732A99 chetabook

Anubhav Dikshit(anudi287)

26 November 2018

Contents

Simple Tasks

library and other

```
library(ggplot2) # plots
library(tree) # decision tree
library(caret) # summary and confusion table
```

```
## Loading required package: lattice
```

```
library(kknn) # kknn
```

```
##
## Attaching package: 'kknn'
```

```
## The following object is masked from 'package:caret':
##
##     contr.dummy
```

```
library(xlsx) # reading excel
library(MASS) # Step AIC
library(jtools) # summ function
library(dplyr) # pipelining
```

```
##
## Attaching package: 'dplyr'
## The following object is masked from 'package:MASS':
##
##     select
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(glmnet) # lasso and ridge
```

```
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-16
```

```
library(mgcv) # spline
```

```
## Loading required package: nlme
##
## Attaching package: 'nlme'
## The following object is masked from 'package:dplyr':
##
##     collapse
## This is mgcv 1.8-24. For overview type 'help("mgcv-package")'.
```

```
library(kernlab) # SVM
```

```
##
## Attaching package: 'kernlab'
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
# colours (colour blind friendly)
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2",
               "#D55E00", "#CC79A7")
```

Reading Excel

```
data <- xlsx::read.xlsx("spambase.xlsx", sheetName= "spambase_data")
data$Spam <- as.factor(data$Spam)
```

Spliting the Datasets

Divide into train/test

```
# 50-50 split
n=nrow(data)
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]
```

train/test/validation

```
# 50-25-25 split
n=nrow(data)
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]

id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=data[id2,]

id3=setdiff(id1,id2)
test=data[id3,]
```

Regression

Logistic Regression along with confusion matrix

```
best_model <- glm(formula = Spam ~., family = binomial, data = train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(best_model)
```

```
##
## Call:
## glm(formula = Spam ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5205  -0.4402  -0.0005   0.6584   3.6196
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.508e+00  2.011e-01   7.499 6.44e-14 ***
## Word1       -7.192e-01  5.015e-01  -1.434 0.151520
## Word2        3.994e-02  3.014e-01   0.133 0.894580
## Word3       -3.529e-01  1.839e-01  -1.919 0.055019 .
```

```
## Word4        1.370e-01  1.117e-01   1.226 0.220230
## Word5        1.221e-01  1.413e-01   0.864 0.387400
## Word6        2.887e-01  4.153e-01   0.695 0.486888
## Word7       -2.948e-01  3.348e-01  -0.880 0.378676
## Word8       -1.034e-01  3.510e-01  -0.295 0.768337
## Word9       -1.085e-01  4.053e-01  -0.268 0.788983
## Word10      -3.091e-02  1.668e-01  -0.185 0.852991
## Word11      -6.088e-01  6.790e-01  -0.897 0.369902
## Word12       1.614e-01  1.073e-01   1.505 0.132378
## Word13       7.811e-01  3.572e-01   2.187 0.028771 *
## Word14      -4.819e-01  3.003e-01  -1.605 0.108598
## Word15      -1.305e-01  3.884e-01  -0.336 0.736861
## Word16       3.171e-01  2.332e-01   1.360 0.173891
## Word17      -9.201e-02  2.823e-01  -0.326 0.744479
## Word18       2.330e-02  2.322e-01   0.100 0.920074
## Word19       3.694e-04  5.746e-02   0.006 0.994871
## Word20       1.688e-02  3.181e-01   0.053 0.957687
## Word21      -2.821e-02  1.072e-01  -0.263 0.792524
## Word22      -4.767e-01  3.200e-01  -1.490 0.136337
## Word23       2.541e-01  3.413e-01   0.745 0.456525
## Word24      -2.483e-01  6.255e-01  -0.397 0.691372
## Word25      -7.828e-02  5.852e-02  -1.338 0.181027
## Word26       3.733e-03  1.358e-01   0.027 0.978071
## Word27      -2.238e-01  1.077e-01  -2.078 0.037749 *
## Word28       1.320e-01  1.880e-01   0.702 0.482776
## Word29      -8.131e-02  9.119e-02  -0.892 0.372564
## Word30      -1.815e+00  6.195e-01  -2.930 0.003391 **
## Word31      -4.694e+00  1.853e+00  -2.533 0.011296 *
## Word32      -1.194e+02  1.513e+04  -0.008 0.993703
## Word33      -2.899e+00  6.794e-01  -4.268 1.98e-05 ***
## Word34      -3.710e+00  4.352e+00  -0.852 0.394004
## Word35      -7.033e+00  1.996e+00  -3.522 0.000428 ***
## Word36      -1.678e+00  3.810e-01  -4.404 1.06e-05 ***
## Word37      -8.583e-01  2.175e-01  -3.947 7.92e-05 ***
## Word38      -6.043e-01  1.279e+00  -0.472 0.636575
## Word39      -1.877e+00  4.282e-01  -4.384 1.16e-05 ***
## Word40       7.393e-02  3.400e-01   0.217 0.827885
## Word41      -3.326e+02  1.656e+04  -0.020 0.983978
## Word42      -5.352e+00  1.302e+00  -4.109 3.98e-05 ***
## Word43      -2.592e+00  7.353e-01  -3.525 0.000423 ***
## Word44      -2.931e+00  6.601e-01  -4.441 8.96e-06 ***
## Word45      -1.141e+00  1.681e-01  -6.785 1.16e-11 ***
## Word46      -3.288e+00  5.178e-01  -6.350 2.15e-10 ***
## Word47      -3.741e+00  2.030e+00  -1.843 0.065399 .
## Word48      -4.390e+00  1.473e+00  -2.980 0.002878 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1696.82  on 1369  degrees of freedom
## Residual deviance:  928.54  on 1321  degrees of freedom
## AIC: 1026.5
##
```

```
## Number of Fisher Scoring iterations: 23
```

