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
library(kknn) # kknn
library(xlsx) # reading excel
library(MASS) # Step AIC
library(jtools) # summ function
library(dplyr) # pipelining
library(glmnet) # lasso and ridge
library(mgcv) # spline
library(kernlab) # SVM
library(mboost) # ensemble ADA boost
library(randomForest) # randomforest
```

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

Reading Excel

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

Spliting the Datasets

Divide into train/test

```r
# 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

```r
# 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,]
```

Cross-Validation custom

```r
#Randomly shuffle the data
data <- data[sample(nrow(data)),]

#Create N equally size folds
folds <- cut(seq(1,nrow(data)), breaks=10,labels=FALSE)

#Perform N fold cross validation
for(i in 1:10){
    #Segement your data by fold using the which() function
    testIndexes <- which(folds==i, arr.ind = TRUE)
```

```
    testData <- data[testIndexes,]
    trainData <- data[-testIndexes,]
    #Use the test and train data partitions however you desire, run model code here
}
```

Regression

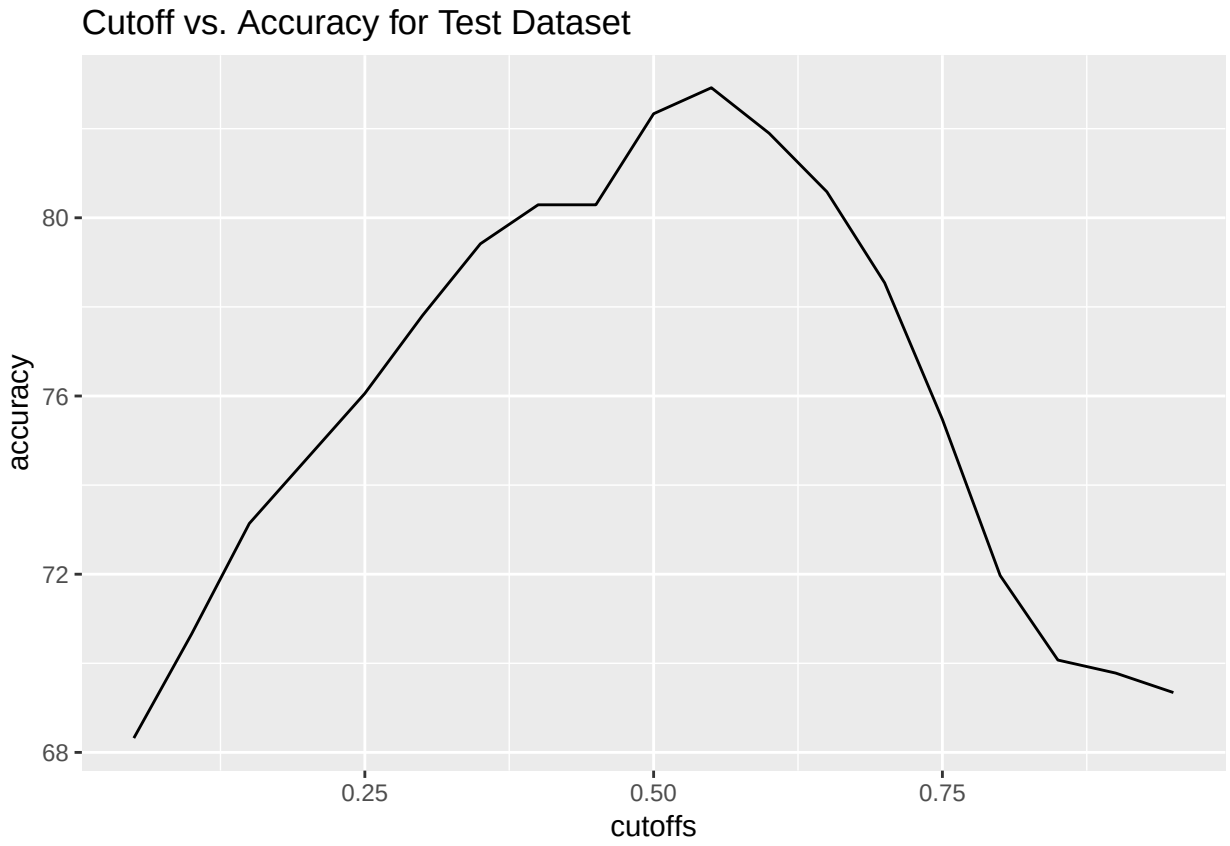
Logistic Regression along with confusion matrix

