Performance Evaluation Metrics

Applied Machine Learning for Educational Data Science

true

09/15/2021

Contents

[Updated: Thu, Sep 16, 2021 - 17:04:10]

In the next few weeks, we will talk about different approaches to build predictive models. We need some performance metrics to evaluate how useful a model is by itself, or how well it performs compared to alternative models. We will consider the performance metrics in two main categories: 1) metrics for predicting a continuous outcome, 2) metrics for predicting a binary outcome. Most of these metrics are easy to calculate with simple mathematical operations, so we will write some code to calculate them. The availability of these metrics may depend on the tools you are using.

We will use two hypothetical datasets for introducing these concepts. The first hypothetical dataset will have a continuous outcome and predictions. The second hypothetical dataset will have a binary outcome and predictions.

You can load these two hypothetical datasets using the code below.

```
d_con <- read.csv('https://raw.githubusercontent.com/uo-datasci-specialization/c4-ml-fall-2021/main/data

head(d_con)
```

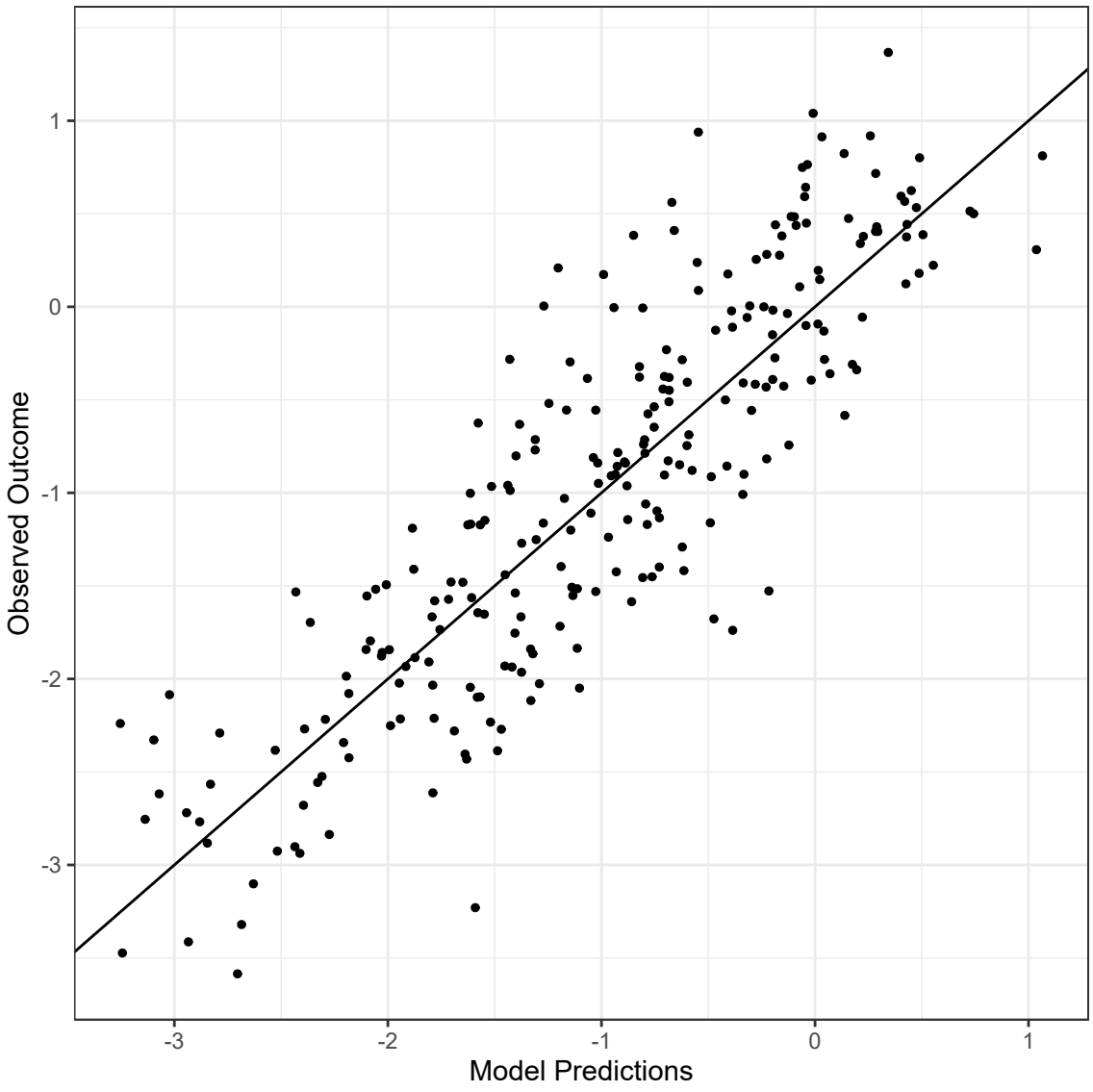
```
  id        obs       pred
1  1 -1.0599708 -0.7932058
2  2 -1.0974086 -0.7400672
3  3 -0.5835326  0.1392355
4  4 -0.4420990 -0.7118047
5  5 -1.2384322 -0.9673121
6  6  0.4851151 -0.1116890
```

```r
d_bin <- read.csv('https://raw.githubusercontent.com/uo-datasci-specialization/c4-ml-fall-2021/main/data

head(d_bin)
```

```
  id      pred out
1  5 0.2047574   1
2 11 0.1557319   0
3 13 0.1769403   0
4 19 0.2377235   0
5 22 0.3164912   0
6 34 0.3459672   1
```

1. Evaluation Metrics for Continuous Outcomes

It is first useful to plot the observed values and predicted values to examine if there are certain areas where the model systematically overpredicts or under-predicts the outcome. The plot below is a simple scatterplot and the diagonal line is a line where the predicted outcome and observed outcome are equal. In this example, there is no indication of systematic bias in model predictions.



