**An introduction to ROC, PPV and Accuracy curves**

David Goody

**Introduction**

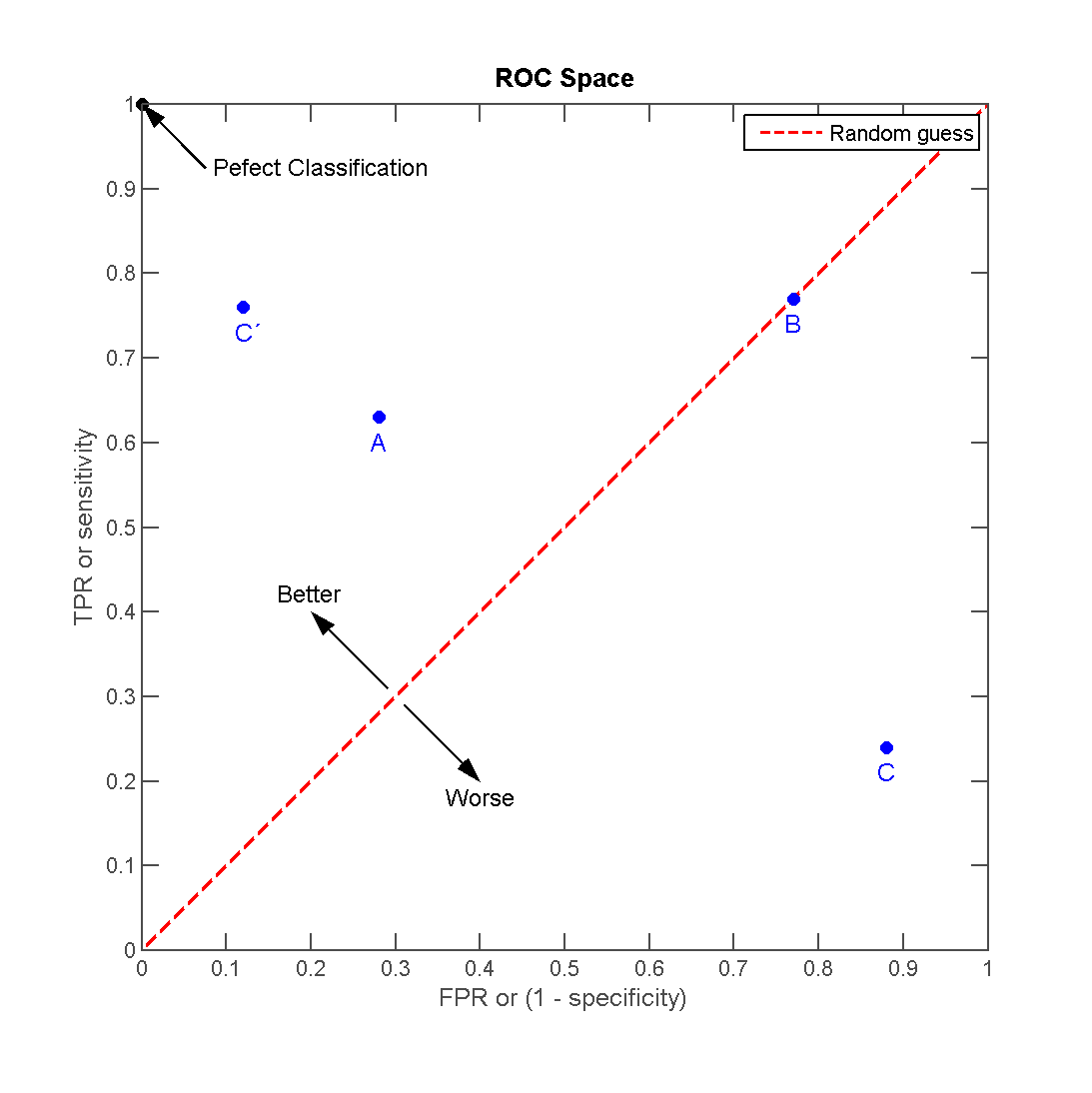
1. The outputs of binomial prediction models such as logistic regression are represented as the percentage chance that an individual case is yes or no. In many cases we will want to force the model to tell us that outcomes are either yes or no. To do this we need to select a percentage data into yes and no categories. For example if the threshold was set at 60% any result greater than 60% would count as yes and any result below 60% would count as no. To work out where to set the threshold we can use ROC, PPV or accuracy curves.

