**R Notebook**

**Question 10.1**

Using the same crime data setuscrime.txtas in Questions8.2 and 9.1, find the best model you can using (a) a regression tree model, and (b) a random forest model. In R, you can use the treepackage or the rpartpackage, and the randomForestpackage.For each model, describe one or two qualitative takeaways you get from analyzing the results (i.e., don’t just stop when you have a good model, but interpret it too)

**we determine a good regression, in the previous lesson was the following:**

pred<-lm(Crime~M+Ed+Po1+U2+Ineq+Prob,data=dat)

**references:** [**https://www.statmethods.net/advstats/cart.html**](https://www.statmethods.net/advstats/cart.html)

**to see how to use the fit to predict** [**https://stats.stackexchange.com/questions/64551/how-to-use-rparts-result-in-prediction**](https://stats.stackexchange.com/questions/64551/how-to-use-rparts-result-in-prediction)

**metadata about file** [**https://www.rdocumentation.org/packages/MASS/versions/7.3-51.4/topics/UScrime**](https://www.rdocumentation.org/packages/MASS/versions/7.3-51.4/topics/UScrime)

[**https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf**](https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf)

The method splits on

**rm**(list = **ls**())  
**set.seed**(1)  
  
**setwd**("c:\\stuff")  
uscrime=**read.table**("uscrime.txt",header=T)  
**dim**(uscrime)

## [1] 47 16

**library**(rpart)  
train\_index=**sample**(47,35,replace=FALSE)  
train\_set=uscrime[train\_index,]  
test\_set=uscrime[**-**train\_index,]  
  
fit=**rpart**(Crime**~**M**+**Ed**+**Po1**+**U2**+**Ineq**+**Prob,data=train\_set)  
  
*#plot(fit)*  
*#text(fit, use.n = TRUE)*  
**summary**(fit)

## Call:  
## rpart(formula = Crime ~ M + Ed + Po1 + U2 + Ineq + Prob, data = train\_set)  
## n= 35   
##   
## CP nsplit rel error xerror xstd  
## 1 0.290475 0 1.000000 1.0223720 0.3937728  
## 2 0.010000 1 0.709525 0.8938774 0.2951910  
##   
## Variable importance  
## Prob Po1 Ed Ineq M U2   
## 31 22 16 14 10 8   
##   
## Node number 1: 35 observations, complexity param=0.290475  
## mean=877.1714, MSE=95255.63   
## left son=2 (19 obs) right son=3 (16 obs)  
## Primary splits:  
## Prob < 0.042399 to the right, improve=0.29047500, (0 missing)  
## Po1 < 7.65 to the left, improve=0.23050830, (0 missing)  
## Ed < 11.05 to the left, improve=0.14694500, (0 missing)  
## Ineq < 23.5 to the right, improve=0.06840644, (0 missing)  
## U2 < 3.55 to the left, improve=0.05542243, (0 missing)  
## Surrogate splits:  
## Po1 < 7.15 to the left, agree=0.857, adj=0.687, (0 split)  
## Ed < 10.6 to the left, agree=0.771, adj=0.500, (0 split)  
## Ineq < 19.55 to the right, agree=0.743, adj=0.438, (0 split)  
## M < 13.25 to the right, agree=0.686, adj=0.313, (0 split)  
## U2 < 2.85 to the left, agree=0.657, adj=0.250, (0 split)  
##   
## Node number 2: 19 observations  
## mean=724.5263, MSE=25382.04   
##   
## Node number 3: 16 observations  
## mean=1058.438, MSE=117703.7

pr=**predict**(fit, test\_set[,**c**("Crime","M","Ed","Po1","U2","Ineq","Prob")])  
**table**(pr,test\_set[,'Crime'])

##   
## pr 342 373 439 511 539 946 1043 1151 1272 1555 1674 1993  
## 724.526315789474 0 0 1 1 1 1 0 0 0 0 0 0  
## 1058.4375 1 1 0 0 0 0 1 1 1 1 1 1

