Session13_predictivemodels

Jeffrey Wijaya

2023-06-09

attach data

```
library(MASS)
library(mlbench)
```

```
## Warning: package 'mlbench' was built under R version 4.2.3
```

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.2.3
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(MLmetrics)
```

```
## Warning: package 'MLmetrics' was built under R version 4.2.3
##
## Attaching package: 'MLmetrics'
## The following object is masked from 'package:base':
##
##     Recall
```

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

```
## Warning: package 'rpart.plot' was built under R version 4.2.3
```

```
data("Boston")
BasicSummary <- function(df, dgts = 3){
m <- ncol(df)
varNames <- colnames(df)
varType <- vector("character",m)
topLevel <- vector("character",m)
topCount <- vector("numeric",m)
missCount <- vector("numeric",m)
levels <- vector("numeric", m)
for (i in 1:m){
x <- df[,i]
```

```r
varType[i] <- class(x)
xtab <- table(x, useNA = "ifany")
levels[i] <- length(xtab)
nums <- as.numeric(xtab)
maxnum <- max(nums)
topCount[i] <- maxnum
maxIndex <- which.max(nums)
lvls <- names(xtab)
topLevel[i] <- lvls[maxIndex]
missIndex <- which((is.na(x)) | (x == "") | (x == " "))
missCount[i] <- length(missIndex)
}
n <- nrow(df)
topFrac <- round(topCount/n, digits = dgts)
missFrac <- round(missCount/n, digits = dgts)
summaryFrame <- data.frame(variable = varNames, type = varType,
levels = levels, topLevel = topLevel,
topCount = topCount, topFrac = topFrac,
missFreq = missCount, missFrac = missFrac)
return(summaryFrame)
}
```

trasnform

```r
df2 <- ifelse(Boston$medv <= 21, "Low", "High")
df <- ifelse(Boston$medv <= 21, 0, 1)
Boston$medv <- df
BasicSummary(Boston)
```

```
##    variable    type levels topLevel topCount topFrac missFreq missFrac
## 1      crim numeric    504  0.01501        2   0.004        0        0
## 2        zn numeric     26        0      372   0.735        0        0
## 3     indus numeric     76     18.1      132   0.261        0        0
## 4      chas integer      2        0      471   0.931        0        0
## 5       nox numeric     81    0.538       23   0.045        0        0
## 6        rm numeric    446    5.713        3   0.006        0        0
## 7       age numeric    356      100       43   0.085        0        0
## 8       dis numeric    412   3.4952        5   0.010        0        0
## 9       rad integer      9       24      132   0.261        0        0
## 10      tax numeric     66      666      132   0.261        0        0
## 11  ptratio numeric     46     20.2      140   0.277        0        0
```

```
## 12    black numeric    357    396.9      121    0.239        0         0
## 13    lstat numeric    455     6.36        3    0.006        0         0
## 14     medv numeric      2        1      257    0.508        0         0
```

```
keepVars <- c("crim","indus","nox","rm","age","dis","rad","tax","ptratio","black","lstat","medv")
BostonSub <- Boston[, keepVars]
BasicSummary(BostonSub)
```

```
##    variable    type levels topLevel topCount topFrac missFreq missFrac
## 1      crim numeric    504  0.01501        2   0.004        0        0
## 2     indus numeric     76     18.1      132   0.261        0        0
## 3       nox numeric     81    0.538       23   0.045        0        0
## 4        rm numeric    446    5.713        3   0.006        0        0
## 5       age numeric    356      100       43   0.085        0        0
## 6       dis numeric    412   3.4952        5   0.010        0        0
## 7       rad integer      9       24      132   0.261        0        0
## 8       tax numeric     66      666      132   0.261        0        0
## 9   ptratio numeric     46     20.2      140   0.277        0        0
## 10    black numeric    357    396.9      121   0.239        0        0
## 11    lstat numeric    455     6.36        3   0.006        0        0
## 12     medv numeric      2        1      257   0.508        0        0
```

Split the dataset into training set and validation set

