Evaluation

Daniel Gardner

2022-12-13

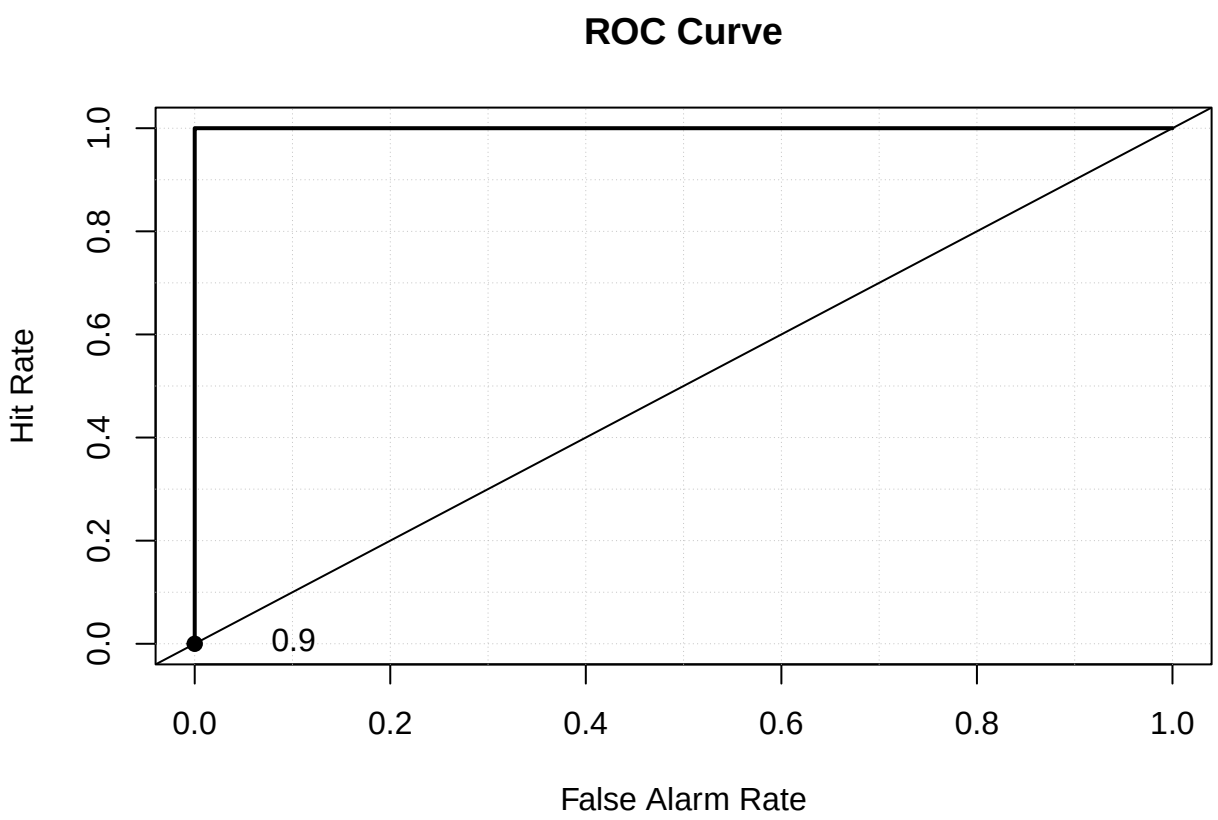
```
#Loading in packages
set.seed(1234)
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(caret))
suppressPackageStartupMessages(library(verification))
library(repr)
library(tidyverse)
library(tidymodels)
library(pROC)
```

As mentioned before, it is a bad idea to optimise just the probability of a false positive or the probability of a true negative as a validation metric, as focusing too much on one can affect the value of the other. Instead we wish to look at the relationship between the false positive rate and true positive rate, or rather their empirical forms.

By plotting the false positive rate against true positive rate at different classification thresholds, we can see how our classifier performs at identifying positives. In all ROC curves the curve is increasing; we can play it safe with a harsh threshold, getting a low amount of false positives but also a low amount of true positives. If we wanted more true positives we risk increasing the false positives.

Ideally we would want a model where we could get a very high proportion of true positives whilst keeping a very low proportion of false positives, i.e. a 'perfect ROC curve' looking like this.

```
#Define actual target value
test<-read.csv('new_test.csv')
actual <- test$target
roc.plot(actual,(actual+0.001)*0.9)
```



With the area under the curve (AUC) being a good metric to determine classification effectiveness. The AUC can be thought of as the probability of the model correctly identifying an observation. An AUC of 0.7 would imply that the model picks the correct answer 70% of the time, with an AUC of 0.5 (The y=x line through the origin) being the case where the model essentially guesses at random. [1]

There are situations where using the AUC without looking at the shape of the curve is improper, as you could have a curve with larger area but a worse shape. However as we will see below, we will not have to take this into consideration as all the ROC curves are very similar in shape and AUC score.