**Random forest reference**

[**https://www.stat.berkeley.edu/~breiman/RandomForests/cc\_home.htm#workings**](https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#workings)

**library**(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

ran=**randomForest**(Crime**~**M**+**Ed**+**Po1**+**U2**+**Ineq**+**Prob,data=train\_set)  
ran

##   
## Call:  
## randomForest(formula = Crime ~ M + Ed + Po1 + U2 + Ineq + Prob, data = train\_set)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## Mean of squared residuals: 77860.5  
## % Var explained: 18.26

pr=**predict**(ran, test\_set[,**c**("Crime","M","Ed","Po1","U2","Ineq","Prob")])  
**table**(pr,test\_set[,'Crime'])

##   
## pr 342 373 439 511 539 946 1043 1151 1272 1555 1674 1993  
## 693.145233333333 0 0 1 0 0 0 0 0 0 0 0 0  
## 744.866033333333 0 0 0 1 0 0 0 0 0 0 0 0  
## 796.4432 0 0 0 0 1 0 0 0 0 0 0 0  
## 814.667233333333 0 0 0 0 0 1 0 0 0 0 0 0  
## 834.195166666666 0 1 0 0 0 0 0 0 0 0 0 0  
## 835.646533333333 1 0 0 0 0 0 0 0 0 0 0 0  
## 1020.1713 0 0 0 0 0 0 0 0 0 1 0 0  
## 1057.87373333333 0 0 0 0 0 0 0 0 0 0 1 0  
## 1066.5845 0 0 0 0 0 0 0 1 0 0 0 0  
## 1090.9403 0 0 0 0 0 0 1 0 0 0 0 0  
## 1204.882 0 0 0 0 0 0 0 0 1 0 0 0  
## 1217.8456 0 0 0 0 0 0 0 0 0 0 0 1

**Question 10.2**

Describe a situation or problem from your job, everyday life, current events, etc., for which a logistic regression model would be appropriate. List some (up to 5) predictors that you might use.

**Answer**

I could use logistic regression to answer the question as to whether I am happy or not happy in life. Some indicators could include child’s health, my health, wife’s health, amount of money, amount of free time to spend with family, a feeling of fullfillment at work.

**Question 10.3**

**Part I**

Using the GermanCredit data set germancredit.txtfromhttp://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/ (description at <http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29>), use logistic regression to find a good predictive model for whether credit applicants are good credit risks or not. Show your model (factors used and their coefficients), the software output, and the quality of fit. You can use the glmfunction in R. To get a logistic regression (logit) model on data where the response is either zero or one, use family=binomial(link=”logit”)in your glmfunction call.

**rm**(list = **ls**())  
  
**set.seed**(1)  
  
credit=**read.csv**("http://freakonometrics.free.fr/german\_credit.csv", header = TRUE, sep = ",")  
  
*#ensure the data is numeric not just factors*  
as.numeric\_from\_fractor <- **function**(x) **if**(**is.factor**(x)) **as.integer**(**as.factor**(x)) **else** x  
credit[] <- **lapply**(credit, as.numeric\_from\_fractor)  
  
*#find relevant variablesby Pr(>|t|)*  
*#AIC = 2\*number of estimate params - 2 LN(maximum likelyhood function)*  
**library**(olsrr)

##   
## Attaching package: 'olsrr'

## The following object is masked from 'package:datasets':  
##   
## rivers

*#h=ols\_step\_backward\_aic(lm(Creditability~.,data=credit))*  
  
*# Split data into test and training using 0.70 (could use cross-validation later)*  
train\_index=**sample**(1000,700,replace=FALSE)  
train\_set=credit[train\_index,]  
test\_set=credit[**-**train\_index,]  
  
logit=**glm**(Creditability**~**.**-**Occupation**-**Duration.in.Current.address**-**Age..years.**-**No.of.dependents**-**Purpose,data=train\_set,family=**binomial**(link="logit"))  
  
**library**(caret)

## Loading required package: lattice

## Loading required package: ggplot2

##   
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':  
##   
## margin

pred=**predict**(logit,newdata=test\_set,type="response")

**Part2**

Because the model gives a result between 0 and 1, it requires setting a threshold probability to separate between “good” and “bad” answers. In this data set, they estimate that incorrectly identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad. Determine a good threshold probability based on your model.

