Model Comparison Logistic Regression, Decision Tree, Random Forest

Merve Yavuz

Logistic Regression, Decision Tree, and Random Forest Model for Breast Cancer Prediction

In this study, we will examine the performance comparison of logistic regression, decision trees and random forest machine learning models for the breast cancer data set, which will classify the breast cancer risk according to the characteristics such as age, race, marital status, tumor size.

- Describe data set
- Training Logistic regression model
- Training Decision tree model
- Training Random Forest model
- Performance comparison of trained models
- Increase the model prediction performance

Packages

We loaded the *caret* library for the confusion matrix and included it in the project. The *rsample* library is used to use classes to create and summarize different types of resampling objects. The *rpart.plot* library is used to automatically adapt and plot the plot for the response type of the model. The *parsnip* is interface library designed to solve a specific problem with model fit in R. *ranger* is a fast implementation library of random forests or iterative partitioning, especially suitable for high dimensional data. The *yardstick* library was used to measure how well the model fits a data set, such as confusion matrices, class probability curve summaries, and regression measures.

```
#install.packages("caret")
#install.packages("ROSE")
#install.packages("caret")
#install.packages("ROSE")
#install.packages("parsnip")
#install.packages("ranger")
#install.packages("mlbench")
```

```
library(caret)
```

```
library(ROSE)
```

```
library(rsample)
```

```
library(rpart.plot)
```

```
library(parsnip)
```

```
library(tidyverse)
```

```
library(yardstick)
```

```
library(ranger)
```

Breast Cancer Data set

This data set for breast cancer patients comes from the November 2017 update of NCI's SEER Program, which provides information on population-based cancer statistics. The data set includes female patients with infiltrating duct and lobular carcinoma breast cancer diagnosed in 2006-2010. Patients with unknown tumor size, regional LNs examined, positive regional LNs, and survival months of less than 1 month were excluded from the data set, thus finally including 4024 patients. We download the data set we used in the study from Breast Cancer Dataset with .csv extension.

```
BreastCancerData<- read.csv("Breast_Cancer.csv")
```

We are clearing the NA data in the data set.

```
BreastCancerData<-na.exclude(BreastCancerData)
```

Column attributes

The columns information of our imported breast cancer data set is as follows.

- Age : Age of the patient (int)
- Race : Race of the patient (char) {White; Black ; Other]
- Maritial Status: Maritial Status of patient (char) {Married ; Divorced ; Single}
- T.Stage : (char) {T1 ; T2 ; T3 ; T4}
- N.Stage : (char) { N1 ; N2 ; N3}
- X6th.Stage: (char) {IIA ; IIIA ; IIB; IIIC}
- Differantiate: (char) {Poorly Differantiated ; Moderatly Differantiated ; Well Differantiated }
- Grade: Grade of patient (char) {1 ; 2; 3}
- A.Stage: (char) { Regional(A neoplasm that has extended); Distant (A neoplasm that has spread to parts of the body remote from)}
- Tumor.Size: Tumor size of patients (Each indicates exact size in millimeters) (int)
- Estrogen.Status : Estrogen hormone status of patients (char) { Positive ; Negative }
- Progesterone.Status: Progesteronehormone status of patients (char) { Positive ; Negative }
- Regional.Node.Examined: Number of Regional Nodes examined by patients (int)
- Regional.Node.Positive: Number of Regional Nodes positive for patients (int)
- Survival.Months : Patients' life expectancy in months (int)
- Status: Patients' status as a class feature (char) {Alive ; Dead}

We can look at the data set with the **str()** function. The function returns values containing the following information:

- 4024 observation
- 16 variables (features)