Accuracy

A family of metrics commonly used for models predicting a continuous outcome is related to the accuracy of predictions. These metrics have a single most important unit in our predictions, the prediction error for a single observation. Error (a.k.a. residual) for an observation is defined as the difference between the observed value and model predicted value for the outcome.

$$e_i = y_i - \hat{y}_i$$

Below is a list of of commonly used metrics that is a function of prediction error.

Mean Absolute Error(MAE)

$$MAE = \frac{\sum_{i=1}^{N} |e_i|}{N}$$

```r
mae <- mean(abs(d_con$obs-d_con$pred))

mae
```

```
[1] 0.4246783
```

Mean Squared Error(MSE)

$$MSE = \frac{\sum_{i=1}^{N} e_i^2}{N}$$

```r
mse <- mean((d_con$obs-d_con$pred)^2)
mse
```

```
[1] 0.2820917
```

Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} e_i^2}{N}}$$

```r
rmse <- sqrt(mean((d_con$obs-d_con$pred)^2))
rmse
```

```
[1] 0.5311231
```

Reduction in Total Amount of Error

Another metric that may give a useful insight about a model's predictive performance is the reduction in the total amount of error. If we consider that we don't have any information about the outcome, our best guess would be the mean value of the outcome to make a prediction for each observation coming from the same population. In other words, our model would yield a constant prediction for each future observation. We can also consider this an intercept-only model or null model.

In our case, if we use the mean to predict the outcome for each observation, the sum of squared error would be equal to 284.003.

$$SSE = \sum_{i=1}^{N} (y - \bar{y})^2$$

```r
mean(d_con$obs)
```

```
[1] -0.9589447
```

```r
sse_null <- sum((d_con$obs-mean(d_con$obs))^2)

sse_null
```

```
[1] 284.0027
```

Instead, if we rely on our model to predict the outcome, the sum of squared error would be equal to 70.523

```
sse_model <- sum((d_con$obs-d_con$pred)^2)

sse_model
```

```
[1] 70.52293
```

By using our model instead of a simple mean, we improved our predictions such that the total amount of error is reduced from 284.003 to 70.523. Therefore, we can say that the total amount of prediction error is reduced by about 75% when we use our model instead of a simple null model.

$$1 - \frac{SSE_{model}}{SSE_{null}}$$

```
1 - (sse_model/sse_null)
```

```
[1] 0.7516822
```

In other words, SSE_null provides a reference point when you do the worst job possible (using mean). If we have a perfect model, then SSE would be 0, the best job you could do. So, we can evaluate a model's predictive performance by measuring where it stands between these two extreme points.

2. Evaluation Metrics for Binary Outcomes

When the outcome is a binary variable, most models typically yield a probability estimate for a class membership (or, a continuous valued prediction between 0 and 1). For instance, in our hypothetical dataset, the predicted value for the first observation (ID=5) is 0.204, indicating that the model predicts that the probability of this observation being in Group 1 is 0.204. On the other hand, the observation indeed belongs to Group 1 (out=1). Therefore, the model doesn't do a good job of prediction for this observation.

```
head(d_bin)
```

```
  id      pred out
1  5 0.2047574   1
2 11 0.1557319   0
3 13 0.1769403   0
4 19 0.2377235   0
5 22 0.3164912   0
6 34 0.3459672   1
```

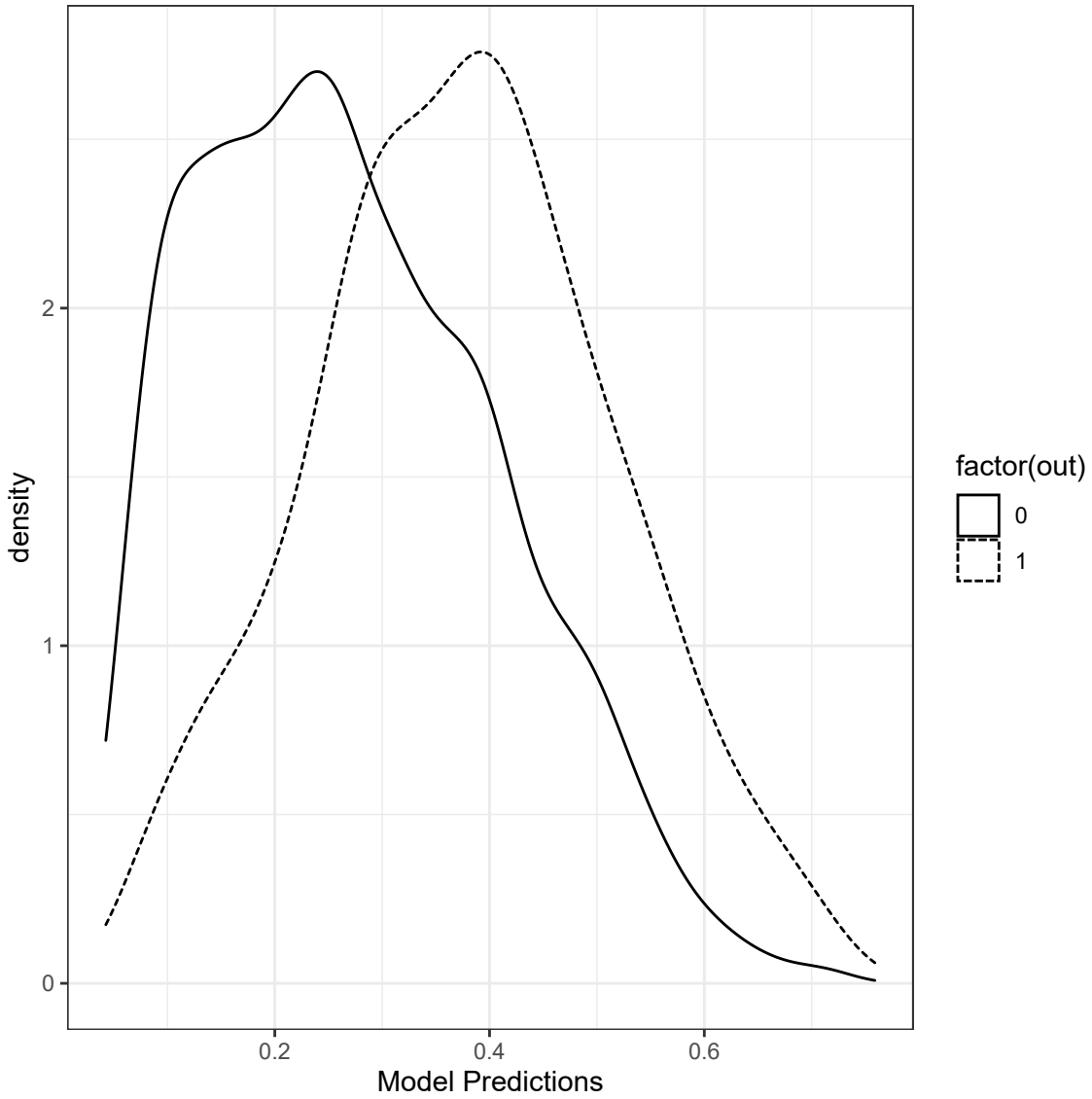
Let's look at another observation. The model predicts a probability of 0.759 for being in Group 1, and the observation is indeed in Group 1. Therefore, the model does a relatively good job for this observation.

```
d_bin[2129,]
```

```
        id      pred out
2129 18783 0.7588673   1
```