**ROC Curves**

1. ROC curves[[1]](#footnote-1) are a method of judging the quality of a model and helping to identify where the threshold should be set. It aims to highlight how we can minimise:
   1. False Negatives: we predict that a case will be false when actually it is true (also known as type I errors)
   2. False Positives: we predict that a case will be true when actually it is false (also known as type II errors)
2. The ROC curve is a plot of the true positive rate and the false positive rate. These are two measures based on these concepts on false positives and false negatives.
   1. True Positive Rate: This is the percentage of the true results we are correctly able to predict. If there are 100 true results and our model predicts 60 of these we have a true positive rate of 60%. The higher this value the higher our “hit rate” (a higher value also means we are minimising our type I error)
   2. False Positive Rate: This is the percentage of false results that we incorrectly predict as being true. If we have 100 false results and our model predicted that 20 of these were actually true we’d have a false positive rate of 20%. The lower this is, the lower our false positives (type II errors) are – meaning we “wrongly accuse” less cases.

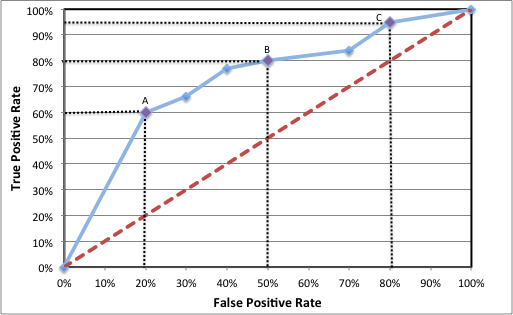
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Outcome or Prediction** | |  |
| **Condition** | **True** | **False** | **Measures** |
| **True** | True Positive (TP) | False Negative (FN)  (Type I Error) | True Positive Rate  = TP / (TP +FN) |
| **False** | False Positive (FP)  (Type II Error) | True Negative (TN) | False Positive Rate  = FP / (FP +TN) |

1. The ROC curve plots the false positive rate on it’s horizontal x-axis and the true positive rate on it’s y-axis. Anything that lies in the bottom right hand side of the chart is worse that random guessing. Anything that lies in the top left hand side of the chart is better than random guess. A line the goes through the top left hand corner of the chart is a perfect model with no type I or type II errors (it predicts every case accurately). For most problems it is nearly impossible to create model like this!

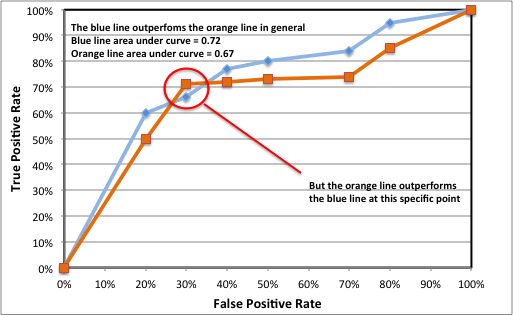


"ROC space-2" by ROC\_space.png: Indonderivative work: Kai walz (talk) - ROC\_space.png. Licensed under CC BY-SA 3.0 via Wikimedia Commons - <http://commons.wikimedia.org/wiki/File:ROC_space-2.png#/media/File:ROC_space-2.png>

1. A single set of predictions would be represented as a dot on the chart. For the binomial models we have created the True Positive Rate and False Positive Rate will vary as we change the threshold we apply to decide whether a results is true or not (i.e. if the result is greater than or equal to 60% the result is true – if the result is less than 60% the result is false). This allows us to ROC curves. We are looking for curves that maximise the true positive rate and minimise the false positive rate (curves that go as close to the top left corner as possible). By reading across these curves we can decide what level of accuracy we want.
   1. Do we want to pick a threshold that allows us to correctly identify 60% of the true cases (a 60% TPR) whilst only have 20% of the false conditions flagged as true (20% FPR)?
   2. Or would we rather correctly identify 80% of the true cases (80% TPR) but at the cost of flagging 50% of the false conditions as true (50% FPR).
   3. Or do we want to correctly identify 100% of true cases (100% TPR) but at the cost of an 80% of the false conditions being flagged as true (80% FPR)
2. As we move towards b and c we would capture more of the true cases, but at the expense of having more false positives in our results.