An inspection of the Confusion Matricies indicates a cutoff of 0.2 is close to optimal.

**for** (i **in** **seq**(0.1,.9,0.1)) {  
 **print**(i)  
 cfm=**confusionMatrix**(data = **as.factor**(pred**>**i), reference = **as.factor**(test\_set**$**Creditability**>**i))  
 **print**(cfm)  
}

## [1] 0.1  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 2 0  
## TRUE 82 216  
##   
## Accuracy : 0.7267   
## 95% CI : (0.6725, 0.7763)  
## No Information Rate : 0.72   
## P-Value [Acc > NIR] : 0.4271   
##   
## Kappa : 0.0339   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.023810   
## Specificity : 1.000000   
## Pos Pred Value : 1.000000   
## Neg Pred Value : 0.724832   
## Prevalence : 0.280000   
## Detection Rate : 0.006667   
## Detection Prevalence : 0.006667   
## Balanced Accuracy : 0.511905   
##   
## 'Positive' Class : FALSE   
##   
## [1] 0.2  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 6 0  
## TRUE 78 216  
##   
## Accuracy : 0.74   
## 95% CI : (0.6865, 0.7887)  
## No Information Rate : 0.72   
## P-Value [Acc > NIR] : 0.2412   
##   
## Kappa : 0.0997   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.07143   
## Specificity : 1.00000   
## Pos Pred Value : 1.00000   
## Neg Pred Value : 0.73469   
## Prevalence : 0.28000   
## Detection Rate : 0.02000   
## Detection Prevalence : 0.02000   
## Balanced Accuracy : 0.53571   
##   
## 'Positive' Class : FALSE   
##   
## [1] 0.3  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 16 7  
## TRUE 68 209  
##   
## Accuracy : 0.75   
## 95% CI : (0.697, 0.798)  
## No Information Rate : 0.72   
## P-Value [Acc > NIR] : 0.1367   
##   
## Kappa : 0.2031   
##   
## Mcnemar's Test P-Value : 4.262e-12   
##   
## Sensitivity : 0.19048   
## Specificity : 0.96759   
## Pos Pred Value : 0.69565   
## Neg Pred Value : 0.75451   
## Prevalence : 0.28000   
## Detection Rate : 0.05333   
## Detection Prevalence : 0.07667   
## Balanced Accuracy : 0.57903   
##   
## 'Positive' Class : FALSE   
##   
## [1] 0.4  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 32 14  
## TRUE 52 202  
##   
## Accuracy : 0.78   
## 95% CI : (0.7288, 0.8256)  
## No Information Rate : 0.72   
## P-Value [Acc > NIR] : 0.01089   
##   
## Kappa : 0.3668   
##   
## Mcnemar's Test P-Value : 5.254e-06   
##   
## Sensitivity : 0.3810   
## Specificity : 0.9352   
## Pos Pred Value : 0.6957   
## Neg Pred Value : 0.7953   
## Prevalence : 0.2800   
## Detection Rate : 0.1067   
## Detection Prevalence : 0.1533   
## Balanced Accuracy : 0.6581   
##   
## 'Positive' Class : FALSE   
##   
## [1] 0.5  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 42 33  
## TRUE 42 183  
##   
## Accuracy : 0.75   
## 95% CI : (0.697, 0.798)  
## No Information Rate : 0.72   
## P-Value [Acc > NIR] : 0.1367   
##   
## Kappa : 0.359   
##   
## Mcnemar's Test P-Value : 0.3556   
##   
## Sensitivity : 0.5000   
## Specificity : 0.8472   
## Pos Pred Value : 0.5600   
## Neg Pred Value : 0.8133   
## Prevalence : 0.2800   
## Detection Rate : 0.1400   