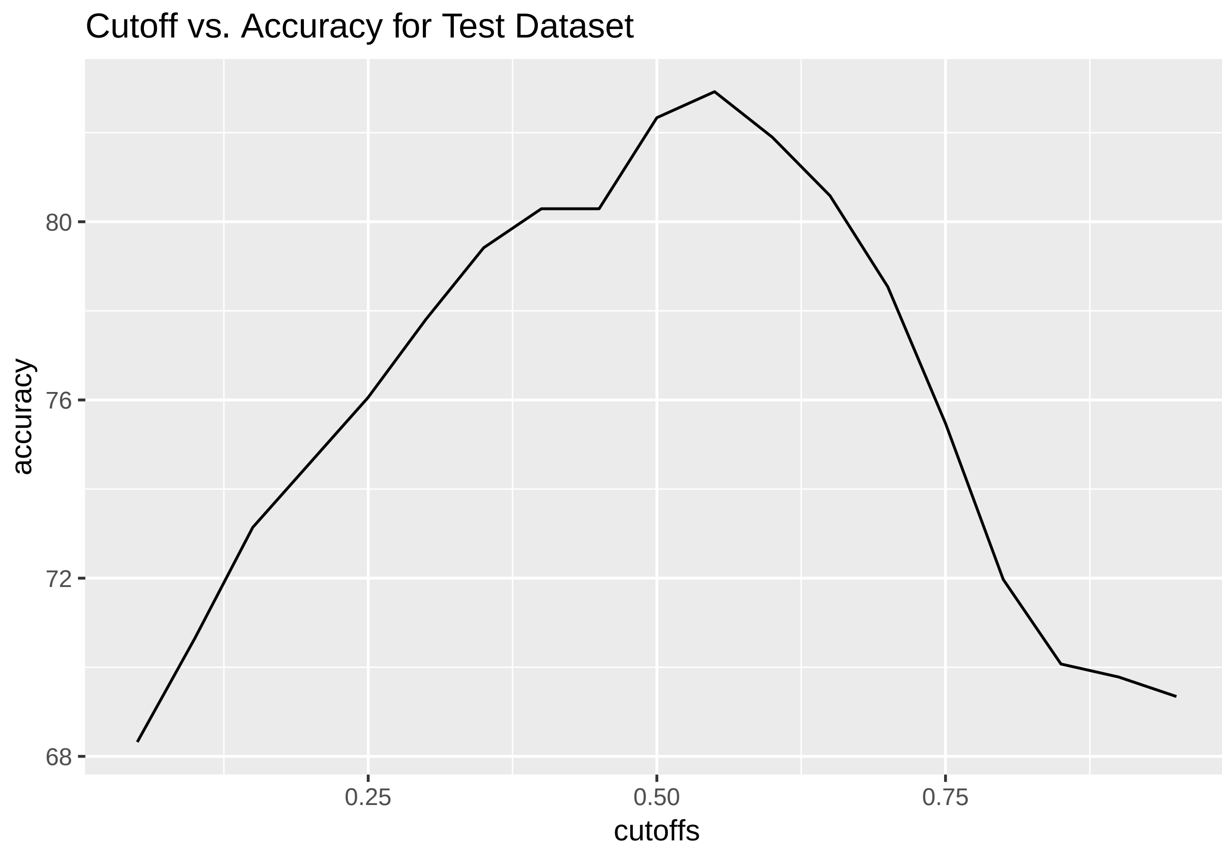
```r
train$prediction_prob <- predict(best_model, newdata = train, type = "response")
train$prediction_class_50 <- ifelse(train$prediction_prob > 0.50, 1, 0)

test$prediction_prob <- predict(best_model, newdata = test, type = "response")
test$prediction_class_50 <- ifelse(test$prediction_prob > 0.50, 1, 0)

conf_train <- table(train$Spam, train$prediction_class_50)
names(dimnames(conf_train)) <- c("Actual Train", "Predicted Train")
caret::confusionMatrix(conf_train)
```

```
## Confusion Matrix and Statistics
##
##               Predicted Train
## Actual Train    0   1
##            0 803 142
##            1  81 344
##
##                Accuracy : 0.8372
##                  95% CI : (0.8166, 0.8564)
##     No Information Rate : 0.6453
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6341
##  Mcnemar's Test P-Value : 5.872e-05
##
##             Sensitivity : 0.9084
##             Specificity : 0.7078
##          Pos Pred Value : 0.8497
##          Neg Pred Value : 0.8094
##              Prevalence : 0.6453
##          Detection Rate : 0.5861
##    Detection Prevalence : 0.6898
##       Balanced Accuracy : 0.8081
##
##        'Positive' Class : 0
##
```

Choosing the best cutoff for test

```r
cutoffs <- seq(from = 0.05, to = 0.95, by = 0.05)
accuracy <- NULL

for (i in seq_along(cutoffs)){
    prediction <- ifelse(test$prediction_prob >= cutoffs[i], 1, 0) #Predicting for cut-off

    accuracy <- c(accuracy,length(which(test$Spam == prediction))/length(prediction)*100)}

cutoff_data <- as.data.frame(cbind(cutoffs, accuracy))

ggplot(data = cutoff_data, aes(x = cutoffs, y = accuracy)) +
  geom_line() +
```

```
  ggtitle("Cutoff vs. Accuracy for Test Dataset")
```



KNN

```
knn_model30 <- train.kknn(Spam ~., data = train, kmax = 30)

test$knn_prediction_class <- predict(knn_model30, test)

conf_test2 <- table(test$Spam, test$knn_prediction_class)
names(dimnames(conf_test2)) <- c("Actual Test", "Predicted Test")
confusionMatrix(conf_test2)
```

```
## Confusion Matrix and Statistics
##
##              Predicted Test
## Actual Test    0   1
##            0 402  74
##            1  66 143
##
##                 Accuracy : 0.7956
##                   95% CI : (0.7634, 0.8252)
##      No Information Rate : 0.6832
##      P-Value [Acc > NIR] : 3.278e-11
##
```

```
##                   Kappa : 0.5231
##  Mcnemar's Test P-Value : 0.5541
##
##             Sensitivity : 0.8590
##             Specificity : 0.6590
##          Pos Pred Value : 0.8445
##          Neg Pred Value : 0.6842
##              Prevalence : 0.6832
##          Detection Rate : 0.5869
##    Detection Prevalence : 0.6949
##       Balanced Accuracy : 0.7590
##
##        'Positive' Class : 0
##
```