1. The information in ROC curves can be condensed into a single figure – the area under the curve (AUC). As its name suggests this measure the space under the curve draw on the chart. A perfect model that goes through top left corner would have an AUC value of 1. Random guessing would have an AUC value of 0.5. A model that got every prediction wrong would have a value of 0. The AUC measure is a good way of summarising the accuracy of different models, but the underlying curves should always be inspected as a model may have a lower AUC value but perform very strongly at a particular threshold. In this case you may decide to pick a model with a lower AUC value.

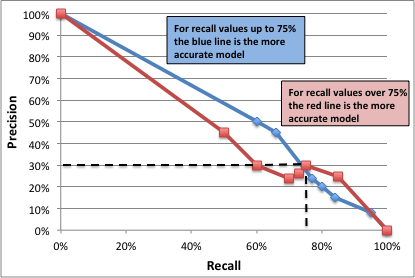


**Precision Recall (PPV) Curves**

1. An alternative to ROC curves are Precision-Recall (or PPV) curves. These again use the true positive rate (now labelled as “recall”) but this is now compared with the positive predicted value (labelled as “precision”). The precision value looks at the number of results we predict that are true which are actually true. This is a different way of measuring how to minimise the number of false positives (type II errors) than the false positive rate used in the ROC curve.

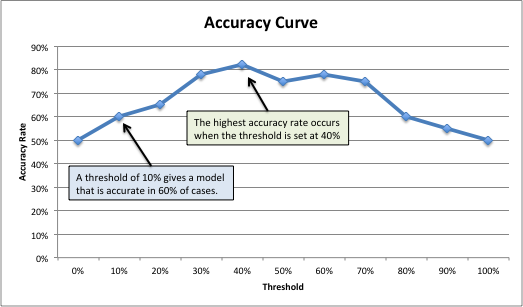
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Outcome or Prediction** | |  |
| **Condition** | **True** | **False** | **Measures** |
| **True** | True Positive (TP) | False Negative (FN)  (Type I Error) | Recall (=TPR)  = TP / (TP +FN) |
| **False** | False Positive (FP)  (Type II Error) | True Negative (TN) |  |
| **Measures** | Precision  = TP / (TP + FP) |  |  |

1. An ideal model on a PPV curve will go through the top right hand corner. This means the model has can identify all the correct values (100% recall) without any false positives (100% precision). As the curve moves away from the top right hand corner the amount of correct values identified (recall) and the purity of the results (precision) drop, making a model less accurate. As with the ROC curves there the best model will be dependent on the level of precision and coverage you require in your results. The following chart highlights an example of this.



**Accuracy Curves**

1. A disadvantage of ROC and PPV curves are that you can’t see the threshold you are applying on either of the axes. One way around this is to add labels at certain points on the ROC or PPV curve. Any alternative approach is the show the threshold on the x-axis of a chart. You can then plot this against a quantity that reflects the overall accuracy of the data. You can calculate this by dividing the number of correctly identified cases (true positive + true negative) by the total number of cases.
2. The accuracy curve this gives you allows a high level view of how the model is working at different thresholds. The main limitation is that you cannot distinguish between the amount of type I errors (false negatives) and type II errors (false positive). This is often an important factor when setting a threshold.



