and follow the same steps.

```
library(nnet)
```

```
fit.reg <- multinom(PainDiagnosis~., data=dat_train)
```

```
## # weights:  54 (53 variable)
## initial  value 124.766493
## iter  10 value 5.980925
## iter  20 value 0.053934
## iter  30 value 0.000115
## iter  30 value 0.000066
## iter  30 value 0.000066
## final  value 0.000066
## converged
```

```
pred_reg <- predict(fit.reg, type = "class", newdata = dat_val)
tab_reg <- table(dat_val$PainDiagnosis, pred_reg)
accuracy_reg <- sum(diag(tab_reg))/sum(tab_reg)
accuracy_reg
```

```
## [1] 0.81
```

And subseuently the same with class trees, as we have seen in the past.

```
library(rpart)
```

```
fit.ct <- rpart(PainDiagnosis~., data=dat_train, method = "class")
pred_ct <- predict(fit.ct, type = "class", newdata = dat_val)
tab_ct <- table(dat_val$PainDiagnosis, pred_ct)
accuracy_ct <- sum(diag(tab_ct))/sum(tab_ct)
accuracy_ct
```

```
## [1] 0.86
```

Here we see our random forest is clearly the most accurate, with a 10 percent higher chance of correctly predicting the data than the class trees and a 15 percent higher chance than the regression model.

Now let's test our test set to see if it matches our validation set.

```
predTest_rf <- predict(fit.rf, type = "class", newdata = dat_test)
tabTest_rf <- table(dat_test$PainDiagnosis, predTest_rf)
acc_rf <- sum(diag(tabTest_rf))/sum(tabTest_rf)
acc_rf
```

```
## [1] 0.97
```

```
predTest_reg <- predict(fit.reg, type = "class", newdata = dat_test)
tabTest_reg <- table(dat_test$PainDiagnosis, predTest_reg)
acc_reg <- sum(diag(tabTest_reg))/sum(tabTest_reg)
acc_reg
```

```
## [1] 0.91
```

```
predTest_ct <- predict(fit.ct, type = "class", newdata = dat_test)
tabTest_ct <- table(dat_test$PainDiagnosis, predTest_ct)
acc_ct <- sum(diag(tabTest_ct))/sum(tabTest_ct)
acc_ct
```

```
## [1] 0.86
```

Discussion

Here we see the same result, where the random forest has is the most accurate. Class trees have been historically known for their inaccuracy, in short they work great with the data used to create them, but they are not flexible when it comes to new samples.

Random Forests combine decision trees, which results in improvements to accurate prediction. So that begs the question, is there a way to make the classification method more stable so as to content with the accuracy of a random forest. Well one option is that of bagging.

Bagging is effectively bootstrapping the data plus using the agregate of observations. This in turn lowers variance and avoids overfitting.

Let's use the adabag package to create a bagging model and compare it to our random forest observations.

```
library(adabag)
```

```
fit.bag <- bagging(PainDiagnosis~., data = dat_train)
pred_bag <- predict(fit.bag, type = "class", newdata = dat_val)
tab_bag <- pred_bag$confusion
accuracy_bag <- 1 - sum(diag(tab_bag))/sum(tab_bag)
accuracy_bag
```

```
## [1] 0.92
```

As we can see this is far closer than either our regression or that of our classification tree. Now lets compare the test data.

```
predtest_bag <- predict(fit.bag, type = "class",  newdata = dat_test)
tabtest_bag <- predtest_bag$confusion
acc_bag <- 1 - sum(diag(tabtest_bag))/sum(tabtest_bag)
acc_bag
```

```
## [1] 0.89
```