Step AIC

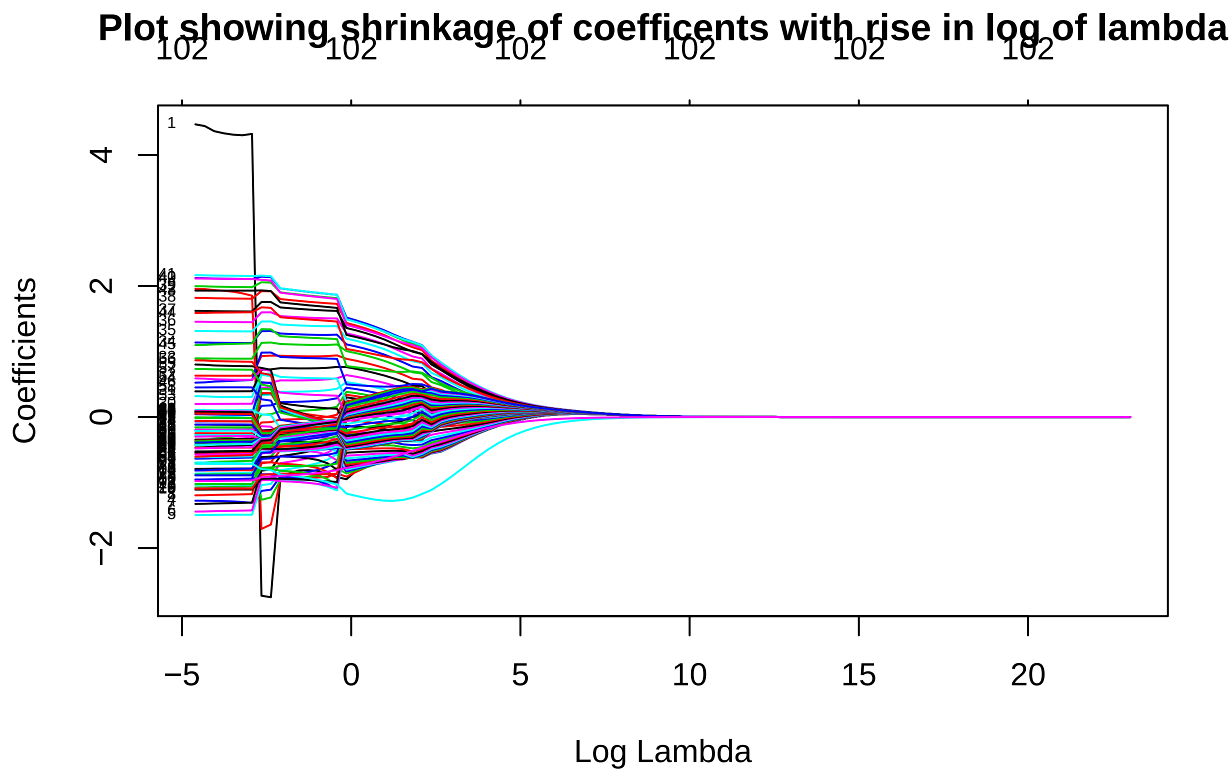
```r
tecator_data <- read.xlsx("tecator.xlsx", sheetName = "data")
tecator_data <- tecator_data[,2:NCOL(tecator_data)] # removing sample column


min.model1 = lm(Fat ~ 1, data=tecator_data[,-1])
biggest1 <- formula(lm(Fat ~.,  data=tecator_data[,-1]))

step.model1 <- stepAIC(min.model1, direction ='forward', scope=biggest1, trace = FALSE)
summ(step.model1)
```

```
## MODEL INFO:
## Observations: 215
## Dependent Variable: Fat
## Type: OLS linear regression
##
## MODEL FIT:
## F(29,185) = 4775.35, p = 0.00
## R² = 1.00
## Adj. R² = 1.00
##
## Standard errors: OLS
##                   Est.    S.E. t val.    p
## (Intercept)      93.46    1.59  58.86 0.00 ***
## Moisture         -1.03    0.02 -54.25 0.00 ***
## Protein          -0.64    0.06 -10.91 0.00 ***
## Channel100       66.56   48.18   1.38 0.17
## Channel41     -3268.11  826.92  -3.95 0.00 ***
## Channel7        -64.03   20.80  -3.08 0.00  **
## Channel48     -2022.46  254.46  -7.95 0.00 ***
## Channel42      4934.22 1124.96   4.39 0.00 ***
## Channel50      1239.52  236.09   5.25 0.00 ***
## Channel45      4796.22  783.38   6.12 0.00 ***
## Channel66      2435.79 1169.85   2.08 0.04   *
## Channel56      2373.00  540.06   4.39 0.00 ***
## Channel90      -258.27  247.22  -1.04 0.30
## Channel60      -264.27  708.11  -0.37 0.71
```

```
## Channel70      14.25  327.12   0.04 0.97
## Channel67   -2015.92  543.74  -3.71 0.00 ***
## Channel59     635.71  996.31   0.64 0.52
## Channel65    -941.61 1009.23  -0.93 0.35
## Channel58    1054.24  927.95   1.14 0.26
## Channel44   -5733.84 1079.19  -5.31 0.00 ***
## Channel18     299.80   88.43   3.39 0.00 ***
## Channel78    2371.11  361.25   6.56 0.00 ***
## Channel84    -428.99  338.35  -1.27 0.21
## Channel62    3062.97  769.59   3.98 0.00 ***
## Channel53    -804.39  203.44  -3.95 0.00 ***
## Channel75   -1461.42  402.26  -3.63 0.00 ***
## Channel57   -3266.79  876.71  -3.73 0.00 ***
## Channel63   -2844.66  906.40  -3.14 0.00  **
## Channel24    -308.71   97.87  -3.15 0.00  **
## Channel37     401.64  151.76   2.65 0.01  **
```