**Further Reading:**

University Of Georgia - <http://ebp.uga.edu/courses/Chapter%204%20-%20Diagnosis%20I/8%20-%20ROC%20curves.html>

Wikipedia (ROC curve) - <http://en.wikipedia.org/wiki/Receiver_operating_characteristic>

Wikipedia (PPV curve) - <http://en.wikipedia.org/wiki/Precision_and_recall>

**Annex A – Creating ROC and PPV curves using R**

#EXAMPLE OF CREATING ROC, PPV AND ACCURACY CURVES IN R

#This example uses is based on the passenger list from the Titanic. We are trying to

#predict who survived and who died based on their age (Age), the fare they paid (Fare)

#gender (Female), embarkation point (South), how many siblings/spouses in each party (SibSp)

#how many parents/children in each passengers party (Parch)

#The dataset is used is publically available. It is the basis of a machine learning

#example on Kaggle (https://www.kaggle.com/c/titanic-gettingStarted).

####################################################################

#SET-UP

#Install the relevant packages

require("glmnet") #For GLM / logistic regression models

require("ridge") #For Ridge regression models

require("rpart") #For recursive partitioning (decision trees) models

require("ROCR") #For ROC, PPV and accuracy curves

#Set the working directory where you will be loading and saving data from

setwd("/Users/datascientist3/Desktop/R Packages/Kaggle Titanic")

#Load Titanic dataset that we will train the data on

Titanic.Train <- read.csv("titanic\_train\_kaggle.csv", stringsAsFactors=FALSE)

#TIDY UP DATA TO ENABLE THE MODELS TO BE FITTED

#Convert text fields to dummy variables with 1/0 flags for use in equation

#Convert gender field into a 1/0 flag for female status

Titanic.Train$Female[Titanic.Train$Sex=='female'] <- 1

Titanic.Train$Female[Titanic.Train$Sex=='male'] <- 0

#Convert embarkation point 1/0 flag for those boarding at Southampton

Titanic.Train$South[Titanic.Train$Embarked=='S'] <- 1

Titanic.Train$South[Titanic.Train$Embarked!='S'] <- 0

#FIT THE MODEL

#Apply Random Forest model to predict whether passengers survived based on the 6 variables

Titanic.Train.GLM.model <- glm(Survived ~ Female + Age + SibSp + Parch + Fare + South,

data= Titanic.Train,family=binomial("logit"))

Titanic.Train.Ridge.model <- logisticRidge(Survived ~ Female + Age + SibSp + Parch + Fare + South,

data=Titanic.Train,lambda="automatic")

Titanic.Train.RP.model <- rpart(Survived ~ Sex + Age + SibSp + Parch + Fare + Embarked,

data=Titanic.Train, method="class",

control=rpart.control(minsplit=8, minbucket=4, cp=0.005, maxdepth = 10))

#APPLY THE MODEL

#In this case we fit logistic regression (GLM), Ridge regression

#and recursive partitioning decision tree models

Titanic.Train.GLM.Preds <-predict(Titanic.Train.GLM.model,Titanic.Train,type="response")

Titanic.Train.Ridge.Preds <-predict(Titanic.Train.Ridge.model,Titanic.Train,type="response")

Titanic.Train.RP.Preds <- predict(Titanic.Train.RP.model,type="prob")

#Transform prediction data into a standardized format for further analysis

pred.GLM <- prediction(Titanic.Train.GLM.Preds,Titanic.Train$Survived)

pred.Ridge <- prediction(Titanic.Train.Ridge.Preds,Titanic.Train$Survived)

pred.RP <- prediction(Titanic.Train.RP.Preds[,2],Titanic.Train$Survived)

#Calculate true positive rate (tpr) and false postive rate (fpr) for these models

perf.roc.GLM <- performance(pred.GLM,"tpr","fpr")

perf.roc.Ridge <- performance(pred.Ridge,"tpr","fpr")

perf.roc.RP <- performance(pred.RP,"tpr","fpr")

#Plot ROC curves

#Different line type (lty) values (1=solid, 2=dashed, 3=dotted)

#To see full range of colours type colors()

plot(perf.roc.GLM,xlim=c(0,1),ylim=c(0,1),lty=1,col='blue')

plot(perf.roc.Ridge,xlim=c(0,1),ylim=c(0,1),lty=1,col='red',add=TRUE)