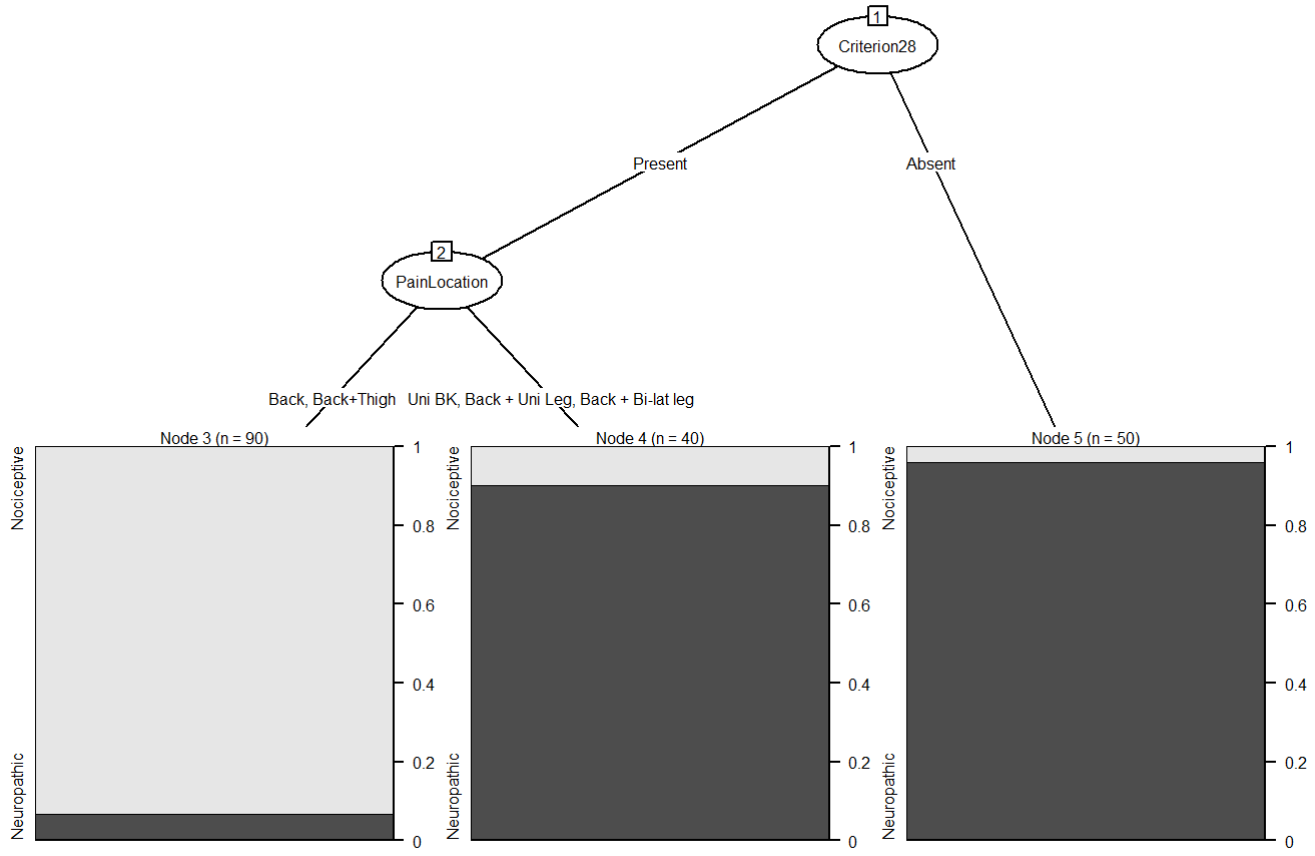
It is still not as accurate as our random forest model. This is due in part to the fact that random forests are an extension of bagging, making the trees more diverse. However we should remain skeptical of our results. Consider the difference between our regression models accuracy for our validation data and our test data; a 10 percent difference in accuracy from 0.81 to 0.91 could be very much be a product of selection bias. If we want to be more thorough so as to be completely sure random forests are the most accurate, this process should be repeated with completely different training, testing and validations, but for the sake of brevity we shall accept this result.

Observing class trees

Now I will be using various packages to explore our previously plotted class tree so as to make further observations on our dataset. First lets load them in.