```
set.seed(123)
n <- nrow(BostonSub)
train <-sample(n,round(0.7 *n))
BostonTrain <- BostonSub[train,]
BostonValidation <- BostonSub[-train, ]
```

Fit the model using full set of variables : Logistic Regression

```
logisticfull <- glm(medv ~., family = "binomial", data= BostonTrain)
summary(logisticfull)
```

```
##
## Call:
## glm(formula = medv ~ ., family = "binomial", data = BostonTrain)
##
## Deviance Residuals:
##      Min         1Q     Median         3Q        Max
```

```
## -2.00324  -0.33153   0.01599   0.29587   2.81677
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 20.349284   4.923650   4.133 3.58e-05 ***
## crim        -0.078817   0.077176  -1.021  0.30713
## indus       -0.011857   0.058327  -0.203  0.83891
## nox         -6.021902   3.098806  -1.943  0.05198 .
## rm           1.112287   0.467236   2.381  0.01729 *
## age         -0.034728   0.013593  -2.555  0.01062 *
## dis         -0.699297   0.168508  -4.150 3.33e-05 ***
## rad          0.320370   0.078556   4.078 4.54e-05 ***
## tax         -0.012035   0.003720  -3.235  0.00121 **
## ptratio     -0.724512   0.137580  -5.266 1.39e-07 ***
## black        0.002531   0.003541   0.715  0.47480
## lstat       -0.326749   0.061620  -5.303 1.14e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 490.34  on 353  degrees of freedom
## Residual deviance: 186.02  on 342  degrees of freedom
## AIC: 210.02
##
## Number of Fisher Scoring iterations: 7
```

Make model 2 with variable that have a small p-value

```
logisticRef <- glm(medv~ dis + rad + ptratio + lstat, data = BostonTrain, family = "binomial")
summary(logisticRef)
```

```
##
## Call:
## glm(formula = medv ~ dis + rad + ptratio + lstat, family = "binomial",
##     data = BostonTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1938  -0.4136   0.0835   0.4678   3.8384
##
```

```
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 15.18270    2.17235   6.989 2.77e-12 ***
## dis         -0.18155    0.09162  -1.982   0.0475 *
## rad          0.02253    0.02511   0.897   0.3697
## ptratio     -0.51940    0.10575  -4.912 9.02e-07 ***
## lstat       -0.41714    0.05202  -8.019 1.06e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 490.34  on 353  degrees of freedom
## Residual deviance: 233.18  on 349  degrees of freedom
## AIC: 243.18
##
## Number of Fisher Scoring iterations: 6
```

Model Validation

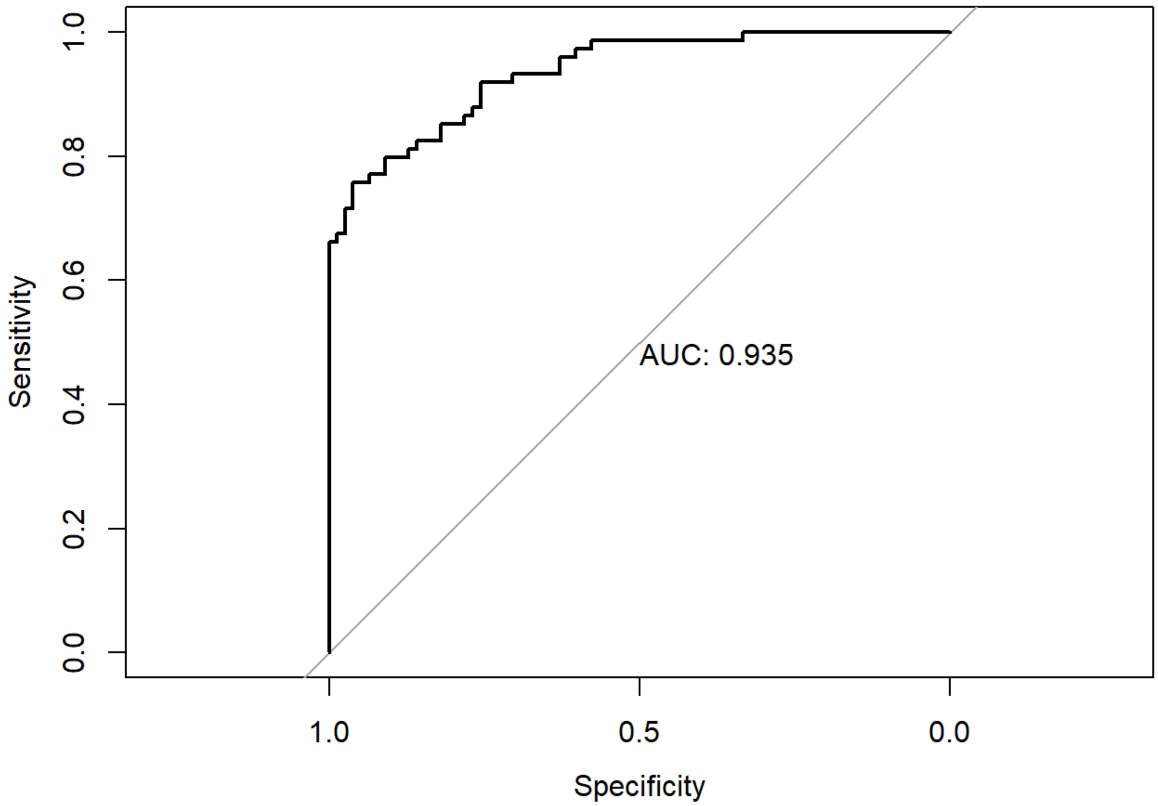
```
phatFullV<-predict(logisticfull, newdata = BostonValidation, type = "response")
phatRefV<-predict(logisticRef, newdata = BostonValidation, type = "response")
```

Model Performance

ROC Curve and AUC