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

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

```
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")
```

```r
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

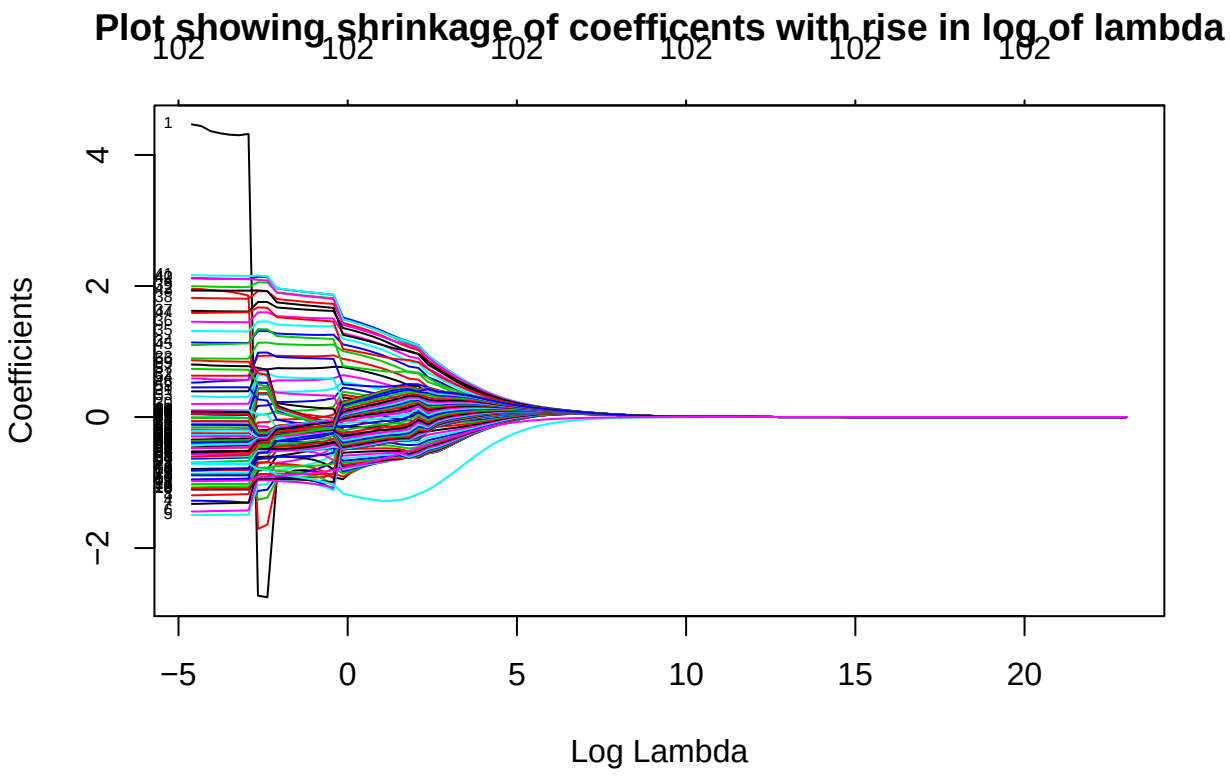
```r
tecator_data <- read.xlsx("tecator.xlsx", sheetName = "data")
tecator_data <- tecator_data[,2:NCOL(tecator_data)] # removing sample column