In an ideal situation where a model does a perfect job to predict a binary outcome, we expect all those observations in Group 0 to have a predicted probability of 0 and all those observations in Group 1 to have a predicted probability of 1. So, predicted values close to 0 for observations in Group 0 and predicted values close to 1 for observations in Group 1 are desirable.

One way to look at the quality of predictions for a binary outcome is to examine the distribution of predictions for both classes. For this hypothetical dataset, we can see that the predicted probability of being in Group 1 are slightly higher on average for those in Group 1. This implies that there is some value in our model predictions. The more separation between the distribution of two classes, the better the model performance is. Later, we will see numerical summaries of separation between these two distributions.



Accuracy of Class Probabilities

We can calculate MAE, MSE, and RMSE to measure the accuracy of predicted probabilities when the outcome is binary. They have the same definition as in the continuous case.

```r
# Mean absolute error

mean(abs(d_bin$out-d_bin$pred))
```

```
[1] 0.3746168
```

```r
# Mean squared error (a.k.a., Brier score for binary outcomes)

mean((d_bin$out-d_bin$pred)^2)
```

```
[1] 0.1851416
```

```r
# Root mean squared error

sqrt(mean((d_bin$out-d_bin$pred)^2))
```

```
[1] 0.4302808
```

We can similarly consider the reduction in total amount of error for binary cases. When the outcome is binary, the proportion of the observation in Group 1 (mean) becomes the best estimate for any observation without any information. So, we can consider the null case is when we assign a predicted probability to each observation such that the probability is equal to the proportion of people in Group 1 in the population (or our best estimate of it). Then, we can apply the same thoughts for this case to measure the reduction in the total amount of error when a model is used to make predictions.

```r
# SSE_null

mean(d_bin$out)
```

```
[1] 0.2926024
```

```r
sse_null <- sum((d_bin$out-mean(d_bin$out))^2)

sse_null
```

```
[1] 626.7543
```

```r
# SSE_model

sse_model <- sum((d_bin$out-d_bin$pred)^2)

sse_model
```

```
[1] 560.6087
```

```r
# Proportional reduction in prediction error

1 - (sse_model/sse_null)
```

```
[1] 0.1055368
```

Accuracy of Class Predictions

Confusion Matrix and Related Metrics

In most situations, accuracy of class probabilities is not very useful because one has to make a decision and place these observations in a group based on these probabilities. For instance, suppose you are predicting whether or not an individual will be recidivated so you can take action based on this decision. Then, you have to transform this continuous probability (or probability-like score) predicted by a model into a binary prediction. Therefore, one has to determine an arbitrary cut-off value. Once a cut-off value for this transformation is determined, then we have class predictions. Consider that we use a cut-off value of 0.5 for the hypothetical dataset. So, if an observation has a predicted class probability less than 0.5, we predict that this person is in Group 0 (or whatever 0 represents). Likewise, if an observation has a predicted class probability higher than 0.5, we predict that this person is in Group 1.

In the code below, I convert the class probabilities to class proportions using a cut-off value of 0.5.

```
d_bin$pred_class <- ifelse(d_bin$pred>.5,1,0)

head(d_bin)
```

```
  id      pred out pred_class
1  5 0.2047574   1          0
2 11 0.1557319   0          0
3 13 0.1769403   0          0
4 19 0.2377235   0          0
5 22 0.3164912   0          0
6 34 0.3459672   1          0
```

As we now have a binary outcome and a binary prediction, we can summarize them in a 2 x 2 table. This is called **confusion matrix**.

```
tab <- table(d_bin$pred_class,
             d_bin$out,
             dnn = c('Predicted','Observed'))

tab
```

```
         Observed
Predicted    0    1
        0 2005  720
        1  137  166
```

The table indicates that there are 2142 observations with an outcome 0 and the model accurately predicts 0 for 2005 of them while inaccurately predicts 1 for 137 of them. Similarly, there are 886 observations with an outcome 1 and model accurately predicts 1 for 166 of them while inaccurately predicts 0 for 720 of them.