The types and some values of the features are shown in the figure below.

```
str(BreastCancerData)
```

```
'data.frame':   4024 obs. of  16 variables:
 $ Age                   : int  68 50 58 58 47 51 51 40 40 69 ...
 $ Race                  : chr  "White" "White" "White" "White" ...
 $ Marital.Status        : chr  "Married" "Married" "Divorced" "Married" ...
 $ T.Stage               : chr  "T1" "T2" "T3" "T1" ...
 $ N.Stage               : chr  "N1" "N2" "N3" "N1" ...
 $ X6th.Stage            : chr  "IIA" "IIIA" "IIIC" "IIA" ...
 $ differentiate         : chr  "Poorly differentiated" "Moderately differentiated" "Moderate
 $ Grade                 : chr  "3" "2" "2" "3" ...
 $ A.Stage               : chr  "Regional" "Regional" "Regional" "Regional" ...
 $ Tumor.Size            : int  4 35 63 18 41 20 8 30 103 32 ...
 $ Estrogen.Status       : chr  "Positive" "Positive" "Positive" "Positive" ...
 $ Progesterone.Status   : chr  "Positive" "Positive" "Positive" "Positive" ...
 $ Regional.Node.Examined: int  24 14 14 2 3 18 11 9 20 21 ...
 $ Reginol.Node.Positive : int  1 5 7 1 1 2 1 1 18 12 ...
 $ Survival.Months       : int  60 62 75 84 50 89 54 14 70 92 ...
 $ Status                : chr  "Alive" "Alive" "Alive" "Alive" ...
```

Factor conversion in columns

We need to transform the character-specific data in the breast cancer dataset into categorical features. We make this change with the *factor(columnname)* function.

```
BreastCancerData$Race<- factor(BreastCancerData$Race)
BreastCancerData$Marital.Status<- factor(BreastCancerData$Marital.Status)
BreastCancerData$T.Stage<- factor(BreastCancerData$T.Stage)
BreastCancerData$N.Stage<- factor(BreastCancerData$N.Stage)
BreastCancerData$X6th.Stage<- factor(BreastCancerData$N.Stage)
BreastCancerData$differentiate<- factor(BreastCancerData$differentiate)
BreastCancerData$Grade<- factor(BreastCancerData$Grade)
BreastCancerData$A.Stage<- factor(BreastCancerData$A.Stage)
BreastCancerData$Estrogen.Status<- factor(BreastCancerData$Estrogen.Status)
BreastCancerData$Progesterone.Status<- factor(BreastCancerData$Progesterone.Status)
```

Task 1: Predict Breast Cancer using Logistic Regression

In this section, we try to estimate the probability of individuals getting cancer using the breast cancer dataset to train the logistic regression model.

Step 1- Splitting the data set

At this stage, we convert the class data to 1 and 0 for logistic regression and make the type an integer.

```
BreastCancerData$Status<- ifelse(BreastCancerData$Status == 'Alive', '1', '0')
BreastCancerData$Status<- as.integer(BreastCancerData$Status)
```

We use the *sample()* function to split 80% of the data into the train and 20% into the test set and set the *seed()* to keep the same values for every future run.

```
set.seed(123)

index <-sample(1:nrow(BreastCancerData),round(nrow(BreastCancerData))*0.80)
traindata <-BreastCancerData[index,]
testdata <-BreastCancerData[-index,]
```

Here, randomly selected indexes are assigned to train and test sets.

Step 2- Train a logistic regression

We use the "glm()" function to train our logistic regression model. This function needs 3 parameters. The first is the formula of the model, the second is the data set used to train the model, and finally the family distribution of the target feature. We set it to "*binomial*" because we will produce binary outputs in family distribution logistic regression models. In the study, we defined the model formula as $y \sim$. Here "y" is the target variable representing the *Status* property to be estimated in our data set. In addition, with this definition, we trained our model according to all variables

```
BreastcancerModel<-glm(Status~.,data=traindata,family = "binomial")
```