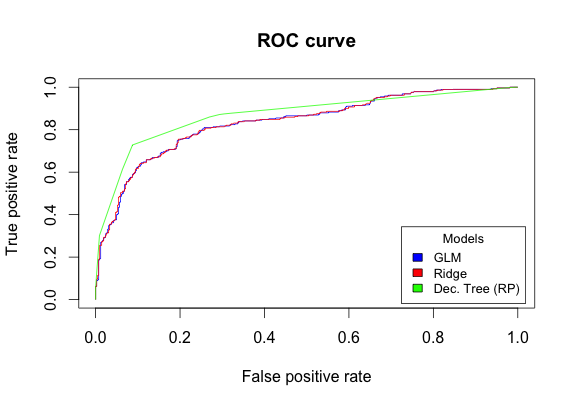
plot(perf.roc.RP,xlim=c(0,1),ylim=c(0,1),lty=1,col='green',add=TRUE)

title(main="ROC curve")

legend("bottomright", inset=.02, title="Models", cex = 0.8,

c("GLM","Ridge","Dec. Tree (RP)"),

fill=c('blue','red','green'), horiz=FALSE)



#Calculate area under the ROC curve – implies decision tree is the strongest model overall

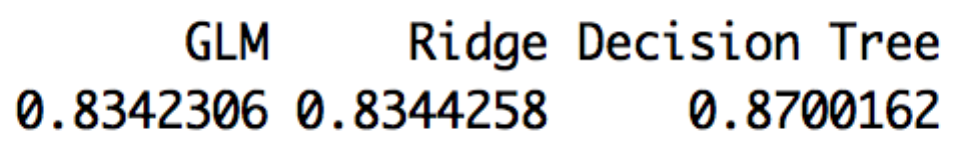
#Ridge regression outperforms logistic regression (GLM) – but only just

Area.Under.Curve <- cbind("GLM"=performance(pred.GLM,"auc")@y.values[[1]],

"Ridge"=performance(pred.Ridge,"auc")@y.values[[1]],

"Decision Tree"=performance(pred.RP,"auc")@y.values[[1]])

Area.Under.Curve



#Calculate precision (True Positive / Total Positives Predicted)

#against recall (true positives / total observed positives)

#Note Prec v Recall is equivalent to PPV v TPR

perf.ppv.GLM <- performance(pred.GLM,"prec","rec")

perf.ppv.Ridge <- performance(pred.Ridge,"prec","rec")

perf.ppv.RP <- performance(pred.RP,"prec","rec")

#Plot Precision/Recall (PPV) curves

plot(perf.ppv.GLM,xlim=c(0,1),ylim=c(0,1),lty=1,col='blue')

plot(perf.ppv.Ridge,xlim=c(0,1),ylim=c(0,1),lty=1,col='red',add=TRUE)

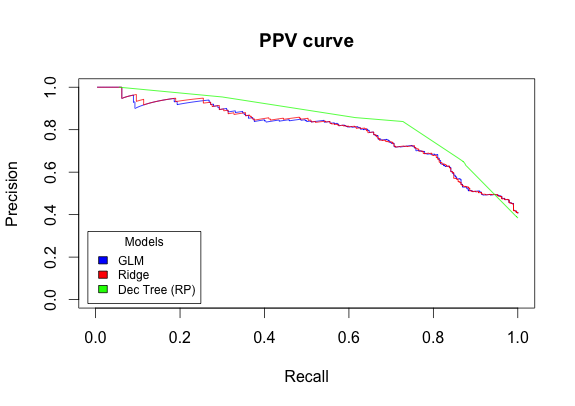
plot(perf.ppv.RP,xlim=c(0,1),ylim=c(0,1),lty=1,col='green',add=TRUE)

title(main="PPV curve")

legend("bottomleft", , inset=0.02, title="Models", cex=0.75,

c("GLM","Ridge","Dec Tree (RP)"),

fill=c('blue','red','green'), horiz=FALSE)