Based on the elements of this table, we can define four key concepts:

- **True Positives(TP)**: True positives are the observations where both the outcome and prediction are equal to 1. They represent correct predictions for those observations the event is observed.
- **True Negative(TN)**: True negatives are the observations where both the outcome and prediction are equal to 0. They also represent correct predictions for those observations the event is not observed.

- **False Positives(FP)**: False positives are the observations where the outcome is 0 but the prediction is 1. They represents incorrect predictions for those observations the event is not observed.
- **False Negatives(FN)**: False negatives are the observations where the outcome is 1 but the prediction is 0. They represents incorrect predictions for those observations the event is observed.

Several metrics can be defined based on these four variables. We define a few important metrics below.

```
tn <- tab[1,1]
tp <- tab[2,2]
fp <- tab[2,1]
fn <- tab[1,2]
```

- **Accuracy**: Overall accuracy simply represent the proportion of correct predictions.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

```
acc <- (tp + tn)/(tp+tn+fp+fn)

acc
```

```
[1] 0.7169749
```

- **True Positive Rate (Sensitivity)**: True positive rate (a.k.a. sensitivity) is the proportion of correct predictions for those observations the outcome is 1 (event is observed).

$$TPR = \frac{TP}{TP + FN}$$

```
tpr <- (tp)/(tp+fn)

tpr
```

```
[1] 0.1873589
```

- **True Negative Rate (Specificity)**: True negative rate (a.k.a. specificity) is the proportion of correct predictions for those observations the outcome is 0 (event is not observed).

$$TNR = \frac{TN}{TN + FP}$$

```
tnr <- (tn)/(tn+fp)

tnr
```

```
[1] 0.9360411
```

- **Positive predicted value (Precision)**: Positive predicted value (a.k.a. precision) is the proportion of correct decisions when the model predicts that the outcome is 1.

$$PPV = \frac{TP}{TP + FP}$$

```
ppv <- (tp)/(tp+fp)

ppv
```

```
[1] 0.5478548
```

- **F1 score**: F1 score is a metric that combines both PPV and TPR.

$$F1 = 2 * \frac{PPV * TPR}{PPV + TPR}$$

```
f1 <- (2*ppv*tpr)/(ppv+tpr)

f1
```

```
[1] 0.2792262
```

Area Under the Receiver Operating Curve (AUC or AUROC)

The confusion matrix and related metrics all depend on the arbitrary cut-off value one picks when transforming continuous predicted probabilities to binary predicted classes. We can change the cut-off value to optimize certain metrics, and there is always a trade-off between these metrics for different cut-off values. For instance, let's pick different cut-off values and calculate these metrics for each one.

```
# Write a generic function to return the metric for a given vector of observed
# outcome, predicted probability and cut-off value

cmat <- function(x,y,cut){
  # x, a vector of predicted probabilities
  # y, a vector of observed outcomes
  # cut, user-defined cut-off value

  x_ <- ifelse(x>cut,1,0)

  tn <- sum(x_==0 & y==0)
  tp <- sum(x_==1 & y==1)
  fp <- sum(x_==1 & y==0)
  fn <- sum(x_==0 & y==1)

  acc <- (tp + tn)/(tp+tn+fp+fn)
  tpr <- (tp)/(tp+fn)
  tnr <- (tn)/(tn+fp)
  ppv <- (tp)/(tp+fp)
  f1 <- (2*ppv*tpr)/(ppv+tpr)

  return(list(acc=acc,tpr=tpr,tnr=tnr,ppv=ppv,f1=f1))
}
```

```r
# Try it out

  # cmat(x=d_bin$pred,y=d_bin$out,cut=0.5)

# Do it for different cut-off values

metrics <- data.frame(cut=seq(0,1,0.01),
                      acc=NA,
                      tpr=NA,
                      tnr=NA,
                      ppv=NA,
                      f1=NA)


for(i in 1:nrow(metrics)){

  cmat_ <- cmat(x   = d_bin$pred,
                y   = d_bin$out,
                cut = metrics[i,1])

  metrics[i,2:6] = c(cmat_$acc,cmat_$tpr,cmat_$tnr,cmat_$ppv,cmat_$f1)

}

metrics
```

```
     cut       acc         tpr         tnr       ppv          f1
1   0.00 0.2926024 1.000000000 0.000000000 0.2926024 0.452733776
2   0.01 0.2926024 1.000000000 0.000000000 0.2926024 0.452733776
3   0.02 0.2926024 1.000000000 0.000000000 0.2926024 0.452733776
4   0.03 0.2926024 1.000000000 0.000000000 0.2926024 0.452733776
5   0.04 0.2926024 1.000000000 0.000000000 0.2926024 0.452733776
6   0.05 0.2959049 1.000000000 0.004668534 0.2935719 0.453893443
7   0.06 0.3018494 0.997742664 0.014005602 0.2950601 0.455435343
8   0.07 0.3130779 0.994356659 0.031279178 0.2980379 0.458615305
9   0.08 0.3266182 0.988713318 0.052754435 0.3015491 0.462147191
10  0.09 0.3398283 0.981941309 0.074229692 0.3049422 0.465365071
11  0.10 0.3570013 0.977426637 0.100373483 0.3100609 0.470780103
12  0.11 0.3741744 0.970654628 0.127450980 0.3151337 0.475795297
13  0.12 0.3916777 0.965011287 0.154528478 0.3207052 0.481418919
14  0.13 0.4029062 0.959367946 0.172735761 0.3241800 0.484606613
15  0.14 0.4164465 0.949209932 0.196078431 0.3281311 0.487677588
16  0.15 0.4329590 0.935665914 0.225023343 0.3330655 0.491259259
17  0.16 0.4494716 0.925507901 0.252567694 0.3387030 0.495917750
18  0.17 0.4640026 0.918735892 0.275910364 0.3441860 0.500768994
19  0.18 0.4798547 0.911963883 0.301120448 0.3505423 0.506424318
20  0.19 0.4930647 0.898419865 0.325396825 0.3551986 0.509114167
21  0.20 0.5056143 0.888261851 0.347338936 0.3601831 0.512536633
22  0.21 0.5188243 0.872460497 0.372549020 0.3651393 0.514818515
23  0.22 0.5366579 0.858916479 0.403361345 0.3732222 0.520341880
24  0.23 0.5521797 0.848758465 0.429505135 0.3809524 0.525874126
25  0.24 0.5660502 0.834085779 0.455182073 0.3877230 0.529369628
26  0.25 0.5785997 0.816027088 0.480392157 0.3937908 0.531227039
27  0.26 0.5977543 0.800225734 0.514005602 0.4051429 0.537936267
```