Ridge Regression

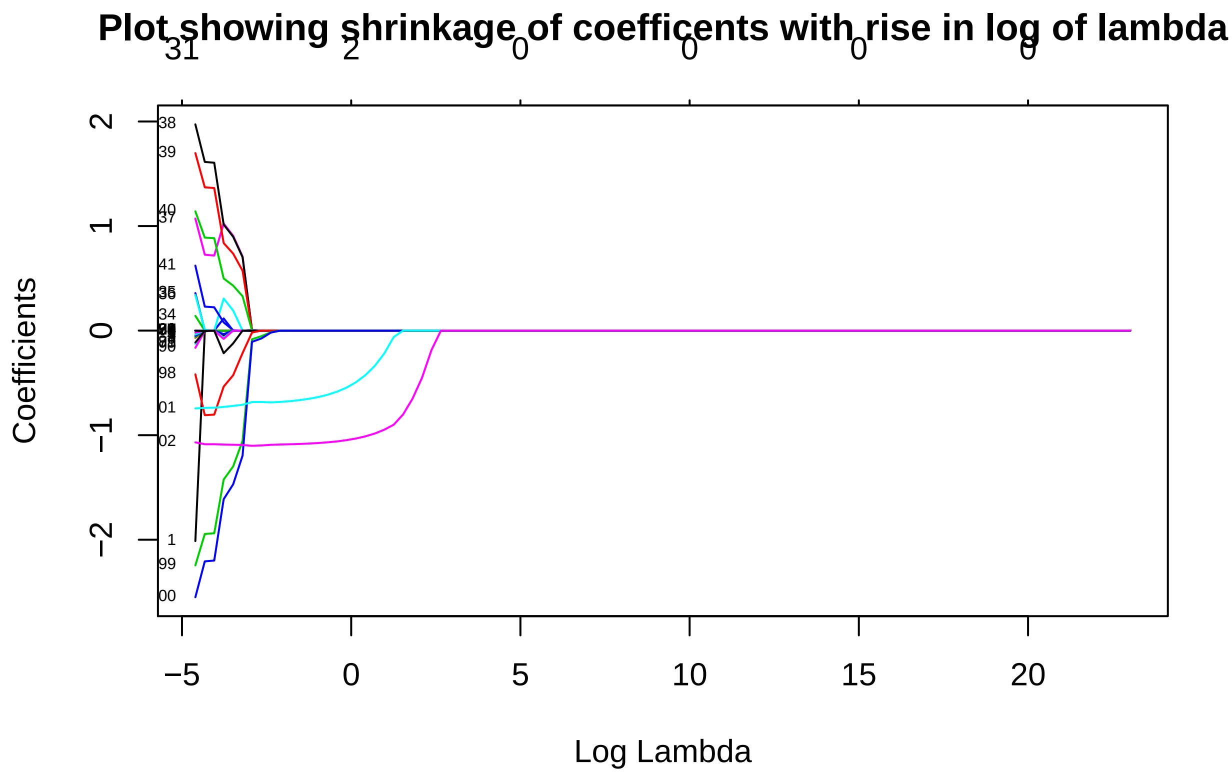
```
y <- tecator_data %>% select(Fat) %>% data.matrix()
x <- tecator_data %>% select(-c(Fat)) %>% data.matrix()

lambda <- 10^seq(10, -2, length = 100)

ridge_fit <- glmnet(x, y, alpha = 0, family = "gaussian", lambda = lambda)
plot(ridge_fit, xvar = "lambda", label = TRUE,
     main = "Plot showing shrinkage of coefficents with rise in log of lambda")
```




```
## Change of coefficent with respect to lambda
result <- NULL
for(i in lambda){
temp <- t(coef(ridge_fit, i)) %>% as.matrix()
temp <- cbind(temp, lambda = i)
result <- rbind(temp, result)
}
result <- result %>% as.data.frame() %>% arrange(lambda)
```

Lasso Regression

```
lambda <- 10^seq(10, -2, length = 100)

lasso_fit <- glmnet(x, y, alpha = 1, family = "gaussian", lambda = lambda)
plot(lasso_fit, xvar = "lambda", label = TRUE,
     main = "Plot showing shrinkage of coefficents with rise in log of lambda")
```