```
ROCFull<- roc(BostonValidation$medv, phatFullV, plot = TRUE, print.auc = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

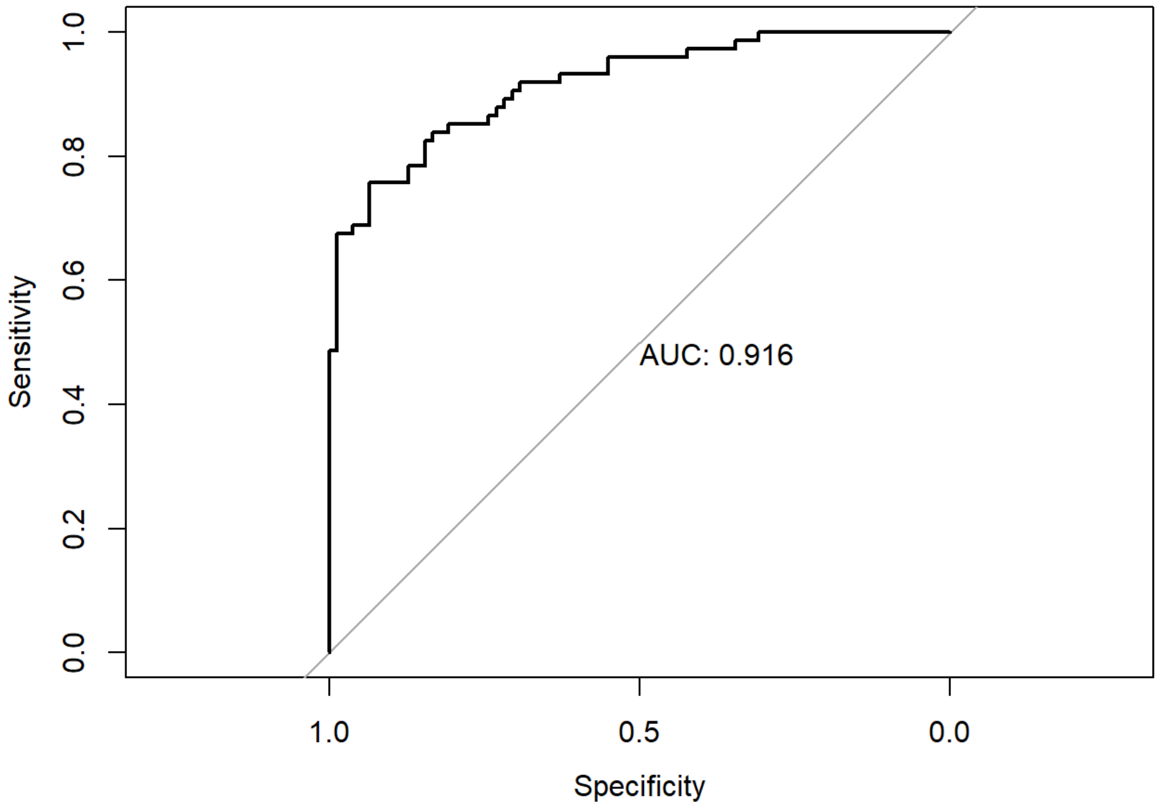



```
ROCFull
```

```
##
## Call:
## roc.default(response = BostonValidation$medv, predictor = phatFullV,     plot = TRUE, print.au
c = TRUE)
##
## Data: phatFullV in 78 controls (BostonValidation$medv 0) < 74 cases (BostonValidation$medv 1).
## Area under the curve: 0.9354
```

```
ROCRef<- roc(BostonValidation$medv, phatRefV, plot = TRUE, print.auc = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```