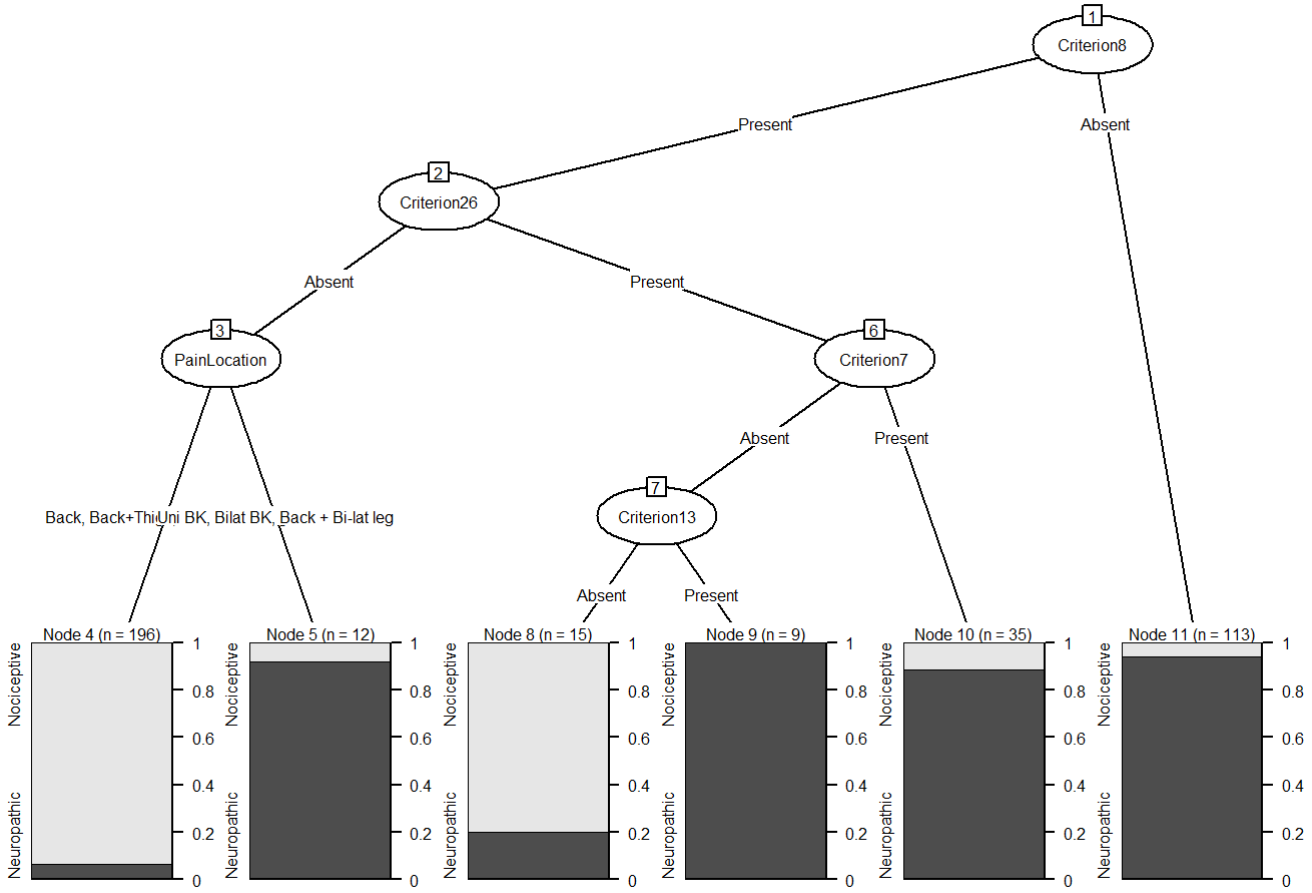
```
library(partykit)
library(rpart.plot)
library(pROC)
library(ROCR)
```

```
ct_plot <- plot(as.party(fit.ct), gp = gpar(fontsize = 6))
```