Regression using CV

```r
#find the best lambda from our list via cross-validation

lambda_lasso <- 10^seq(10, -2, length = 100)
lambda_lasso[101] <- 0
lasso_cv <- cv.glmnet(x,y, alpha=1, lambda = lambda_lasso, type.measure="mse")
coef(lasso_cv, lambda = lasso_cv$lambda.min)
```

```
## 103 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) 100.1826458
## Channel1      .
## Channel2      .
## Channel3      .
## Channel4      .
## Channel5      .
## Channel6      .
## Channel7      .
## Channel8      .
## Channel9      .
## Channel10     .
## Channel11     .
## Channel12     .
## Channel13     .
```

```
## Channel14      .
## Channel15      .
## Channel16      .
## Channel17      .
## Channel18      .
## Channel19      .
## Channel20      .
## Channel21      .
## Channel22      .
## Channel23      .
## Channel24      .
## Channel25      .
## Channel26      .
## Channel27      .
## Channel28      .
## Channel29      .
## Channel30      .
## Channel31      .
## Channel32      .
## Channel33      .
## Channel34      .
## Channel35      .
## Channel36      .
## Channel37      0.7085073
## Channel38      0.7023525
## Channel39      0.5726923
## Channel40      0.3285077
## Channel41      .
## Channel42      .
## Channel43      .
## Channel44      .
## Channel45      .
## Channel46      .
## Channel47      .
## Channel48      .
## Channel49      .
## Channel50      .
## Channel51      .
## Channel52      .
## Channel53      .
## Channel54      .
## Channel55      .
## Channel56      .
## Channel57      .
## Channel58      .
## Channel59      .
## Channel60      .
## Channel61      .
## Channel62      .
## Channel63      .
## Channel64      .
## Channel65      .
## Channel66      .
## Channel67      .
```

```
## Channel68      .
## Channel69      .
## Channel70      .
## Channel71      .
## Channel72      .
## Channel73      .
## Channel74      .
## Channel75      .
## Channel76      .
## Channel77      .
## Channel78      .
## Channel79      .
## Channel80      .
## Channel81      .
## Channel82      .
## Channel83      .
## Channel84      .
## Channel85      .
## Channel86      .
## Channel87      .
## Channel88      .
## Channel89      .
## Channel90      .
## Channel91      .
## Channel92      .
## Channel93      .
## Channel94      .
## Channel95      .
## Channel96      .
## Channel97      .
## Channel98    -0.2147181
## Channel99    -1.0547607
## Channel100   -1.1957408
## Protein      -0.7093446
## Moisture     -1.0939227
```

```
lasso_cv$lambda.min
```

```
## [1] 0
```

```
## Change of coefficent with respect to lambda
result_lasso <- NULL
for(i in 1:length(lambda_lasso)){
temp <- lasso_cv$cvm[i] %>% as.matrix()
temp <- cbind(CVM_error = temp, lambda = lasso_cv$lambda[i])
result_lasso <- rbind(temp, result_lasso)
}
```

Classification

Classification using Decision trees (tree lib)

```r
set.seed(12345)

data <- read.csv("crx.csv", header = TRUE)
data$Class <- as.factor(data$Class)

# 50-50 split
n=nrow(data)
id=sample(1:n, floor(n*0.8))
train=data[id,]
test=data[-id,]

tree_deviance <- tree::tree(Class~., data=train, split = c("deviance"))
tree_gini <- tree::tree(Class~., data=train, split = c("gini"))

# Visualize the decision tree with rpart.plot
summary(tree_deviance)
```

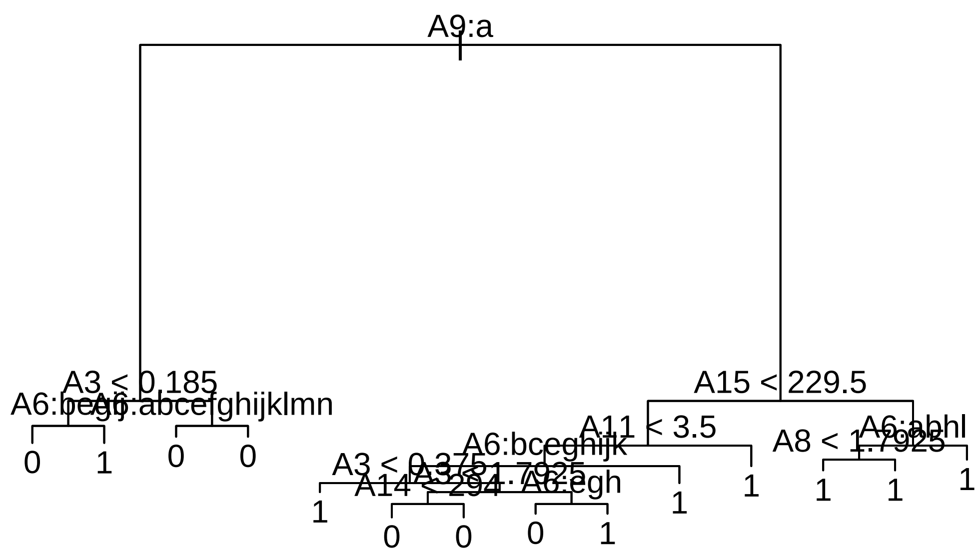
```
##
## Classification tree:
## tree::tree(formula = Class ~ ., data = train, split = c("deviance"))
## Variables actually used in tree construction:
## [1] "A9"  "A3"  "A6"  "A15" "A11" "A14" "A8"
## Number of terminal nodes:  14
## Residual mean deviance:  0.4752 = 255.6 / 538
## Misclassification error rate: 0.09601 = 53 / 552
```

```r
# predicting on the test dataset to get the misclassification rate.
predict_tree_deviance <- predict(tree_deviance, newdata = test, type = "class")
predict_tree_gini <- predict(tree_deviance, newdata = test, type = "class")

conf_tree_deviance <- table(test$Class, predict_tree_deviance)
names(dimnames(conf_tree_deviance)) <- c("Actual Test", "Predicted Test")
caret::confusionMatrix(conf_tree_deviance)
```

```
## Confusion Matrix and Statistics
##
##            Predicted Test
## Actual Test  0  1
##           0 62 15
##           1  4 57
##
##                Accuracy : 0.8623
##                  95% CI : (0.7934, 0.915)
##     No Information Rate : 0.5217
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.726
##  Mcnemar's Test P-Value : 0.02178
##
##             Sensitivity : 0.9394
```

```
##             Specificity : 0.7917
##          Pos Pred Value : 0.8052
##          Neg Pred Value : 0.9344
##              Prevalence : 0.4783
##          Detection Rate : 0.4493
##    Detection Prevalence : 0.5580
##       Balanced Accuracy : 0.8655
##
##        'Positive' Class : 0
##
```

```r
# plot of the tree
plot(tree_deviance)
text(tree_deviance)
```