#Calculate accuracy (correctly identified cases / all cases = acc)

#and cutoff (the threshold applied) for these models

perf.acc.GLM <- performance(pred.GLM,"acc","cutoff")

perf.acc.Ridge <- performance(pred.Ridge,"acc","cutoff")

perf.acc.RP <- performance(pred.RP,"acc","cutoff")

#Plot Accuracy/Cutoff curves

plot(perf.acc.GLM,xlim=c(0,1),ylim=c(0,1),lty=1,col='blue')

plot(perf.acc.Ridge,xlim=c(0,1),ylim=c(0,1),lty=1,col='red',add=TRUE)

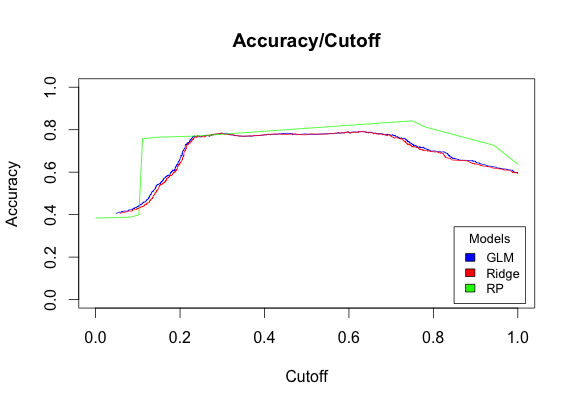
plot(perf.acc.RP,xlim=c(0,1),ylim=c(0,1),lty=1,col='green',add=TRUE)

title(main="Accuracy/Cutoff")

legend("bottomright", inset=.02, title="Models", cex = 0.8,

c("GLM","Ridge","Dec Tree (RP)"),

fill=c('blue','red','green'), horiz=FALSE)



#Plot predictions from models against each other - shows level of correlation

plot(Titanic.Train.GLM.Preds, Titanic.Train.Ridge.Preds, xlab="GLM", ylab="Ridge")

plot(Titanic.Train.GLM.Preds, Titanic.Train.RP.Preds[,2], xlab="GLM", ylab="Dec Tree (RP)")

|  |  |
| --- | --- |
| **GLM v Ridge results**  The results are basically  the same from both models | **GLM v Decision Tree results**  There is a greater level of variation between the GLM and decision tree models |

#Add predictions into a data frame then calculate the correlation between them

Predictions.All.Models <- cbind("GLM" = Titanic.Train.GLM.Preds,

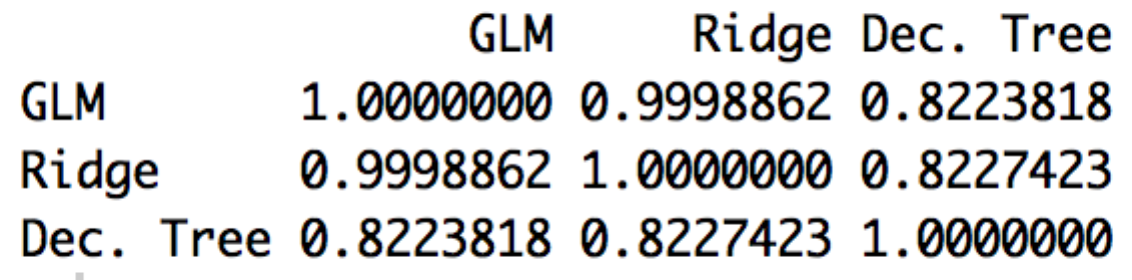
"Ridge" = Titanic.Train.Ridge.Preds[,1],

"Dec. Tree" = Titanic.Train.RP.Preds[,2])

cor(Predictions.All.Models)

#Shows GLM and Ridge are entirely correlated (~0.999), but decision tree differs. (~0.82)

#This reflects what we saw in the charts above.



#If you want to get the underlying cut-off values for the ROC and PPV. For example:

perf.roc.GLM

perf.ppv.GLM

1. Reciever Operating Characteristic (ROC) curves [↑](#footnote-ref-1)