Classification Metrics

Rob McCulloch

1. Classification Metrics

To examine the *fit* of a model we need a metric to measure our success or, conversely, a loss to measure our failure.

For numeric outcomes, the industry standard is RMSE (root mean squared error) or just MSE.

For classification, there are a few different metrics that are used that we need to be aware of.

We will look at

- cross entropy
- the confusion matrix and miss-classification
- ROC and AUC

The cross-entropy is just the minus log likelihood we saw with the logit.

We have already used the second one in text classification with Naive Bayes.

2. Cross Entropy

For categorical outcomes cross entropy is just another name for the (- log likelihood loss).

For a binary out come if $y \in \{0, 1\}$ and $\hat{p}$ is the probability of y=1 from a model then we often write

$$L(y, \hat{p}) = -[y \log(\hat{p}) + (1-y) \log(1-\hat{p})].$$

Which is the same as $-\log(\hat{p})$ if $y = 1$ and $-\log(1-\hat{p})$ if $y = 0$ which is minus the log of the probability of what happened.

For data (train or test) $\{x_i, y_i\}$, with $\hat{p}_i$ the estimated prob $Y = 1$ given x_i and a model, then we sum (or average) the loss:

$$L(y, \hat{p}) = \frac{1}{n} \sum_{i=1}^{n} -[y_i \log(\hat{p}_i) + (1-y_i) \log(1-\hat{p}_i))].$$

In sklearn.metrics this is simply called the log loss.

```
In [5]: #from sklearn.metrics import log_loss
   ...: y_true = [0, 0, 1, 1]
   ...: y_pred = [[.9, .1], [.8, .2], [.3, .7], [.01, .99]]
   ...: print(log_loss(y_true, y_pred))
   ...: temp = -np.log(.9) -np.log(.8) - np.log(.7) - np.log(.99)
   ...: print(temp/4.0)
0.1738073366910675
0.1738073366910675
```

For a multinomial outcome with $y \in \{1, 2, \ldots, K\}$.

Given $x_i,\ i = 1, 2, \ldots, n$, let $p_{ij} = P(Y = j \mid x_i)$.

Let $y_{ij} = 1$ if $Y_i = j$ and 0 otherwise.

Then the loss is (average of - log lik) is

$$L = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{K} y_{ij} \log(p_{ij}).$$

in sklearn.metrics

3.3.2.12. Log loss

Log loss, also called logistic regression loss or cross-entropy loss, is defined on probability estimates. It is commonly used in (multinomial) logistic regression and neural networks, as well as in some variants of expectation-maximization, and can be used to evaluate the probability outputs (`predict_proba`) of a classifier instead of its discrete predictions.

For binary classification with a true label $y \in \{0, 1\}$ and a probability estimate $p = \Pr(y = 1)$, the log loss per sample is the negative log-likelihood of the classifier given the true label:

$$L_{\log}(y, p) = -\log \Pr(y|p) = -(y \log(p) + (1 - y) \log(1 - p))$$

This extends to the multiclass case as follows. Let the true labels for a set of samples be encoded as a 1-of-K binary indicator matrix Y, i.e., $y_{i,k} = 1$ if sample i has label k taken from a set of K labels. Let P be a matrix of probability estimates, with $p_{i,k} = \Pr(y_{i,k} = 1)$. Then the log loss of the whole set is

$$L_{\log}(Y, P) = -\log \Pr(Y|P) = -\frac{1}{N} \sum_{i=0}^{N-1} \sum_{k=0}^{K-1} y_{i,k} \log p_{i,k}$$

To see how this generalizes the binary log loss given above, note that in the binary case, $p_{i,0} = 1 - p_{i,1}$ and $y_{i,0} = 1 - y_{i,1}$, so expanding the inner sum over $y_{i,k} \in \{0, 1\}$ gives the binary log loss.

The `log_loss` function computes log loss given a list of ground-truth labels and a probability matrix, as returned by an estimator's `predict_proba` method.

```
>>> from sklearn.metrics import log_loss
>>> y_true = [0, 0, 1, 1]
>>> y_pred = [[.9, .1], [.8, .2], [.3, .7], [.01, .99]]
>>> log_loss(y_true, y_pred)
0.1738...
```

The first `[.9, .1]` in `y_pred` denotes 90% probability that the first sample has label 0. The log loss is non-negative.

3. Confusion and Miss-classification

Let's use the forensic glass data.

Glass has been shattered and you are trying to guess the type of glass which is one of the three categores WinF, WinNF, Other, from three features obtained from the glass shards.

summary(ddf)

```
   type           RI                 Al                 Na        
 WinF :70   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 WinNF:76   1st Qu.:0.2358   1st Qu.:0.2804   1st Qu.:0.3274  
 Other:68   Median :0.2867   Median :0.3333   Median :0.3865  
            Mean   :0.3167   Mean   :0.3598   Mean   :0.4027  
            3rd Qu.:0.3515   3rd Qu.:0.4174   3rd Qu.:0.4654  
            Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
```

Note that the three x's are already standardized.