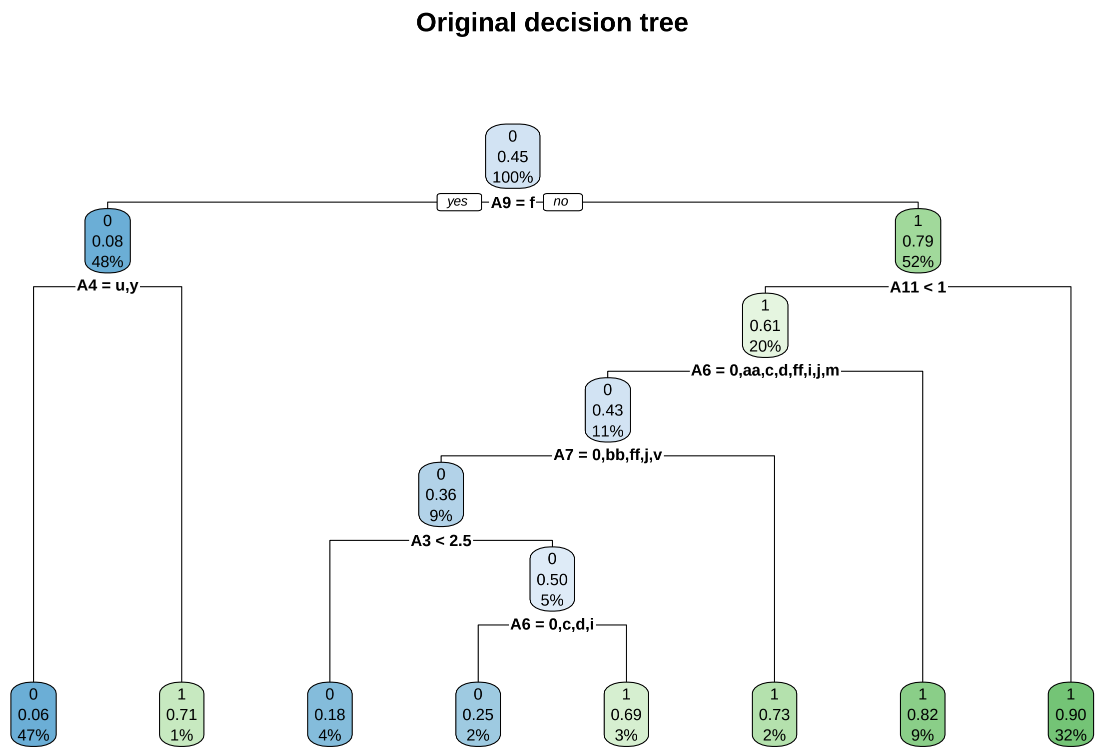


Classification using rpart

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

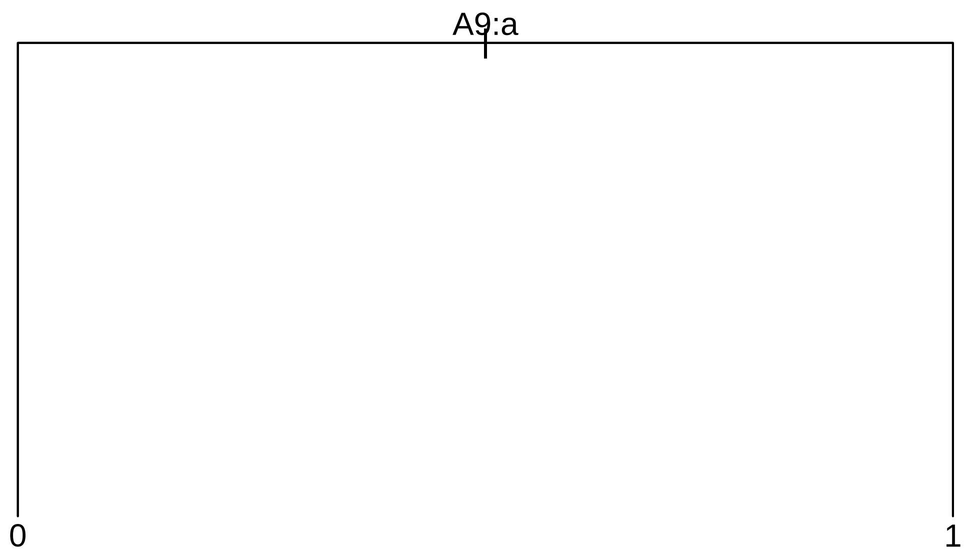
```
## Loading required package: rpart
```

```r
library(rpart)

set.seed(12345)


decision_tree_rpart <- rpart::rpart(data = train, formula = Class~., method = "class")
rpart.plot::rpart.plot(decision_tree_rpart, main= "Original decision tree")
```