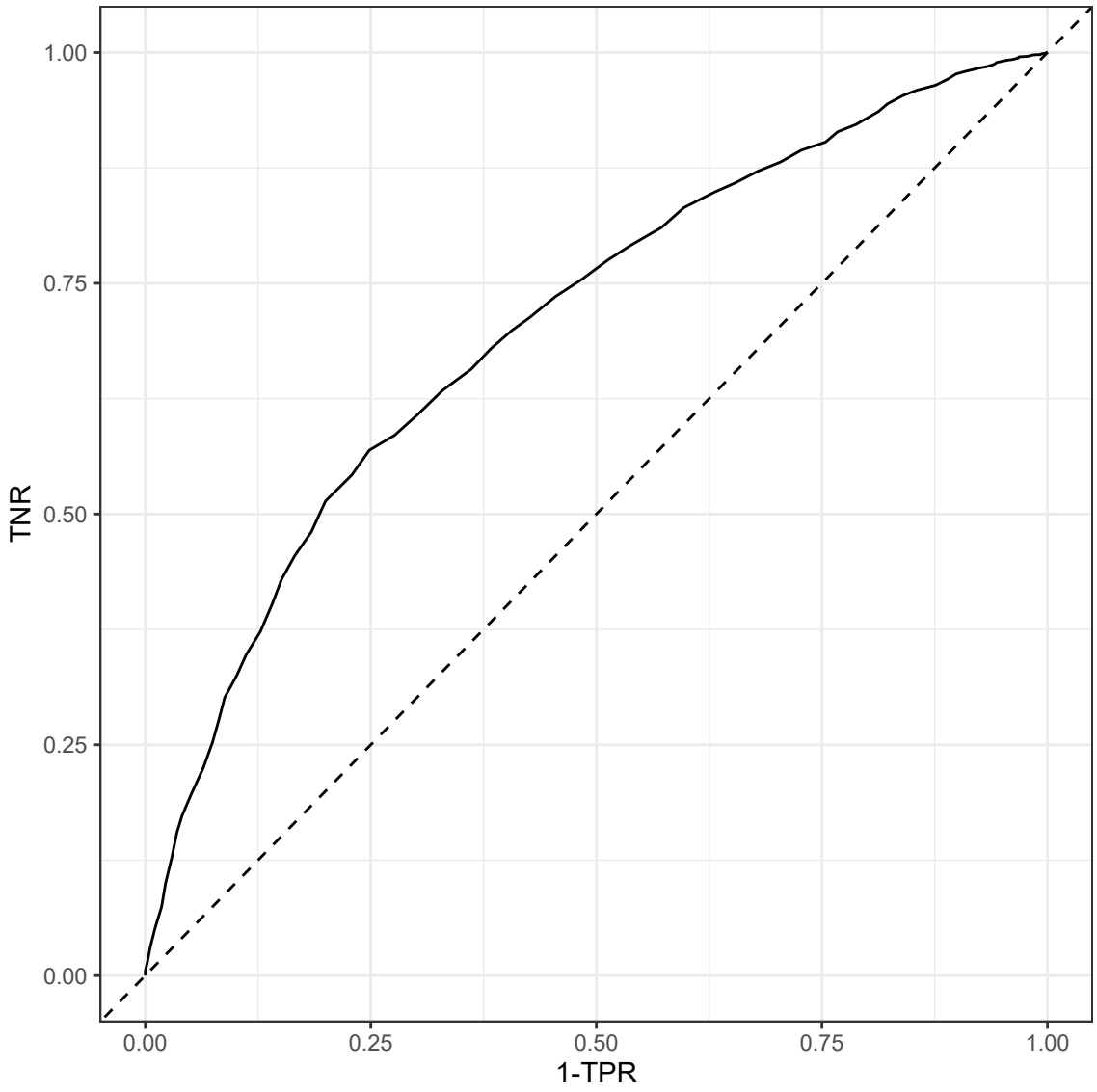
28	0.27	0.6093131	0.770880361	0.542483660	0.4107035	0.535896430
29	0.28	0.6225231	0.751693002	0.569094304	0.4191315	0.538181818
30	0.29	0.6258256	0.723476298	0.585434174	0.4192283	0.530848861
31	0.30	0.6344122	0.697516930	0.608309991	0.4241592	0.527528809
32	0.31	0.6446499	0.670428894	0.633986928	0.4310595	0.524734982
33	0.32	0.6515852	0.638826185	0.656862745	0.4350500	0.517604024
34	0.33	0.6611625	0.616252822	0.679738562	0.4431818	0.515580737
35	0.34	0.6680978	0.593679458	0.698879552	0.4491887	0.511424404
36	0.35	0.6723910	0.573363431	0.713352007	0.4527629	0.505976096
37	0.36	0.6799868	0.545146727	0.735760971	0.4604385	0.499224806
38	0.37	0.6846103	0.515801354	0.754435107	0.4649034	0.489031568
39	0.38	0.6908851	0.487584650	0.774976657	0.4726477	0.480000000
40	0.39	0.6948481	0.461625282	0.791316527	0.4778037	0.469575201
41	0.40	0.6984808	0.427765237	0.810457516	0.4828025	0.453620586
42	0.41	0.7064069	0.402934537	0.831932773	0.4979079	0.445414847
43	0.42	0.7083884	0.369074492	0.848739496	0.5023041	0.425504229
44	0.43	0.7087186	0.346501129	0.858543417	0.5032787	0.410427807
45	0.44	0.7103699	0.320541761	0.871615313	0.5080501	0.393079585
46	0.45	0.7100396	0.295711061	0.881419234	0.5077519	0.373751783
47	0.46	0.7123514	0.273137698	0.894024276	0.5159915	0.357195572
48	0.47	0.7107001	0.246049661	0.902894491	0.5117371	0.332317073
49	0.48	0.7146631	0.232505643	0.914098973	0.5282051	0.322884013
50	0.49	0.7143329	0.212189616	0.922035481	0.5295775	0.302981467
51	0.50	0.7169749	0.187358916	0.936041083	0.5478548	0.279226241
52	0.51	0.7199472	0.177200903	0.944444444	0.5688406	0.270223752
53	0.52	0.7212682	0.160270880	0.953314659	0.5867769	0.251773050
54	0.53	0.7209379	0.145598194	0.958916900	0.5944700	0.233907525
55	0.54	0.7186262	0.124153499	0.964519141	0.5913978	0.205223881
56	0.55	0.7192867	0.110609481	0.971055089	0.6125000	0.187380497
57	0.56	0.7206077	0.101580135	0.976657330	0.6428571	0.175438596
58	0.57	0.7196169	0.091422122	0.979458450	0.6480000	0.160237389
59	0.58	0.7176354	0.075620767	0.983193277	0.6504854	0.135490394
60	0.59	0.7163144	0.067720090	0.984593838	0.6451613	0.122574055
61	0.60	0.7156539	0.058690745	0.987394958	0.6582278	0.107772021
62	0.61	0.7163144	0.056433409	0.989262372	0.6849315	0.104275287
63	0.62	0.7146631	0.045146727	0.991596639	0.6896552	0.084745763
