caret train() predicts very different then predict.glm()

I am trying to estimate a logistic regression, using the 10-fold cross-validation.

```
#import libraries
library(car); library(caret); library(e1071); library(verification)

#data import and preparation
data(Chile)
chile        <- na.omit(Chile)  #remove "na's"
chile        <- chile[chile$vote == "Y" | chile$vote == "N" , ] #only "Y" and "N" required
chile$vote   <- factor(chile$vote)      #required to remove unwanted levels
chile$income <- factor(chile$income)  # treat income as a factor
```

Goal is to estimate a glm - model that predicts to outcome of vote "Y" or "N" depended on relevant explanatory variables and, based on the final model, compute a confusion matrix and ROC curve to grasp the models behaviour for different threshold levels.

Model selection leads to:

```
res.chileIII <- glm(vote ~
                          sex       +
                          education +
                          statusquo ,
                          family = binomial(),
                          data = chile)
#prediction
chile.pred <- predict.glm(res.chileIII, type = "response")
```

generates:

```
> head(chile.pred)
         1          2          3          4          5          6
0.974317861 0.008376988 0.992720134 0.095014139 0.040348115 0.090947144
```

to compare the observed with estimation:

```
chile.v     <- ifelse(chile$vote == "Y", 1, 0)          #to compare the two arrays
chile.predt <- function(t) ifelse(chile.pred > t , 1,0) #t is the threshold for which the
confusion matrix shall be computed
```

confusion matrix for t = 0.3:

```
confusionMatrix(chile.predt(0.3), chile.v)

> confusionMatrix(chile.predt(0.3), chile.v)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 773  44
         1  94 792

               Accuracy : 0.919
                 95% CI : (0.905, 0.9315)
    No Information Rate : 0.5091
    P-Value [Acc > NIR] : < 2.2e-16
```

and the Roc-curve:

```
roc.plot(chile.v, chile.pred)
```

which seems as a reasonable model.

Now instead of using the "normal" predict.glm() function I want to test out the performance difference to a 10-fold cross-validation estimation.

```
tc <- trainControl("cv", 10, savePredictions=T)  #"cv" = cross-validation, 10-fold
fit <- train(chile$vote ~ chile$sex            +
                          chile$education      +
                          chile$statusquo      ,
                          data      = chile    ,
                          method    = "glm"    ,
                          family    = binomial ,
                          trControl = tc)

> summary(fit)$coef
                    Estimate Std. Error   z value      Pr(>|z|)
(Intercept)         1.0152702  0.1889646  5.372805  7.752101e-08
`chile$sexM`       -0.5742442  0.2022308 -2.839549  4.517738e-03
`chile$educationPS` -1.1074079  0.2914253 -3.799971  1.447128e-04
`chile$educationS`  -0.6827546  0.2217459 -3.078996  2.076993e-03
`chile$statusquo`    3.1689305  0.1447911 21.886224 3.514468e-106
```

all parameters significant.

```
fitpred <- ifelse(fit$pred$pred == "Y", 1, 0) #to compare with chile.v

> confusionMatrix(fitpred,chile.v)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 445 429
         1 422 407

 Accuracy : 0.5003
                 95% CI : (0.4763, 0.5243)
    No Information Rate : 0.5091
    P-Value [Acc > NIR] : 0.7738
```

which is obviously very different from the previous confusion matrix. My expectation was that the cross validated results should not perform much worse then the first model. However the results show something else.

My assumption is that there is a mistake with the settings of the train() parameters but I can't figure it out what it is.

I would really appreciate some help, thank you in advance.

r glm r-caret confusion-matrix

edited Dec 10 '15 at 13:27

grubjesic
1,655 ● 4 ● 9 ● 18

asked May 22 '14 at 12:15

Vincent
378 ● 3 ● 11

1 Answer

You are trying to get an idea of the in sample fit using a confusion matrix. Your first approach using the `glm()` function is fine.

The problem with the second approach using `train()` lies in the returned object. You are trying to extract the in sample fitted values from it by `fit$pred$pred`. However, `fit$pred` does not contain the fitted values that are aligned to `chile.v` or `chile$vote`. It contains the observations and fitted values of the different (10) folds:

```
> head(fit$pred)
  pred obs rowIndex parameter Resample
1    N   N        2      none   Fold01
2    Y   Y       20      none   Fold01
3    Y   Y       28      none   Fold01
4    N   N       38      none   Fold01
5    N   N       55      none   Fold01
6    N   N       66      none   Fold01
> tail(fit$pred)
     pred obs rowIndex parameter Resample
1698    Y   Y     1592      none   Fold10
1699    Y   N     1594      none   Fold10
1700    N   N     1621      none   Fold10
1701    N   N     1656      none   Fold10
1702    N   N     1671      none   Fold10
1703    Y   Y     1689      none   Fold10
```

So, due to the randomness of the folds and because you are predicting 0 or 1 you get an accuracy of roughly 50%.

The in sample fitted values you are looking for are in `fit$finalModel$fitted.values`. Using those:

```
fitpred <- fit$finalModel$fitted.values
fitpredt <- function(t) ifelse(fitpred > t , 1,0)
> confusionMatrix(fitpredt(0.3),chile.v)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 773  44
         1  94 792

               Accuracy : 0.919
                 95% CI : (0.905, 0.9315)
    No Information Rate : 0.5091
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8381
 Mcnemar's Test P-Value : 3.031e-05

            Sensitivity : 0.8916
            Specificity : 0.9474
         Pos Pred Value : 0.9461
         Neg Pred Value : 0.8939
```

```
            Prevalence : 0.5091
        Detection Rate : 0.4539
  Detection Prevalence : 0.4797
     Balanced Accuracy : 0.9195

      'Positive' Class : 0
```

Now the accuracy is around the expected value. Setting the threshold to 0.5 yields about the same accuracy as the estimate from the 10-fold cross validation:

```
> confusionMatrix(fitpredt(0.5),chile.v)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 809  64
         1  58 772

               Accuracy : 0.9284
                 95% CI : (0.9151, 0.9402)
[rest of the output omitted]

> fit
Generalized Linear Model

1703 samples
   7 predictors
   2 classes: 'N', 'Y'

No pre-processing
Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 1533, 1532, 1532, 1533, 1532, 1533, ...

Resampling results

  Accuracy  Kappa  Accuracy SD  Kappa SD
  0.927     0.854  0.0134       0.0267
```

Additionally, regarding your expectation "that the cross validated results should not perform much worse than the first model" please check `summary(res.chileIII)` and `summary(fit)`. The fitted models and coefficients are exactly the same so they will give the same results.

PS: I know my answer to this question is late, i.e. this is quite an old question. Is it OK to answer these questions anyway? I am new here and did not find anything about "late answers" in the help.

edited Sep 24 '14 at 23:07

Unihedron
7,500 ● 10 ● 30 ● 52

answered Sep 24 '14 at 23:03

thie1e
901 ● 6 ● 10

Hi, I appreciate it. Well you never know who's reading what and when. I read quite a few "old posts" that helped me a lot. – Vincent Oct 14 '14 at 8:06

Agreed, I just found this useful, myself. Great answer! – pbnelson Dec 14 '14 at 20:00

WOW! This is quite helpful! – EsBee Apr 29 '15 at 19:29

The question and this answer combined is a succinct tutorial off how to use caret! Outstanding! – masfenix Jul 28 '15 at 2:28