With the *summary()* function, we can see detailed information about our logistic regression model.

```
summary(BreastcancerModel)
```

```
Call:
glm(formula = Status ~ ., family = "binomial", data = traindata)

Deviance Residuals:
```

```
    Min       1Q   Median       3Q      Max
-3.3042   0.1356   0.2630   0.4645   2.3078

Coefficients: (5 not defined because of singularities)
                                   Estimate Std. Error z value Pr(>|z|)
(Intercept)                       -1.336453   0.654331  -2.042 0.041105 *
Age                               -0.030790   0.007371  -4.177 2.95e-05 ***
RaceOther                          0.907500   0.318588   2.849 0.004392 **
RaceWhite                          0.523049   0.206978   2.527 0.011502 *
Marital.StatusMarried              0.273521   0.185107   1.478 0.139505
Marital.StatusSeparated           -0.594978   0.533067  -1.116 0.264361
Marital.StatusSingle               0.204478   0.229203   0.892 0.372326
Marital.StatusWidowed              0.122028   0.286436   0.426 0.670092
T.StageT2                         -0.510554   0.169238  -3.017 0.002555 **
T.StageT3                         -0.706854   0.348632  -2.028 0.042611 *
T.StageT4                         -1.593291   0.421406  -3.781 0.000156 ***
N.StageN2                         -0.250872   0.168619  -1.488 0.136802
N.StageN3                         -0.243822   0.319912  -0.762 0.445968
X6th.StageN2                            NA         NA      NA       NA
X6th.StageN3                            NA         NA      NA       NA
differentiatePoorly differentiated -0.374642   0.137985  -2.715 0.006626 **
differentiateUndifferentiated     -1.730804   0.823694  -2.101 0.035617 *
differentiateWell differentiated    0.712999   0.237945   2.996 0.002731 **
Grade1                                  NA         NA      NA       NA
Grade2                                  NA         NA      NA       NA
Grade3                                  NA         NA      NA       NA
A.StageRegional                   -0.224378   0.361407  -0.621 0.534701
Tumor.Size                         0.003039   0.005259   0.578 0.563311
Estrogen.StatusPositive            0.420084   0.256644   1.637 0.101665
Progesterone.StatusPositive        0.531346   0.171208   3.104 0.001912 **
Regional.Node.Examined             0.028959   0.008823   3.282 0.001030 **
Reginol.Node.Positive             -0.086203   0.020233  -4.261 2.04e-05 ***
Survival.Months                    0.062850   0.003133  20.061  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2780.2  on 3218  degrees of freedom
Residual deviance: 1791.2  on 3196  degrees of freedom
AIC: 1837.2

Number of Fisher Scoring iterations: 6
```

Step 3-The Performance of logistic regression trained model

Checking the performance of the model with the test data set is necessary for the model to have good generalization. For this, we first try to estimate the *Status* values in the test set. When calculating this, we need to exclude the *Status* property from the test set.

```
predicted_hmodel <- predict(BreastcancerModel, testdata[,-16], type = "response")
```

```
Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
prediction from a rank-deficient fit may be misleading
```

```
head(predicted_hmodel)
```

```
        3         6        14        22        43        47
0.8681136 0.9895550 0.6411238 0.7187249 0.9552715 0.6080710
```

predicted_hmodel is a vector that calculates the probability of an individual's cancer risk. We convert these values into classes.

```
predicted_output <- ifelse(predicted_hmodel > 0.5, 1, 0)
```

```
head(predicted_output)
```

```
 3  6 14 22 43 47
 1  1  1  1  1  1
```

To calculate metrics based on the confusion matrix, assume that individuals with an output value of 1 (with breast cancer status) are in the positive class and individuals with 0 (dead with breast cancer status) are in the negative class.

```
lrConf<-confusionMatrix(table(ifelse(testdata$Status == '1', '1', '0'),
                      predicted_output),
                positive = '1' )
```

```
lrConf
```

```
Confusion Matrix and Statistics

   predicted_output
      0   1
  0  52  64
  1  22 667

               Accuracy : 0.8932
                 95% CI : (0.8697, 0.9137)
    No Information Rate : 0.9081
    P-Value [Acc > NIR] : 0.9338

                  Kappa : 0.4901

 Mcnemar's Test P-Value : 9.818e-06

            Sensitivity : 0.9124
            Specificity : 0.7027
         Pos Pred Value : 0.9681
         Neg Pred Value : 0.4483
             Prevalence : 0.9081
         Detection Rate : 0.8286
   Detection Prevalence : 0.8559
      Balanced Accuracy : 0.8076

       'Positive' Class : 1
```

We can use the confusion matrix to see performance metrics (accuracy.precision, recall and others) in detail and to see correct and incorrectly classified values.