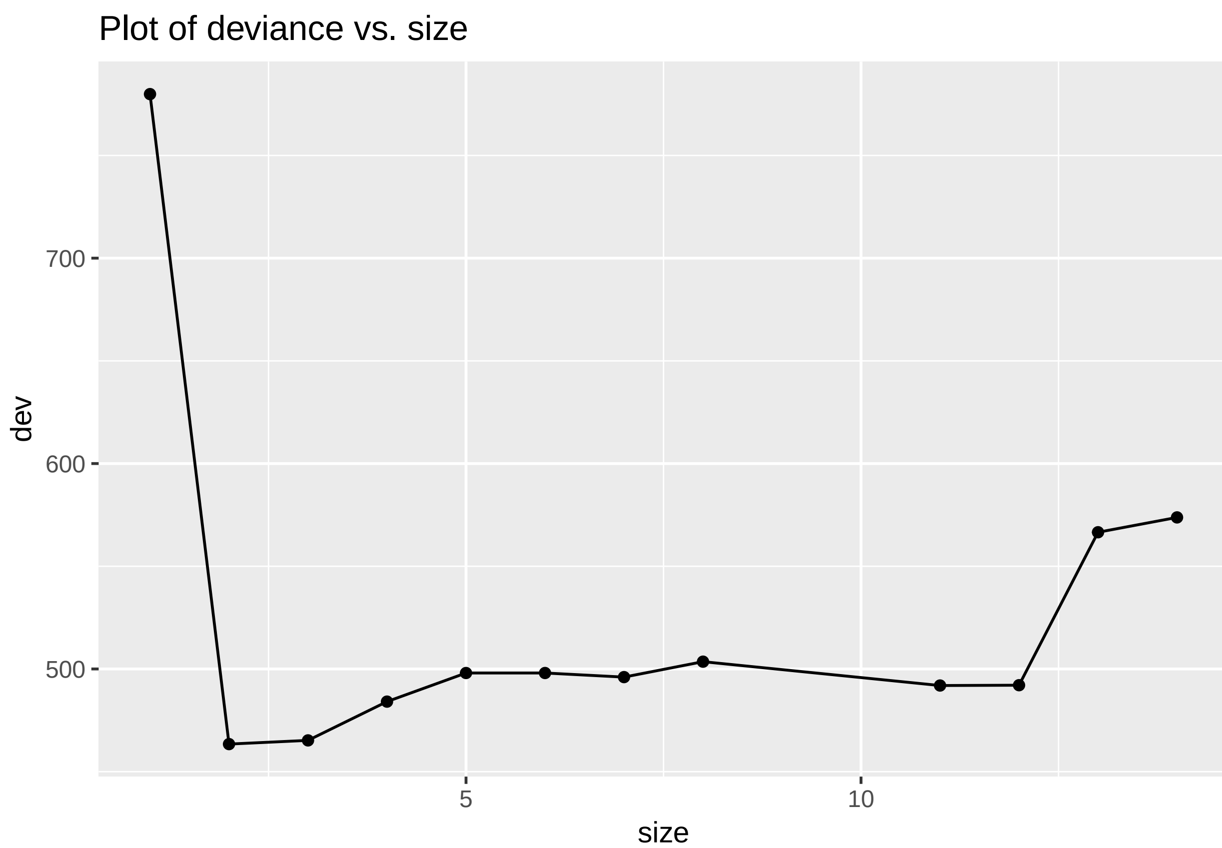


prune the tree using cross validation

```
library(ggplot2)

set.seed(12345)
cv_tree <- cv.tree(tree_deviance, FUN = prune.tree, K = 10)
df_result <- as.data.frame(cbind(size = cv_tree$size, dev = cv_tree$dev))
# puring the tree for leaf size of 3
best_tree <- prune.tree(tree_deviance, best = 2)
plot(best_tree, main="Pruned Tree for the given dataset")
text(best_tree)
```

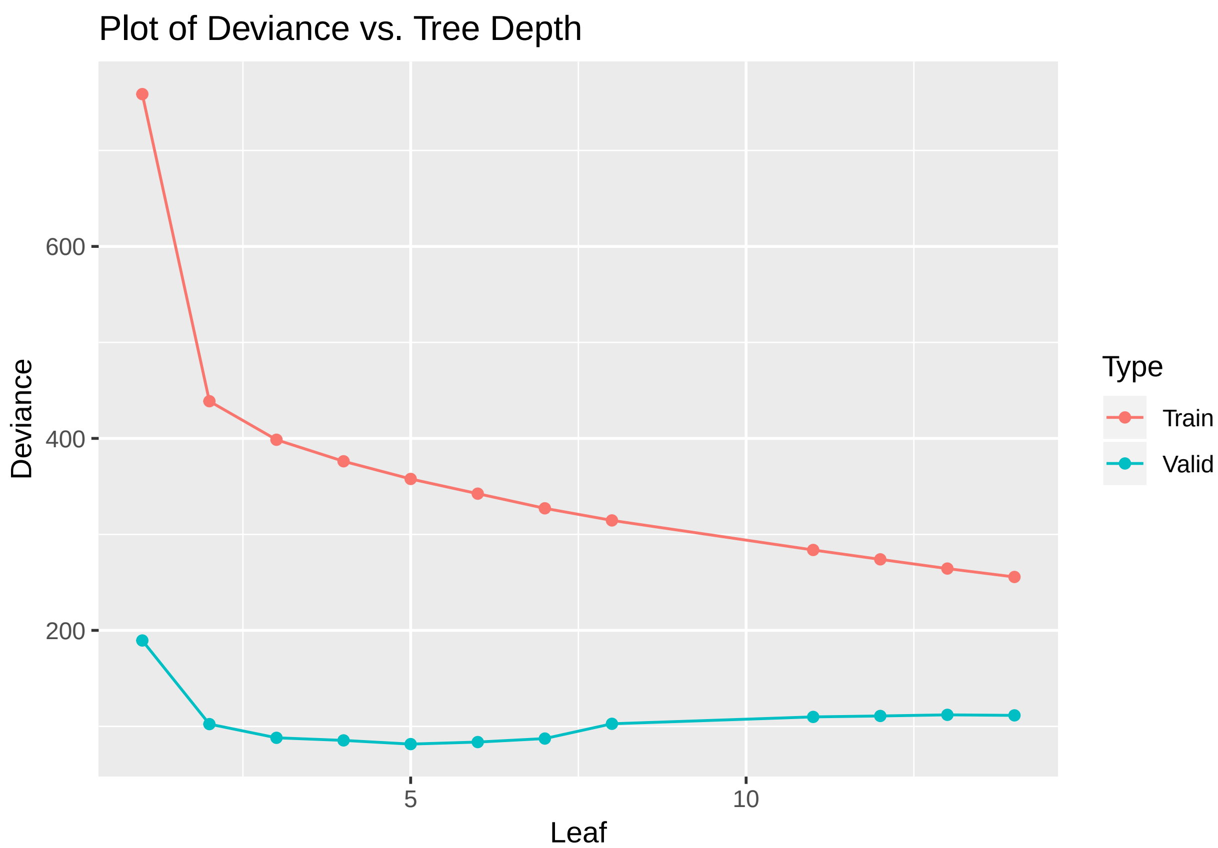



```r
ggplot(df_result, aes(x = size, y = dev)) + geom_point() + geom_line() + ggtitle("Plot of deviance vs. s
```