64	0.63	0.7133421	0.038374718	0.992530345	0.6800000	0.072649573
65	0.64	0.7126816	0.032731377	0.993930906	0.6904762	0.062500000
66	0.65	0.7133421	0.031602709	0.995331466	0.7368421	0.060606061
67	0.66	0.7110304	0.022573363	0.995798319	0.6896552	0.043715847
68	0.67	0.7103699	0.018058691	0.996732026	0.6956522	0.035203520
69	0.68	0.7103699	0.016930023	0.997198880	0.7142857	0.033076075
70	0.69	0.7093791	0.012415350	0.997665733	0.6875000	0.024390244
71	0.70	0.7080581	0.007900677	0.997665733	0.5833333	0.015590200
72	0.71	0.7083884	0.006772009	0.998599440	0.6666667	0.013407821
73	0.72	0.7073976	0.002257336	0.999066293	0.5000000	0.004494382
74	0.73	0.7077279	0.001128668	1.000000000	1.0000000	0.002254791
75	0.74	0.7077279	0.001128668	1.000000000	1.0000000	0.002254791
76	0.75	0.7077279	0.001128668	1.000000000	1.0000000	0.002254791
77	0.76	0.7073976	0.000000000	1.000000000	NaN	NaN
78	0.77	0.7073976	0.000000000	1.000000000	NaN	NaN
79	0.78	0.7073976	0.000000000	1.000000000	NaN	NaN
80	0.79	0.7073976	0.000000000	1.000000000	NaN	NaN
81	0.80	0.7073976	0.000000000	1.000000000	NaN	NaN

```
82  0.81 0.7073976 0.000000000 1.000000000       NaN         NaN
83  0.82 0.7073976 0.000000000 1.000000000       NaN         NaN
84  0.83 0.7073976 0.000000000 1.000000000       NaN         NaN
85  0.84 0.7073976 0.000000000 1.000000000       NaN         NaN
86  0.85 0.7073976 0.000000000 1.000000000       NaN         NaN
87  0.86 0.7073976 0.000000000 1.000000000       NaN         NaN
88  0.87 0.7073976 0.000000000 1.000000000       NaN         NaN
89  0.88 0.7073976 0.000000000 1.000000000       NaN         NaN
90  0.89 0.7073976 0.000000000 1.000000000       NaN         NaN
91  0.90 0.7073976 0.000000000 1.000000000       NaN         NaN
92  0.91 0.7073976 0.000000000 1.000000000       NaN         NaN
93  0.92 0.7073976 0.000000000 1.000000000       NaN         NaN
94  0.93 0.7073976 0.000000000 1.000000000       NaN         NaN
95  0.94 0.7073976 0.000000000 1.000000000       NaN         NaN
96  0.95 0.7073976 0.000000000 1.000000000       NaN         NaN
97  0.96 0.7073976 0.000000000 1.000000000       NaN         NaN
98  0.97 0.7073976 0.000000000 1.000000000       NaN         NaN
99  0.98 0.7073976 0.000000000 1.000000000       NaN         NaN
100 0.99 0.7073976 0.000000000 1.000000000       NaN         NaN
101 1.00 0.7073976 0.000000000 1.000000000       NaN         NaN
```