Task 2: Predict Breast Cancer using Decision Tree

In this section, we try to estimate the cancer classes of individuals using the breast cancer data set to train the decision trees model.

```r
BreastCancerData<- read.csv("Breast_Cancer.csv")
#Kategorik verileri factor olarak düzenliyoruz.
BreastCancerData$Race<- factor(BreastCancerData$Race)
BreastCancerData$Marital.Status<- factor(BreastCancerData$Marital.Status)
BreastCancerData$T.Stage<- factor(BreastCancerData$T.Stage)
```

```
BreastCancerData$N.Stage<- factor(BreastCancerData$N.Stage)
BreastCancerData$X6th.Stage<- factor(BreastCancerData$N.Stage)
BreastCancerData$differentiate<- factor(BreastCancerData$differentiate)
BreastCancerData$Grade<- factor(BreastCancerData$Grade)
BreastCancerData$A.Stage<- factor(BreastCancerData$A.Stage)
BreastCancerData$Estrogen.Status<- factor(BreastCancerData$Estrogen.Status)
BreastCancerData$Progesterone.Status<- factor(BreastCancerData$Progesterone.Status)
BreastCancerData$Status<- factor(BreastCancerData$Status)
```

Step 1- Splitting the data set

With the *initial_split()* function, we separate our dataset as 80% training 20% test set. We assign train data to *breast_train* and test data to *breast_test*.

```
breast_split <- initial_split(data = BreastCancerData,
                              prop = 0.80)

breast_train <- breast_split |> training()
breast_test  <- breast_split |> testing()
```

Step 2- Train a decision tree

We set the decision tree to classification mode.

```
dt_model <- decision_tree() |>  set_engine("rpart") |> set_mode("classification")
```

We train the decision tree with the breast_train training data set. We set our target variable as *Status*.

```
dt_breast <- dt_model |>
  fit(Status ~., data = breast_train)
```

We display the trained data model.

```
dt_breast
```

```
parsnip model object

n= 3219
```

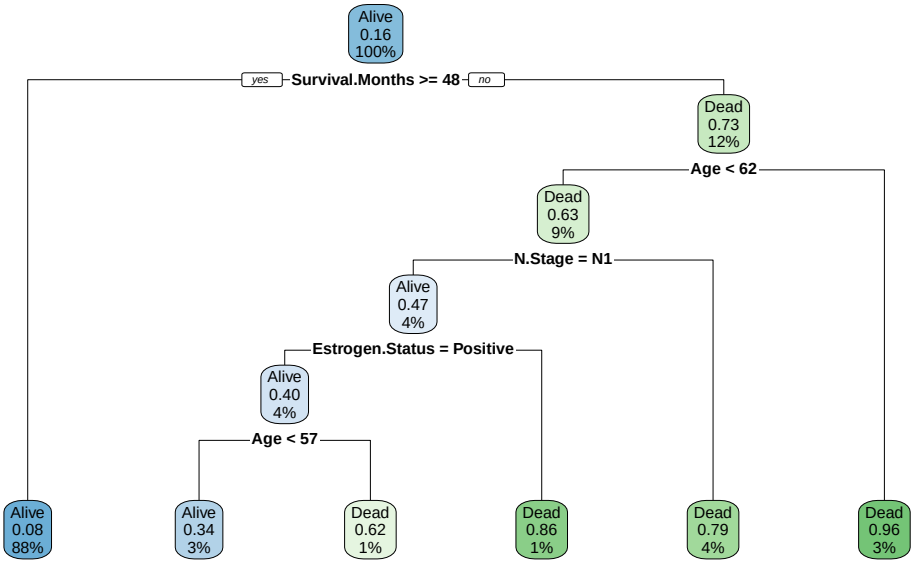
```
node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 3219 511 Alive (0.84125505 0.15874495)
   2) Survival.Months>=47.5 2830 228 Alive (0.91943463 0.08056537) *
   3) Survival.Months< 47.5 389 106 Dead (0.27249357 0.72750643)
     6) Age< 61.5 278 102 Dead (0.36690647 0.63309353)
      12) N.Stage=N1 137  64 Alive (0.53284672 0.46715328)
        24) Estrogen.Status=Positive 116  46 Alive (0.60344828 0.39655172)
          48) Age< 56.5 92  31 Alive (0.66304348 0.33695652) *
          49) Age>=56.5 24   9 Dead (0.37500000 0.62500000) *
        25) Estrogen.Status=Negative 21   3 Dead (0.14285714 0.85714286) *
      13) N.Stage=N2,N3 141  29 Dead (0.20567376 0.79432624) *
     7) Age>=61.5 111   4 Dead (0.03603604 0.96396396) *
```