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

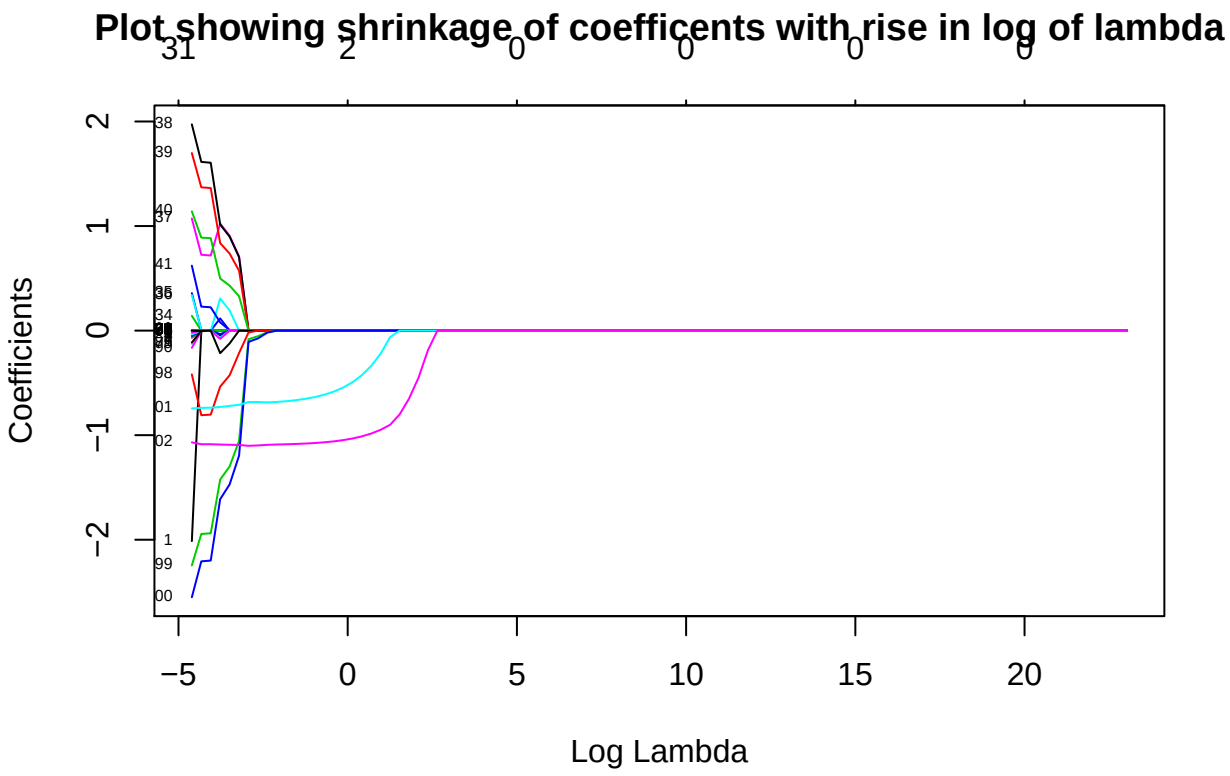
```
y <- tecator_data %>% select(Fat) %>% data.matrix()
x <- tecator_data %>% select(-c(Fat)) %>% data.matrix()

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

```
#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)

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)

```
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)
```