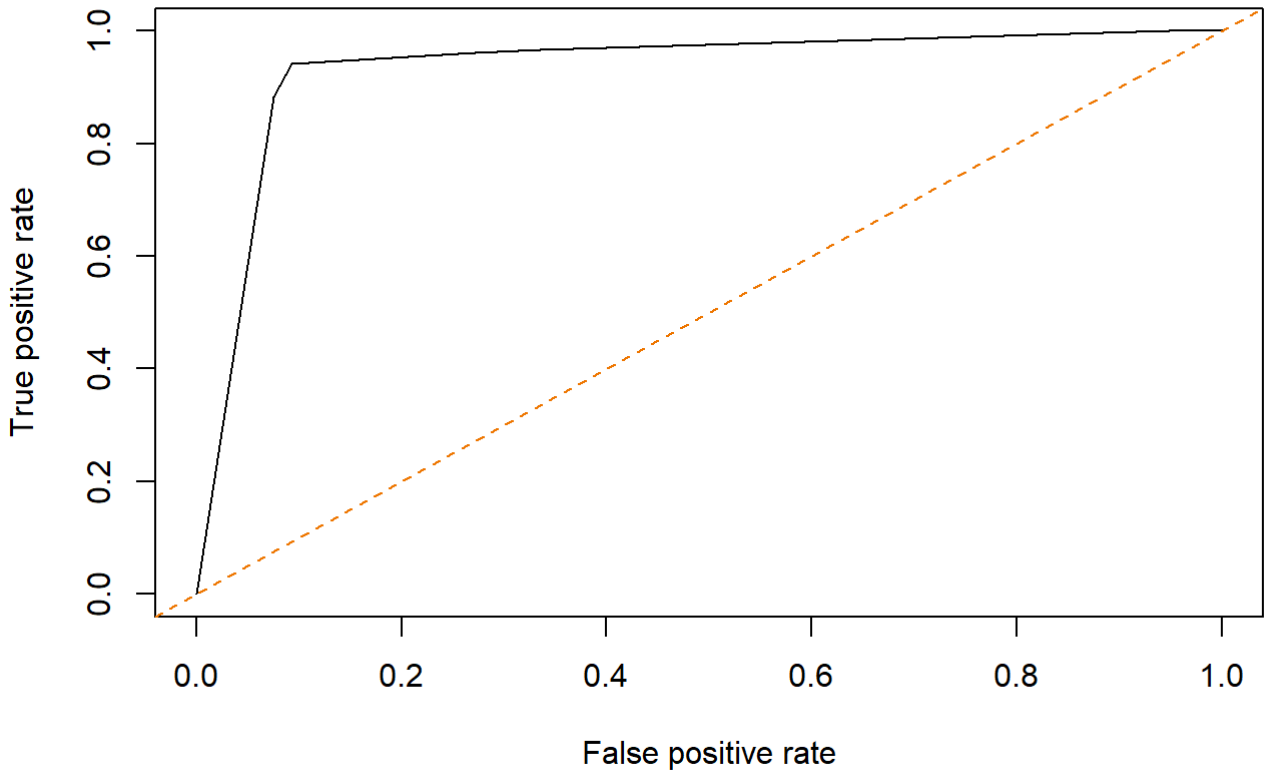
Using codeVariables, we can see that the variable with the lowest Gini impurity is “Consistent, proportionate pain reproduction on mechanical testing.”. It should be noted however that this is likely entirely due to the fact that only so many nodes are being taken into consideration. By lowering the cp value of our tree, we can make it more complex.

```
c2 <- rpart(PainDiagnosis~., data=dat, method="class", cp = 0.1/4)
plot(as.party(c2), gp = gpar(fontsize = 6))
```



Here, we see that at a higher level of complexity criterion 8 (“Localised to area of injury, dysfunction.”) is the opening rott node, primarily due to the ability to include more nodes. Now let’s graph the roc curve of this model and see what the AUc is.

```
c1 <- rpart(PainDiagnosis~., data=dat, method="class")
phat1<-predict(c1)
predob1<-prediction(1-phat1[,2],dat$PainDiagnosis)
roc1<- performance(predob1, "tpr","fpr")
plot(roc1)
abline(0,1, col = "darkorange2", lty = 2)
```