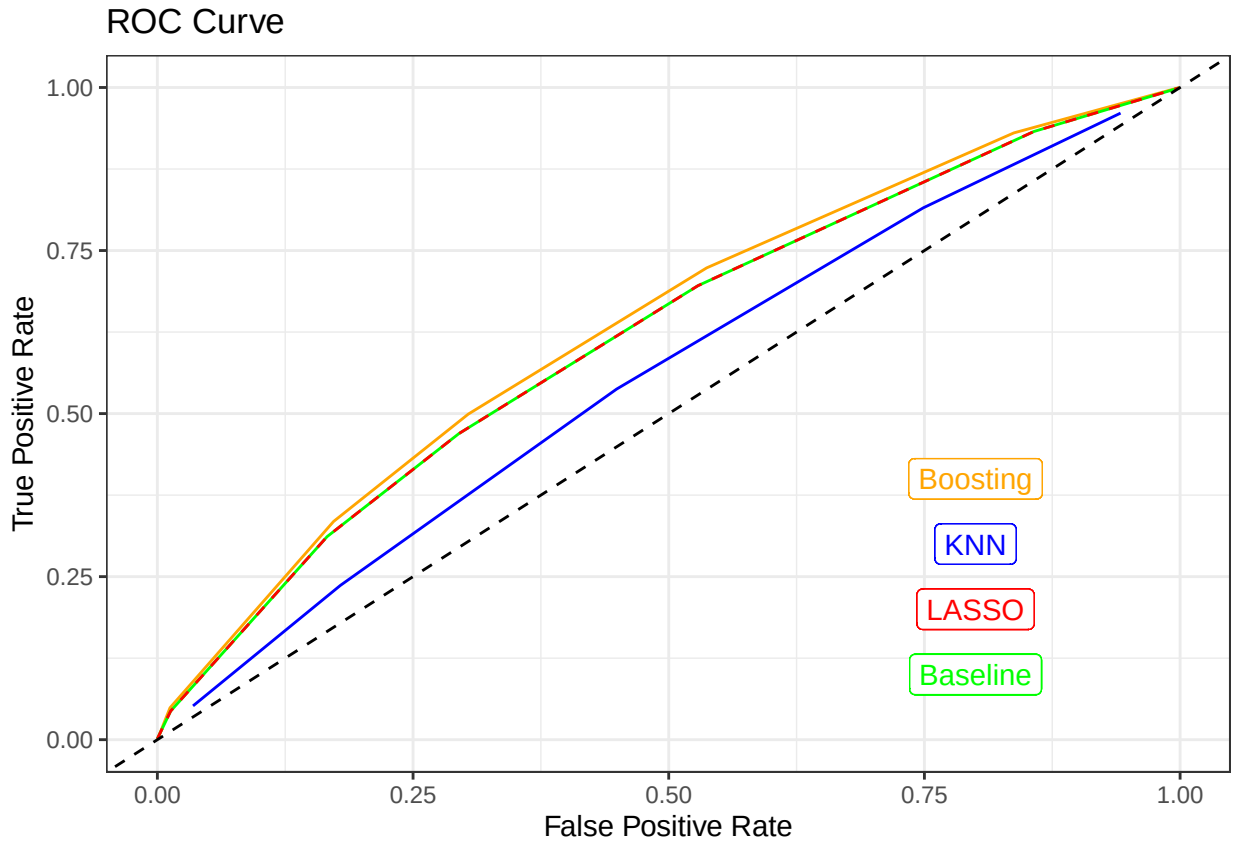
In fact, there is a direct relationship between the AUC and the Gini-Coefficient, with the Gini = 2*(AUC)-1. All the information found in the AUC is kept in the transformation to the Gini, with the main benefit being that the Gini is a score between 0 and 1 as opposed to 0.5 and 1. [2]

```
#LOADING IN PREDICTED VALUES
baselines<-read.csv('baseline_preds.csv')
baselines<-baselines[,3]
lassos<-read.csv('lasso_preds.csv')
lassos<-lassos[,3]
knns<-read.csv('knns.csv')
knns<-knns[,3]
boosts<-read.csv('lgbm_bayesian_opt.csv')
boosts<-boosts$target
```