```
ROCRef
##
## Call:
## roc.default(response = BostonValidation$medv, predictor = phatRefV,     plot = TRUE, print.auc = TRUE)
##
## Data: phatRefV in 78 controls (BostonValidation$medv 0) < 74 cases (BostonValidation$medv 1).
## Area under the curve: 0.916
```

Confusion Matrix Accuracy

```
threshold1 <- table(BostonValidation$medv, phatFullV > 0.5)
threshold2 <- table(BostonValidation$medv, phatRefV > 0.5)
accuracy1 <- round(sum(diag(threshold1)) / sum(threshold1), 2)
accuracy2 <- round(sum(diag(threshold2)) / sum(threshold2), 2)
threshold1
##
##      FALSE TRUE
##    0    64   14
##    1    11   63
sprintf("Accuracy is %s", accuracy1)
## [1] "Accuracy is 0.84"
```

```
threshold2
##
##      FALSE TRUE
##    0    65   13
##    1    12   62
sprintf("Accuracy is %s", accuracy2)
## [1] "Accuracy is 0.84"
```

Conclusion of logistic regression model

Model 1 was created using all the variables and we see that there is many large p-value, so we make Model 2 using all small p-value and when we test the accuracy it is the same while using all variables. When we compare the accuracy it is the same as the other.

Because all the two models accuracy result are the same then we can take any model result to compare with decision tree model to see which one has better accuracy. Final accuracy in Logistic Regression : 84%