## Detection Prevalence : 0.2500   
## Balanced Accuracy : 0.6736   
##   
## 'Positive' Class : FALSE   
##   
## [1] 0.6  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 53 44  
## TRUE 31 172  
##   
## Accuracy : 0.75   
## 95% CI : (0.697, 0.798)  
## No Information Rate : 0.72   
## P-Value [Acc > NIR] : 0.1367   
##   
## Kappa : 0.408   
##   
## Mcnemar's Test P-Value : 0.1659   
##   
## Sensitivity : 0.6310   
## Specificity : 0.7963   
## Pos Pred Value : 0.5464   
## Neg Pred Value : 0.8473   
## Prevalence : 0.2800   
## Detection Rate : 0.1767   
## Detection Prevalence : 0.3233   
## Balanced Accuracy : 0.7136   
##   
## 'Positive' Class : FALSE   
##   
## [1] 0.7  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 58 69  
## TRUE 26 147  
##   
## Accuracy : 0.6833   
## 95% CI : (0.6274, 0.7356)  
## No Information Rate : 0.72   
## P-Value [Acc > NIR] : 0.929   
##   
## Kappa : 0.3208   
##   
## Mcnemar's Test P-Value : 1.639e-05   
##   
## Sensitivity : 0.6905   
## Specificity : 0.6806   
## Pos Pred Value : 0.4567   
## Neg Pred Value : 0.8497   
## Prevalence : 0.2800   
## Detection Rate : 0.1933   
## Detection Prevalence : 0.4233   
## Balanced Accuracy : 0.6855   
##   
## 'Positive' Class : FALSE   
##   
## [1] 0.8  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 68 101  
## TRUE 16 115  
##   
## Accuracy : 0.61   
## 95% CI : (0.5523, 0.6655)  
## No Information Rate : 0.72   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.2612   
##   
## Mcnemar's Test P-Value : 8.113e-15   
##   
## Sensitivity : 0.8095   
## Specificity : 0.5324   
## Pos Pred Value : 0.4024   
## Neg Pred Value : 0.8779   
## Prevalence : 0.2800   
## Detection Rate : 0.2267   
## Detection Prevalence : 0.5633   
## Balanced Accuracy : 0.6710   
##   
## 'Positive' Class : FALSE   
##   
## [1] 0.9  
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 76 157  
## TRUE 8 59  
##   
## Accuracy : 0.45   
## 95% CI : (0.3928, 0.5082)  
## No Information Rate : 0.72   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1154   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.9048   
## Specificity : 0.2731   
## Pos Pred Value : 0.3262   
## Neg Pred Value : 0.8806   
## Prevalence : 0.2800   
## Detection Rate : 0.2533   
## Detection Prevalence : 0.7767   
## Balanced Accuracy : 0.5890   
##   
## 'Positive' Class : FALSE   
##

**confusionMatrix**(data = **as.factor**(pred**>**.2), reference = **as.factor**(test\_set**$**Creditability**>**0.2))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 6 0  
## TRUE 78 216  
##   
## Accuracy : 0.74   
## 95% CI : (0.6865, 0.7887)  
## No Information Rate : 0.72   
## P-Value [Acc > NIR] : 0.2412   
##   
## Kappa : 0.0997   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.07143   
## Specificity : 1.00000   
## Pos Pred Value : 1.00000   
## Neg Pred Value : 0.73469   
## Prevalence : 0.28000   
## Detection Rate : 0.02000   
## Detection Prevalence : 0.02000   
## Balanced Accuracy : 0.53571   
##   
## 'Positive' Class : FALSE   
##