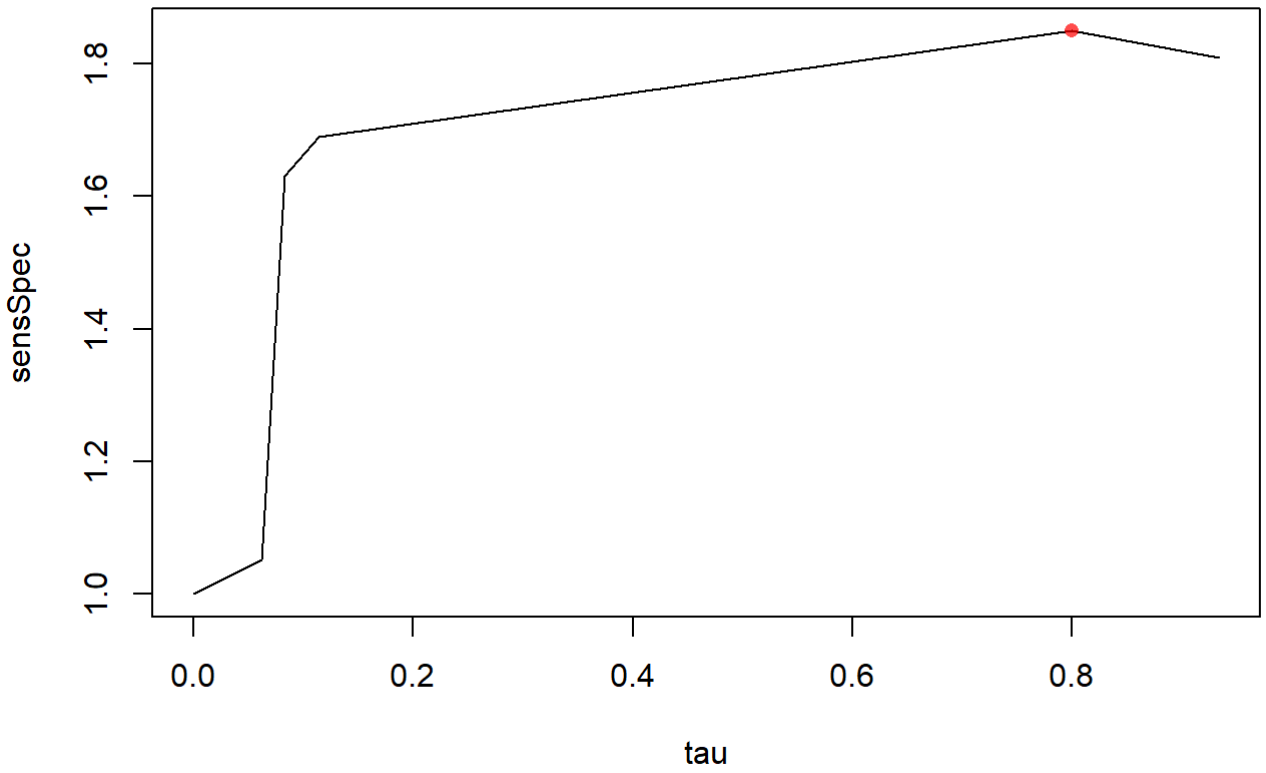



```
auc1 <- performance(predob1, "auc")
auc1@y.values
```

```
## [[1]]
## [1] 0.9352434
```

The AUC is a measure of classification performance. A value of 0.9352434 implies a high classification accuracy for our model. But what if we want to find what parameter optimises the classification rate for our model

```
sens <-performance(predob1,"sens")
spec <-performance(predob1,"spec")
tau <-sens@x.values[[1]]
sensSpec <-sens@y.values[[1]]+spec@y.values[[1]]
best <-which.max(sensSpec)
plot(tau, sensSpec,type ="l")
points(tau[best], sensSpec[best],pch =16,col =adjustcolor("red",0.7))
```




```
tau[best]
```

```
## 460
## 0.8
```

```
sens@y.values[[1]][best]
```

```
## [1] 0.942029
```

```
spec@y.values[[1]][best]
```

```
## [1] 0.9075145
```

So we have found that our model is maximised at a threshold of 0.8, with sensitivity and specificity above 0.9.

Conclusion

To conclude, I have created a wide variety of models with classification methods and found the random forest method to be the best. In doing so, I explained the methods used. I modeled and plotted the classification tree model and illustrated various levels of complexity and discussed which Criterion had the highes. Furthermore I plotted the ROC curve of the class tree model and calculated the optimum parameter, showing the point at which classification accuracy was maximised.