```
Boston$medv <- df2
BasicSummary(Boston)
##    variable      type levels topLevel topCount topFrac missFreq missFrac
## 1      crim   numeric    504  0.01501        2   0.004        0        0
## 2        zn   numeric     26        0      372   0.735        0        0
## 3     indus   numeric     76     18.1      132   0.261        0        0
## 4      chas   integer      2        0      471   0.931        0        0
## 5       nox   numeric     81    0.538       23   0.045        0        0
## 6        rm   numeric    446    5.713        3   0.006        0        0
## 7       age   numeric    356      100       43   0.085        0        0
## 8       dis   numeric    412   3.4952        5   0.010        0        0
## 9       rad   integer      9       24      132   0.261        0        0
## 10      tax   numeric     66      666      132   0.261        0        0
## 11  ptratio   numeric     46     20.2      140   0.277        0        0
## 12    black   numeric    357    396.9      121   0.239        0        0
## 13    lstat   numeric    455     6.36        3   0.006        0        0
## 14     medv character      2     High      257   0.508        0        0
set.seed(123)
n <- nrow(Boston)
train <-sample(n,round(0.7 *n))
BostonTrain <- Boston[train,]
BostonValidation <- Boston[-train, ]
```

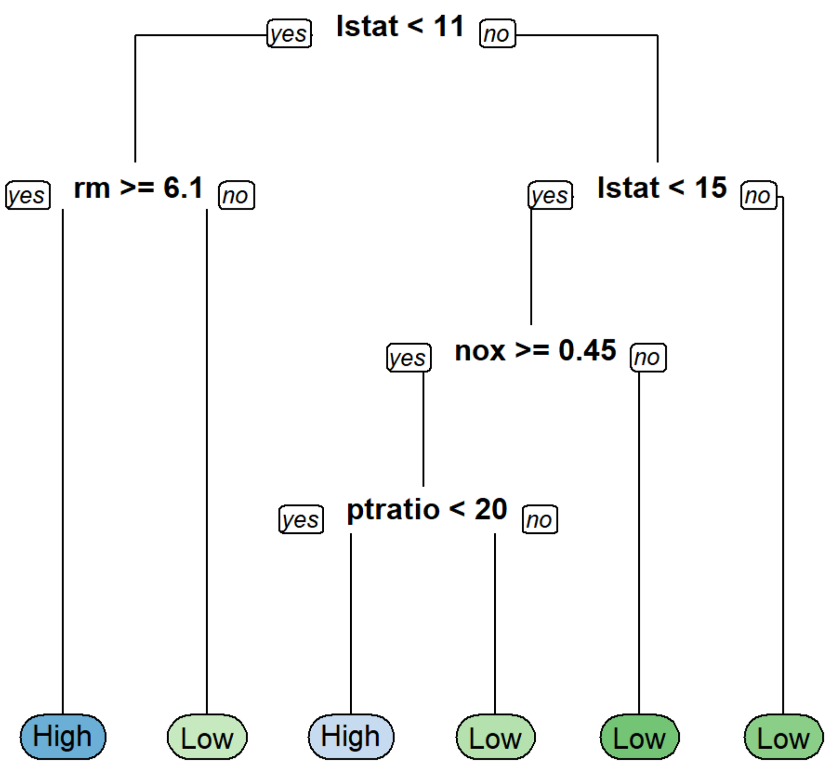
Model with decision Tree with all variables

```
Model1 <- rpart(formula = medv ~.,
```

```
                   data = BostonTrain,
                   method = "class")
Model1$variable.importance
```

```
##     lstat        rm       nox       age     indus       dis   ptratio      crim
## 88.200960 61.049274 50.220325 49.841110 47.486241 45.505345  6.362580  5.850207
##       rad       tax        zn
##  3.045725  2.280022  1.531406
```

```
rpart.plot(x = Model1, yesno=2, type=0, extra = 0)
```

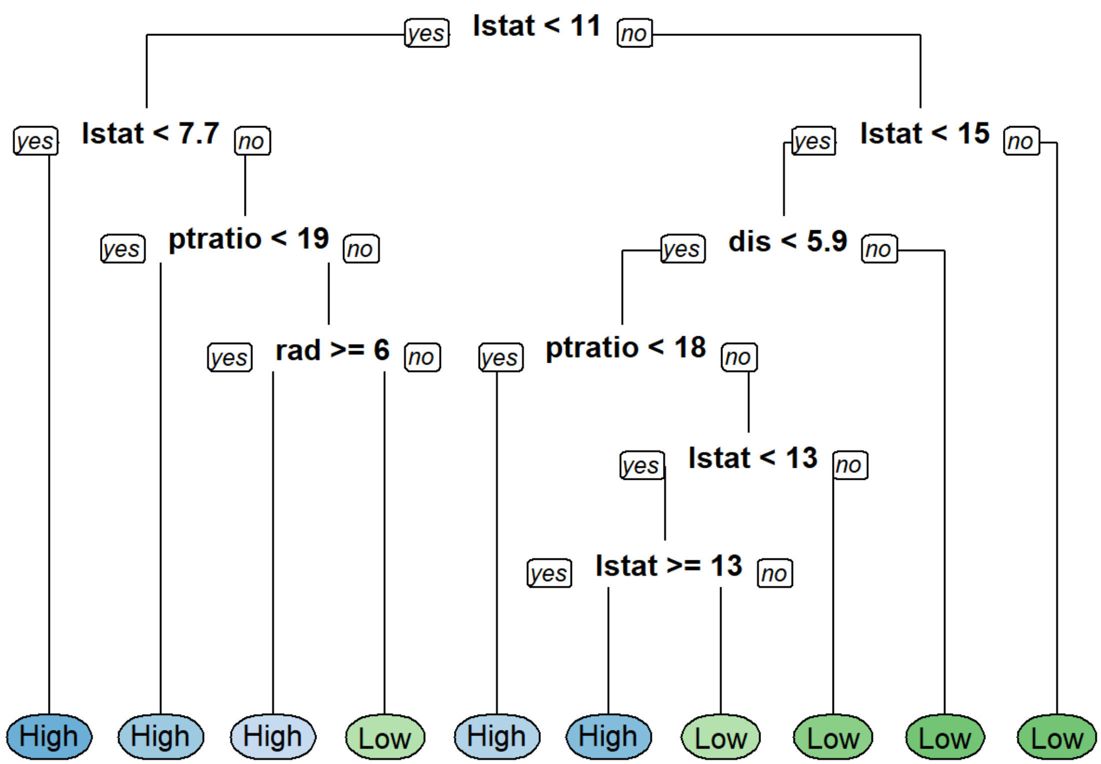


Model with decision tree with variables that importance is more than 40