prune the tree using error

```
set.seed(12345)
tree_deviance <- tree::tree(Class~., data=train, split = c("deviance"))

tree_prune_train <- prune.tree(tree_deviance, method = c("deviance"))
tree_prune_valid <- prune.tree(tree_deviance, newdata = test ,method = c("deviance"))

result_train <- cbind(tree_prune_train$size,
tree_prune_train$dev, "Train")

result_valid <- cbind(tree_prune_valid$size,
tree_prune_valid$dev, "Valid")

result <- as.data.frame(rbind(result_valid, result_train))
colnames(result) <- c("Leaf", "Deviance", "Type")
result$Leaf <- as.numeric(as.character(result$Leaf))
result$Deviance <- as.numeric(as.character(result$Deviance))

# plot of deviance vs. number of leafs
ggplot(data = result, aes(x = Leaf, y = Deviance, colour = Type)) +
geom_point() + geom_line() +
ggtitle("Plot of Deviance vs. Tree Depth")
```



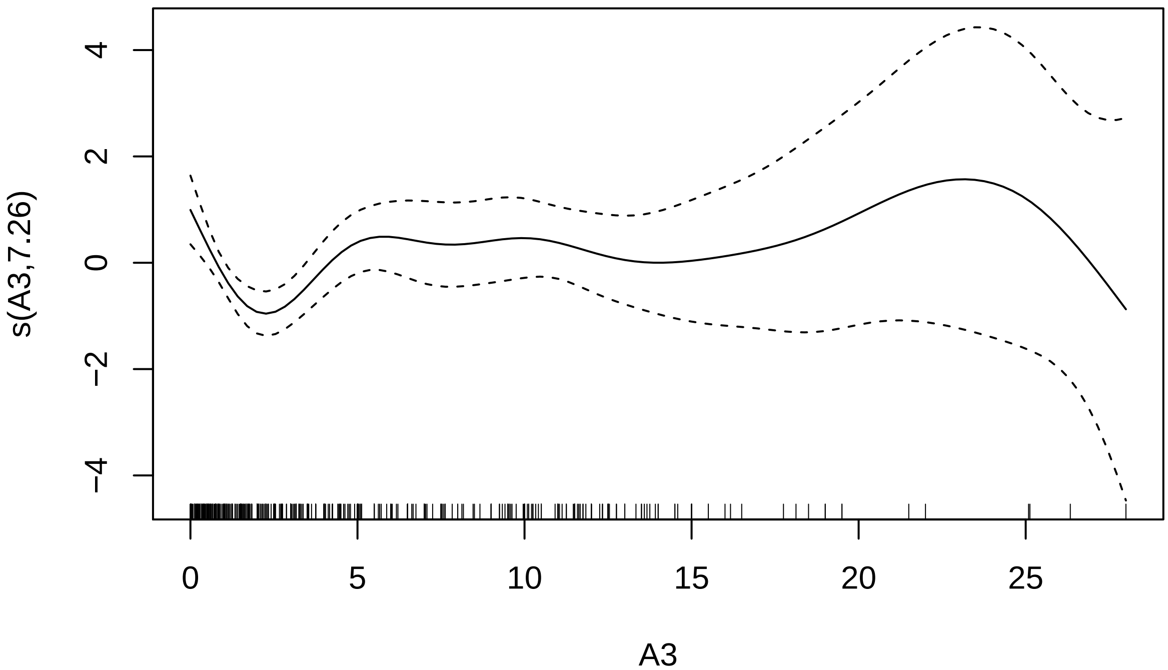
GAM Model or Spline for classification

```r
set.seed(12345)

# using family = binomial for classfication
gam_model <- mgcv::gam(data=train, formula = Class~s(A3)+A9, family=binomial)
summary(gam_model)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## Class ~ s(A3) + A9
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.6202     0.2479  -10.57   <2e-16 ***
## A9t           3.9741     0.3004   13.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##         edf Ref.df Chi.sq p-value
## s(A3) 7.264  8.259  22.28  0.0057 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.525   Deviance explained = 45.7%
## UBRE = -0.22005  Scale est. = 1         n = 552
```

```r
plot(gam_model)
```



SVM, width is the sigma here. kernel rbfdot is gaussian. vanilladot is linear

```r
data(spam)

## create test and training set
index <- sample(1:dim(spam)[1])
spamtrain <- spam[index[1:floor(dim(spam)[1]/2)], ]
spamtest <- spam[index[((ceiling(dim(spam)[1]/2)) + 1):dim(spam)[1]], ]

spamtrain$type <- as.factor(spamtrain$type)

model_0.05 <- kernlab::ksvm(type~., data=spamtrain, kernel="rbfdot", kpar=list(sigma=0.05), C=0.5)
model_0.05
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
```

```
##  parameter : cost C = 0.5
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.05
##
## Number of Support Vectors : 1077
##
## Objective Function Value : -313.3263
## Training error : 0.056087
```

```r
conf_model_0.05 <- table(spamtrain[,58], predict(model_0.05, spamtrain[,-58]))
names(dimnames(conf_model_0.05)) <- c("Actual Test", "P2redicted Test")
caret::confusionMatrix(conf_model_0.05)
```

```
## Confusion Matrix and Statistics
##
##            P2redicted Test
## Actual Test nonspam spam
##     nonspam    1379   32
##     spam         97  792
##
##                Accuracy : 0.9439
##                  95% CI : (0.9337, 0.953)
##     No Information Rate : 0.6417
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8801
##  Mcnemar's Test P-Value : 1.752e-08
##
##             Sensitivity : 0.9343
##             Specificity : 0.9612
##          Pos Pred Value : 0.9773
##          Neg Pred Value : 0.8909
##              Prevalence : 0.6417
##          Detection Rate : 0.5996
##    Detection Prevalence : 0.6135
##       Balanced Accuracy : 0.9477
##
##        'Positive' Class : nonspam
##
```