Notice the trade-off between TPR (sensitivity) and TNR (specificity). The more conservative cut-off value we pick (a higher probability), the higher TNR is and the lower TPR is.

A receiver operating characteristic curve (ROC) is plot that represents this dynamic relationship between TPR and TNR for varying levels of a cut-off value.

```
ggplot(data = metrics, aes(x=1-tpr,y=tnr))+
  geom_line()+
  xlab('1-TPR')+
  ylab('TNR')+
  geom_abline(lty=2)+
  theme_bw()
```



ROC may be a useful plot to inform about model's predictive power as well as choosing an optimal cut-off value. The area under the ROC curve (AUC or AUROC) is typically used to evaluate the predictive power of classification models. The diagonal line in this plot represents a hypothetical model with no predictive power and AUC for the diagonal line is 0.5 (it is half of the whole square). The more ROC curve resembles with the diagonal line, less the predictive power is. The solid line is the ROC curve for our predictions in the hypothetical dataset. It is not easy to calculate AUC by hand or straight formula as it requires calculus and numerical approximations. There are many alternatives in R to calculate AUC. We will use the `cutpointr` package as it also provides other tools to select an optimal cut-off point.

```
# install.packages('cutpointr')

require(cutpointr)

cut.obj <- cutpointr(x     = d_bin$pred,
                     class = d_bin$out)

auc(cut.obj)
```

```
[1] 0.7048538
```

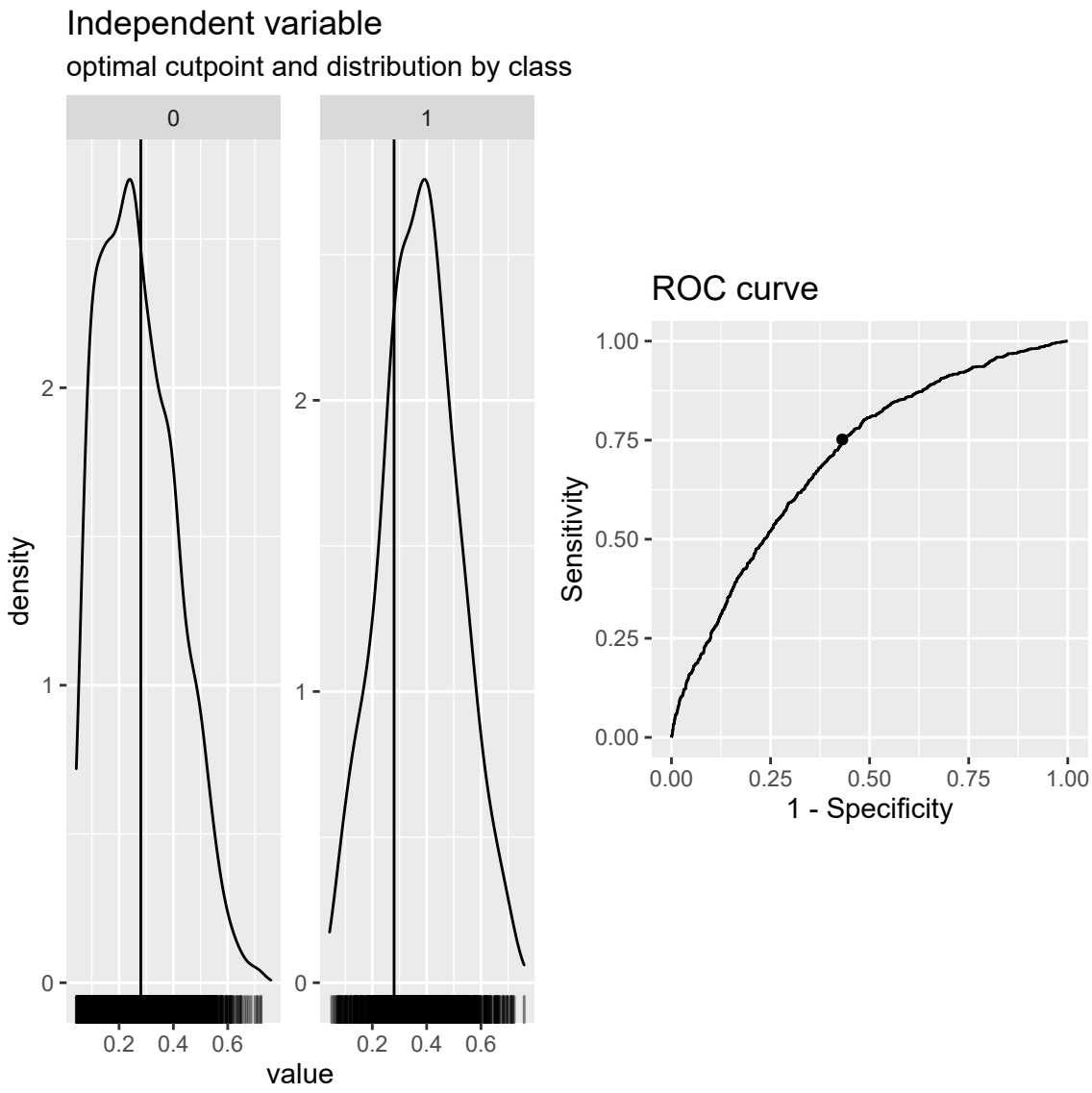
We see that AUC for the predictions in the hypothetical dataset is 0.705. This can be considered as mediocre in terms of predictive power. The closer AUC is to 1, the more predictive power the model has. The closer AUC is to 0.5, the closer predictive power is to random guessing. The magnitude of AUC is also closely related to how well the predicted probabilities are separated for two classes.