```
Model2 <- rpart(formula = medv ~ dis + rad + ptratio + lstat,
                   data = BostonTrain,
                   method = "class")
Model2$variable.importance
```

```
##     lstat   ptratio       dis       rad
## 105.32372  50.08433  48.36459  37.32552
```

```
rpart.plot(x = Model2, yesno=2, type=0, extra = 0)
```




```
PreModel1 <- predict(Model1, newdata = BostonValidation, type = "class")
PreModel2 <- predict(Model2, newdata = BostonValidation, type = "class")
```

Confusion Matrix for first model

```
# Built confusion matrix with a threshold of 0.5
cm <- table(PreModel1, BostonValidation$medv)
cm
##
## PreModel1 High Low
##      High   58  11
##      Low    16  67
accuracy <- sum(cm[1], cm[4]) / sum(cm[1:4])
precision <- cm[4] / sum(cm[4], cm[2])
sensitivity <- cm[4] / sum(cm[4], cm[3])
fscore <- (2 * (sensitivity*precision)) / (sensitivity+precision)
specificity <- cm[1] / sum(cm[1], cm[2])
sprintf("Accuracy is %s", round(accuracy, 3))
## [1] "Accuracy is 0.822"
```

Confusion Matrix for second model

```r
# Built confusion matrix with a threshold of 0.5
cm <- table(PreModel2, BostonValidation$medv)
cm
```

```
##
## PreModel2 High Low
##      High   60  10
##      Low    14  68
```

```r
accuracy <- sum(cm[1], cm[4]) / sum(cm[1:4])
precision <- cm[4] / sum(cm[4], cm[2])
sensitivity <- cm[4] / sum(cm[4], cm[3])
fscore <- (2 * (sensitivity*precision)) / (sensitivity+precision)
specificity <- cm[1] / sum(cm[1], cm[2])
sprintf("Accuracy is %s", round(accuracy, 3))
```

```
## [1] "Accuracy is 0.842"
```

Conclusion of decision tree model

Model 1 was created using all the variables and we see that there are some variables that have small importance so we make Model 2. When we comparing the result the accuracy in Model 2 is higher than Model 1. So, we will take Model 2 as our final model to be compared to logistic regression model. Final Decision Tree Model accuracy: 84.2%

Conclusion of all

After we compare between logistic regression model and decision tree model, the result is logistic regression model is better because it’s accuracy is greater than decision tree model accuracy. So we will take any model from logistic regression and the final model that we choose is logisticfull. The reason why we should choose logisticfull is because from the ROC Curve plot we can see the AUC value in Model 1 is higher than logisticref.The higher the AUC score, the better the model is able to classify observations into classes. All of the models in logistic regression has AUC value higher than 90%, so all of them is a good model. But we take the highest AUC value, which is logisticfull with 0.938. So the final model we will use to predict future medv is : logisticfull from logistic regression model