We show a visual of the decision tree classified by variables.

```
rpart.plot(dt_breast$fit)
```

```
Warning: Cannot retrieve the data used to build the model (so cannot determine roundint and i
To silence this warning:
    Call rpart.plot with roundint=FALSE,
    or rebuild the rpart model with model=TRUE.
```



Step 3-The Performance of decision tree trained model

We make predictions on the breast test data of the decision tree.

```
breast_predictions <- dt_breast |>
  predict(new_data = breast_test)
```

We list predicted values.

```
breast_predictions
```

```
# A tibble: 805 x 1
   .pred_class
   <fct>
 1 Alive
 2 Alive
 3 Alive
 4 Alive
 5 Alive
 6 Alive
 7 Alive
```

```
 8 Alive
 9 Dead
10 Alive
# i 795 more rows
```

Its functions require a data.frame or tibble containing the model results. We combine the model prediction with the actual/observed values of the target variable in the test data.

```
breast_results <- tibble(predicted = breast_predictions$.pred_class,
                         actual    = breast_test$Status)
```

We create the confusion matrix with the *conf_mat()* function. Here, truth shows the actual class values, and estimate shows the predicted class value.

```
breast_results |> conf_mat(truth    = actual,
                           estimate = predicted)
```

```
          Truth
Prediction Alive Dead
     Alive   688   55
     Dead     12   50
```

The accuracy metric performance is calculated with the accuracy() function.

```
accuracy<-breast_results |> accuracy(truth = actual, estimate = predicted)
accuracy
```

```
# A tibble: 1 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.917
```

The sensivity metric performance is calculated with the sens() function.

```
sensivity<-breast_results |> sens(truth = actual, estimate = predicted)
sensivity
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 sens    binary         0.983
```

The precision metric performance is calculated with the precision() function.

```
precision<-breast_results |> precision(truth = actual, estimate = predicted)
precision
```

```
# A tibble: 1 x 3
  .metric   .estimator .estimate
  <chr>     <chr>          <dbl>
1 precision binary         0.926
```

The recall metric performance is calculated with the recall() function.

```
recall<-breast_results |> recall(truth = actual, estimate = predicted)
recall
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 recall  binary         0.983
```

The f-measure metric performance is calculated with the f_meas() function.

```
fmeasure<-breast_results |> f_meas(truth = actual, estimate = predicted)
fmeasure
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 f_meas  binary         0.954
```

Task 3: Predict Breast Cancer using Random Forest Tree

In this section, we try to estimate the cancer classes of individuals using the breast cancer data set to train the random forest model.

Step 1- Splitting the data set

With the *initial_split()* function, we separate our dataset as 80% training 20% test set. We assign train data to *breast_train_rf* and test data to *breast_test_rf*.

```r
breast_split_rf <- initial_split(data = BreastCancerData, # dataset to split
                                 prop = 0.80)    # proportion of train set

breast_train_rf <- breast_split |> training()
breast_test_rf  <- breast_split |> testing()
```

Step 2- Train a Random Forest

We train the random forest with the breast_train_rf training data set. We set our target variable as *Status*.

```r
trained_rf <- ranger(Status ~ .,
                     data = breast_train_rf)
```

Let's see the output of the trained_rf model :

```r
trained_rf
```

```
Ranger result

Call:
 ranger(Status ~ ., data = breast_train_rf)

Type:                             Classification
Number of trees:                  500
Sample size:                      3219
Number of independent variables:  15
Mtry:                             3
Target node size:                 1
Variable importance mode:         none
Splitrule:                        gini
OOB prediction error:             9.91 %
```

Step 3-The Performance of random forest trained model

We make predictions on the breast_test_rf data of the random forest model.

```r
preds_rf <- predict(trained_rf, breast_test_rf)
```

Let's see the output of the preds_rf model

```r
preds_rf
```

```
Ranger prediction

Type:                             Classification
Sample size:                      805
Number of independent variables:  15
```