We will manually plot the ROC curves for each set of predicted values. We first define a dataframe of classification thresholds, going from 0.01-0.3 for the baseline, LASSO and boosting models, and 0.1-0.8 in the KNN model, as it is less effective. At each threshold we calculate the empirical true and false positive rate, then plot these to produce an ROC curve.

```r
#ROC CURVE


#Creating a function for the true positive rate
tpr <- function(pred, actual) {
    res <- data.frame(pred, actual)
    sum(res$actual == 1 & res$pred == 1) / sum(actual == 1)
}

#Creating a function for the false positive rate
fpr <- function(pred, actual) {
    res <- data.frame(pred, actual)
    sum(res$actual == 0 & res$pred == 1) / sum(actual == 0)
}

#For Boosting, Baseline and LASSO we group the predicted values as such for plotting
roc_data_frame<-function(preds){
  roc_data <- data.frame(
    p0.3 = ifelse(preds > 0.3, 1, 0),
    p0.2 = ifelse(preds > 0.2, 1, 0),
    p0.1 = ifelse(preds > 0.1, 1, 0),
    p0.05 = ifelse(preds > 0.05, 1, 0),
    p0.04 = ifelse(preds > 0.04, 1, 0),
    p0.03 = ifelse(preds > 0.03, 1, 0),
    p0.02 = ifelse(preds > 0.02, 1, 0),
    p0.01 = ifelse(preds > 0.01, 1, 0))
  # reshape to long format and get fpr and tpr for each threshold
  roc_data <- roc_data %>%
      gather(key = 'threshold', value = 'pred') %>%
      group_by(threshold) %>%
      summarize(tpr = tpr(pred, actual = actual),
                fpr = fpr(pred, actual = actual))
}
#We then apply this to our predicted values to get a data frame of fpr and tpr for these 8 segments of
roc_boosts<-roc_data_frame(boosts)
roc_lassos<-roc_data_frame(lassos)
roc_baselines<-roc_data_frame(baselines)

#For the KNN predictions the predictions come in either (0,0.2,0.4,0.6,0.8,1) so we need to slightly al
preds<-knns
roc_data2 <- data.frame(
    p0.3 = ifelse(preds > 0.8, 1, 0),
    p0.2 = ifelse(preds > 0.7, 1, 0),
    p0.1 = ifelse(preds > 0.6, 1, 0),
    p0.05 = ifelse(preds > 0.5, 1, 0),
    p0.04 = ifelse(preds > 0.4, 1, 0),
    p0.03 = ifelse(preds > 0.3, 1, 0),
    p0.02 = ifelse(preds > 0.2, 1, 0),
    p0.01 = ifelse(preds > 0.1, 1, 0))
# reshape to long format and get fpr and tpr for each threshold
roc_data2 <- roc_data2 %>%
    gather(key = 'threshold', value = 'pred') %>%
    group_by(threshold) %>%
```

```
    summarize(tpr = tpr(pred, actual = actual),
              fpr = fpr(pred, actual = actual))

#Plotting the ROC curve
ggplot(data=roc_boosts, aes(x = fpr, y = tpr)) +
    geom_line(col='orange') +
    geom_line(data=roc_baselines,col='green') +
    geom_line(data=roc_data2,col='blue') +
    #(Note that the ROC curve for baseline and LASSO are so close that at this scale their lines overla
    geom_line(data=roc_baselines,col='red',lty='dashed') +
    geom_abline(intercept = 0, slope = 1, linetype = 'dashed') +
    geom_label(label='Baseline',x=0.8,y=0.1,color='green') +
    geom_label(label='LASSO',x=0.8,y=0.2,color='red') +
    geom_label(label='KNN',x=0.8,y=0.3,color='blue') +
    geom_label(label='Boosting',x=0.8,y=0.4,color='orange') +
    labs(x = 'False Positive Rate', y = 'True Positive Rate', title = 'ROC Curve') +
    theme_bw()
```



From this we can immediately see the difference in model performance. Firstly, the LASSO model is indistinguishable from the baseline at this scale, we have had to make its ROC line dashed to be able to separate the two. This implies that their ability to classify is incredibly similar. As for the Boosting model, it performs slightly better, giving the largest curve away from the null model of choosing randomly (black dotted line). The KNN model is noticeably worse than the rest, giving an ROC curve not far from the null line, implying that the model does not do much better than simply guessing at random.

We can also use the pROC package to calculate the area under these curves.

```r
#AUC SCORES
AUC<-function(preds){
  roc.area(actual,preds)$A
}
auc.data<-data.frame('Baselines'=AUC(baselines),
                     'LASSO'=AUC(lassos),
                     'Boosting'=AUC(boosts),
                     'KNN'=AUC(knns))

auc.data
```

```
##   Baselines    LASSO  Boosting       KNN
## 1 0.6274538 0.627835 0.6408526 0.5574609
```

We can now see that the LASSO model does gain a slight advantage over the baseline regression model, albeit incredibly slight compared to the major improvement of the boosting model. It is clear from these scores along with the shape of the curves that the boosting model performs best at classification at any threshold, according to this evaluation metric.

#References

Code for building ROC curve taken from https://www.kaggle.com/code/captcalculator/logistic-regression-and-roc-curve-primer

[1] https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[2] https://lenddoefl.com/news/2018/6/26/blog-on-the-use-and-misuse-of-gini-coefficients-in-credit-scoring-comparing-ginis#:~:text=How%20does%20AUC%20relate%20to,They%20measure%20exactly%20the%20same.