The `cutpointr` package provides more in terms of finding optimal values to maximize certain metrics. For instance, suppose we want to find the optimal cut-off value that maximizes the sum of specificity and sensitivity.

```
cut.obj <- cutpointr(x      = d_bin$pred,
                     class  = d_bin$out,
                     method = maximize_metric,
                     metric = sum_sens_spec)

cut.obj
```

```
# A tibble: 1 x 16
  direction optimal_cutpoint method          sum_sens_spec      acc sensitivity
  <chr>                <dbl> <chr>                   <dbl>    <dbl>       <dbl>
1 >=                0.280027 maximize_metric       1.32079 0.622523    0.751693
  specificity      AUC pos_class neg_class prevalence outcome predictor
        <dbl>    <dbl>     <int>     <int>      <dbl> <chr>   <chr>
1    0.569094 0.704854         1         0   0.292602 class   x
  data                 roc_curve                    boot
  <list>               <list>                       <lgl>
1 <tibble [3,028 x 2]> <roc_cutpointr [3,029 x 10]> NA
```

```
plot(cut.obj)
```



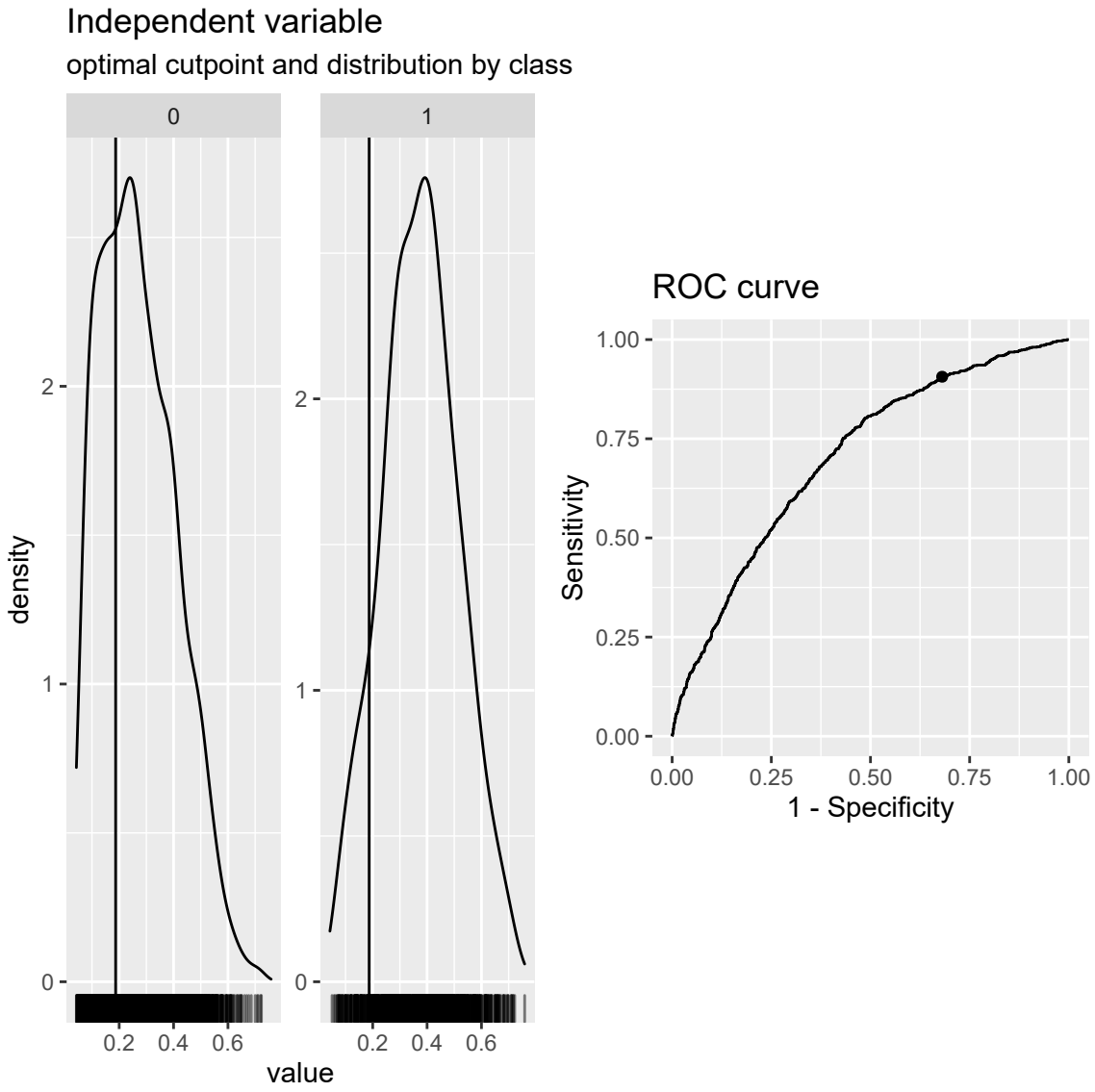
You can also create custom cost functions to maximize if you can attach a $ value to TP, FP, TN, FN. For instance, suppose that a true-positive prediction made by the model brings a 5 dollars profit and a false-positive prediction made by the model costs 1 dollars. A true-negative or a false-negative doesn't cost or profit anything. Then, you can find an optimal cut-off value that maximizes the total profit.

```
# Custom function

cost <- function(tp,fp,tn,fn,...){
  my.cost <- matrix(5*tp - 1*fp + 0*tn + 0*fn, ncol = 1)
  colnames(my.cost) <- 'my.cost'
  return(my.cost)
}

cut.obj <- cutpointr(x      = d_bin$pred,
                     class  = d_bin$out,
                     method = maximize_metric,
                     metric = cost)
```

```
cut.obj
```

```
# A tibble: 1 x 16
  direction optimal_cutpoint method          my.cost      acc sensitivity
  <chr>                <dbl> <chr>             <dbl>    <dbl>       <dbl>
1 >=                0.186995 maximize_metric    2557 0.491083    0.906321
  specificity      AUC pos_class neg_class prevalence outcome predictor
        <dbl>    <dbl>     <int>     <int>      <dbl> <chr>   <chr>
1    0.319328 0.704854         1         0   0.292602 class   x
  data                 roc_curve                       boot
  <list>               <list>                          <lgl>
1 <tibble [3,028 x 2]> <roc_cutpointr [3,029 x 10]> NA
```

```
plot(cut.obj)
```



Check this page to find more information about the `cutpointr` package.