Let's use KNN in R.

```
near = kknn(type~.,ddf,ddf,k=10,kernel = "rectangular")
```

Note that I am looking at the *in-sample* "fit".
I only have 214 observations.

```
near$fitted[1:50]:
```

```
 [1] WinF  WinNF WinNF WinF  WinF  WinNF WinF  WinF  WinF  WinF  WinNF WinF
[13] WinNF WinF  WinF  WinF  WinF  WinF  WinF  WinNF WinNF WinF  WinF  WinF
[25] WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinF  Other
[37] WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinNF WinNF WinF
[49] WinF  WinF
Levels: WinF WinNF Other
```

```
near$prob[1:5,]
```

```
     WinF WinNF Other
[1,]  0.6   0.3   0.1
[2,]  0.4   0.4   0.2
[3,]  0.1   0.9   0.0
[4,]  0.7   0.3   0.0
[5,]  0.8   0.2   0.0
```

The two-way table relating the observed Y with the predicted (or fitted) Y is called the *confusion matrix*.

Data label on columns, "fitted" label on rows.

So, there are 58+11+1 observations with Y =WinF.
Of those 11 were predicted to be WinNF.

```
knnfit  WinF WinNF Other
  WinF    58    13    14
  WinNF   11    57    12
  Other    1     6    42
```

We like the diagonals big!
Missclassification rate: (214-(58+57+42))/214 = 0.27

Here is the confusion matrix from the multinomial logit fit:

```
logitfit WinF WinNF Other
   WinF    45    19    15
   WinNF   21    45    13
   Other    4    12    40
```

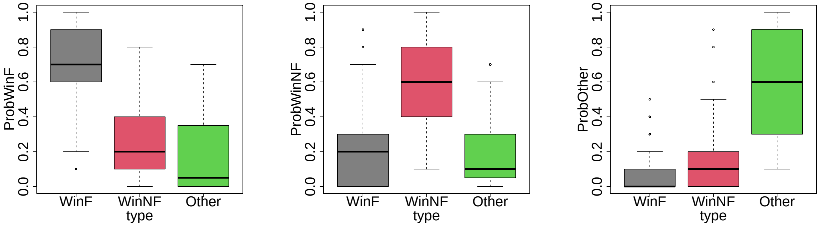
```
> (214-(45+45+40))/214
[1] 0.3925234
```

Not as good as from KNN.

But this is in-sample !!!!!

How good are the probabilities ??

The first plot is $P(Y = WinF \mid x)$ vs. y=glass type.
The second plot is $P(Y = WinNF \mid x)$ vs. y=glass type.
The third plot is $P(Y = Other \mid x)$ vs. y=glass type.



pretty good !!

4. ROC and AUC

ROC and AUC are two popular methods for assessing the quality of a classifier for a binary y.

ROC stands for the incomprehensible term "receiver operator characteristics".

We look at missclassification rates for various values of s using the rule: classify $Y = 1$ if $P(Y = 1 \mid x) \approx \hat{p} > s$.

In particular, we consider probability cutoffs s other than .5.

The ROC curve summarizes the 2x2 confusion matrix given $\hat{y} = 1$ if $\hat{p} > s$ and 0 otherwise as s varies.

Given an s value we have the confusion matrix:

```
        y=0  y=1
yhat=0  TN   FN
yhat=1  FP   TP
```

where:

TN: correctly classified 0
FP: incorrectly classified 1
FN: incorrectly classified 0
TP: correctly classified 1

```
         y=0  y=1
yhat=0   TN   FN
yhat=1   FP   TP
```

The Sensitivity is

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

out of the $y = 1$ *observations, what fraction do we get right*

The Specificity is

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

out of the $y = 0$ *observations, what fraction do we get right*

For the tabloid data y=1 if a customer responds to a promotion and 0 otherwise.

x are things about the customer, mostly from past purchase behaviour.

We classify $Y = 1$ if $P(Y = 1 \mid x) \approx \hat{p} > .02$ we get this confusion matrix.

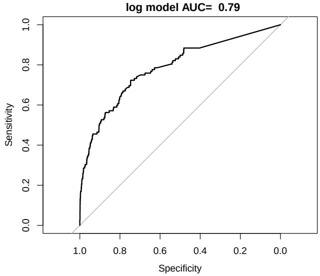
```
    y
yhat    0    1
   0 3616   31
   1 1272   81
```

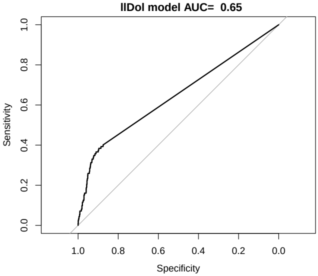
ROC looks at:

- Sensitivity: % of y=1 correctly classified: 81/(81+31) = 0.72
- Specificity: % of y=0 correctly classified: 3616/(1272+3616) = 0.74

We want Sensitivity and Specificity big.

For each value of the cutoff s you will get a pair of Sensitivity and Specificity values.
The ROC curve plots the Specificity vs. the Sensitivity as you vary the cutoff s.
AUC is the area under the ROC curve.





As we go from left to right, s goes from 1 to 0.

At $s = 1$, $\hat{y} = 0$ for all the observations so, we get all the 0's right but none of the 1's. (Specificity,Sensitivity) = (1,0).

At $s = 0$, $\hat{y} = 1$ for all the observations so, we get all the 1's but none of the 0's. (Specificity,Sensitivity) = (0,1).

There are many measures based on these same quantities!!!

Precision is TP/(TP + FP).

Recall (same as Sensitivity) is TP/(TP + FN).

F1 score is harmonic mean of Precision and recall.