We can use the confusion matrix to see performance metrics (accuracy.precision, recall and others) in detail and to see correct and incorrectly classified values for random forest.

```r
rfConfMat<-confusionMatrix(preds_rf$predictions,
                breast_test$Status,
                positive = "Alive")
```

```r
rfConfMat
```

```
Confusion Matrix and Statistics

          Reference
Prediction Alive Dead
     Alive   690   49
     Dead     10   56

               Accuracy : 0.9267
                 95% CI : (0.9065, 0.9437)
    No Information Rate : 0.8696
    P-Value [Acc > NIR] : 1.574e-07

                  Kappa : 0.6163

 Mcnemar's Test P-Value : 7.530e-07

            Sensitivity : 0.9857
            Specificity : 0.5333
         Pos Pred Value : 0.9337
         Neg Pred Value : 0.8485
             Prevalence : 0.8696
         Detection Rate : 0.8571
   Detection Prevalence : 0.9180
```

```
      Balanced Accuracy : 0.7595

       'Positive' Class : Alive
```

Performance Comparison of Trained Models

The performance outputs of the logistic regression, decision trees and random forest methods trained and tested with the breast cancer data set are given below.

Logistic Regression:

```
lrConf
```

```
Confusion Matrix and Statistics

   predicted_output
      0   1
  0  52  64
  1  22 667

               Accuracy : 0.8932
                 95% CI : (0.8697, 0.9137)
    No Information Rate : 0.9081
    P-Value [Acc > NIR] : 0.9338

                  Kappa : 0.4901

 Mcnemar's Test P-Value : 9.818e-06

            Sensitivity : 0.9124
            Specificity : 0.7027
         Pos Pred Value : 0.9681
         Neg Pred Value : 0.4483
             Prevalence : 0.9081
         Detection Rate : 0.8286
   Detection Prevalence : 0.8559
      Balanced Accuracy : 0.8076

       'Positive' Class : 1
```

Decision Tree:

```
accuracy
```

```
# A tibble: 1 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.917
```

```
sensivity
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 sens    binary         0.983
```

```
precision
```

```
# A tibble: 1 x 3
  .metric   .estimator .estimate
  <chr>     <chr>          <dbl>
1 precision binary         0.926
```

```
recall
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 recall  binary         0.983
```

Random Forest:

```
rfConfMat
```

```
Confusion Matrix and Statistics

          Reference
Prediction Alive Dead
     Alive   690   49
     Dead     10   56

               Accuracy : 0.9267
                 95% CI : (0.9065, 0.9437)
    No Information Rate : 0.8696
    P-Value [Acc > NIR] : 1.574e-07

                  Kappa : 0.6163

 Mcnemar's Test P-Value : 7.530e-07

            Sensitivity : 0.9857
            Specificity : 0.5333
         Pos Pred Value : 0.9337
         Neg Pred Value : 0.8485
             Prevalence : 0.8696
         Detection Rate : 0.8571
   Detection Prevalence : 0.9180
      Balanced Accuracy : 0.7595

       'Positive' Class : Alive
```

Considering the accuracy value of all 3 models, **Logistic regression < Decision trees < Random forest,** respectively, according to the best.

Considering the sensivity value of all 3 models, **Logistic regression < Decision trees < Random forest**

respectively, are sorted according to the best.

Increase the Model Prediction Performance.

In this section, various methods are used to improve the prediction performance of logistic regression, decision trees and random forest models.

Cross Validation Method

Cross-validation is a re-sampling method that uses different portions of the data to test and train a model on different iterations. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

```
set.seed(123)
control<-trainControl(method = "cv",
                      number = 10)
```

```
bmodel<-train(as.factor(Status)~.,
              data = BreastCancerData,
              trControl=control,
              method="glm")
```

```
bmodel$resample
```

```
    Accuracy     Kappa Resample
1  0.8905473 0.4581214   Fold01
2  0.8957816 0.5089928   Fold02
3  0.8957816 0.5243902   Fold03
4  0.9007444 0.5167286   Fold04
5  0.9156328 0.6209262   Fold05
6  0.8728180 0.4073033   Fold06
7  0.9029851 0.5678849   Fold07
8  0.8905473 0.4581214   Fold08
9  0.9054726 0.5695446   Fold09
10 0.8808933 0.4729730   Fold10
```

```
bmodel
```

```
Generalized Linear Model

4024 samples
  15 predictor
   2 classes: 'Alive', 'Dead'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3622, 3621, 3621, 3621, 3621, 3623, ...
```

```
Resampling results:

  Accuracy   Kappa
  0.8951204  0.5104987
```