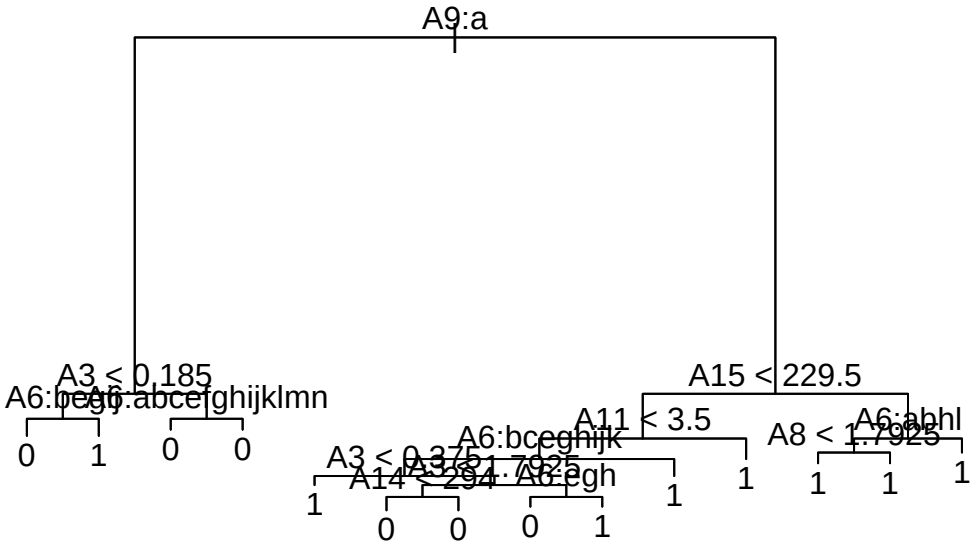
```
##
## Classification tree:
## tree::tree(formula = Class ~ ., data = train, split = c("deviance"))
## Variables actually used in tree construction:
## [1] "A9"  "A3"  "A6"  "A15" "A11" "A14" "A8"
## Number of terminal nodes:  14
## Residual mean deviance:  0.4752 = 255.6 / 538
## Misclassification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

```
# 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
##
```

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



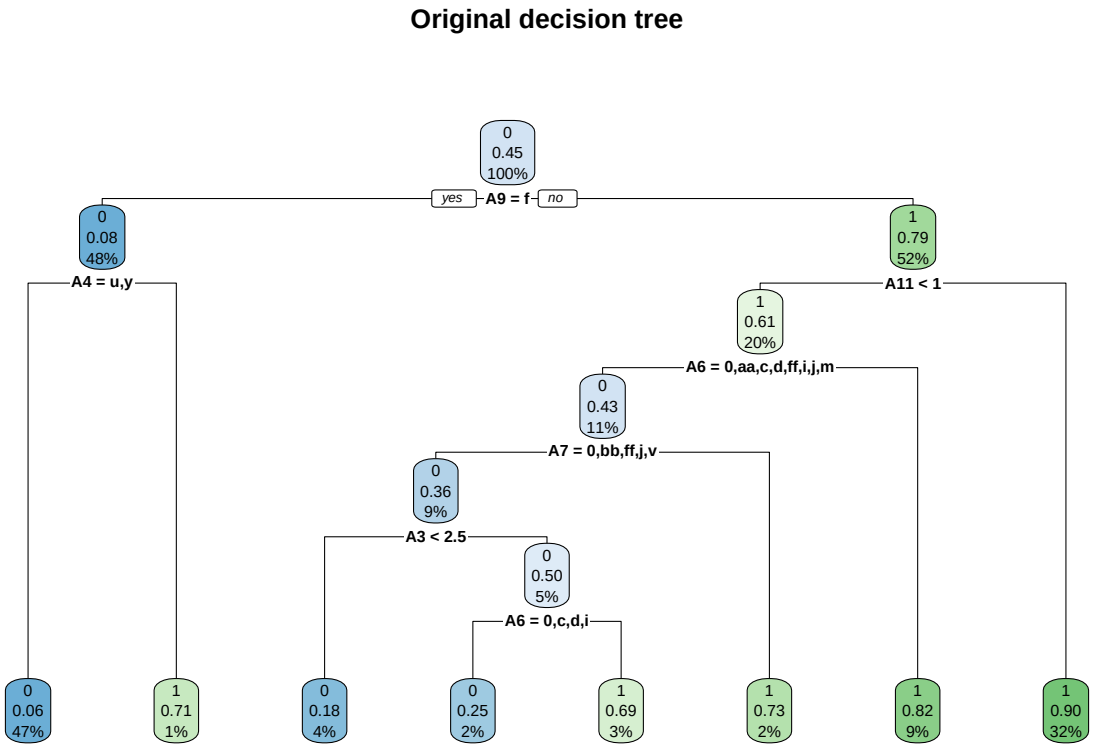
Classification using rpart

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

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

```
library(rpart)

set.seed(12345)
```

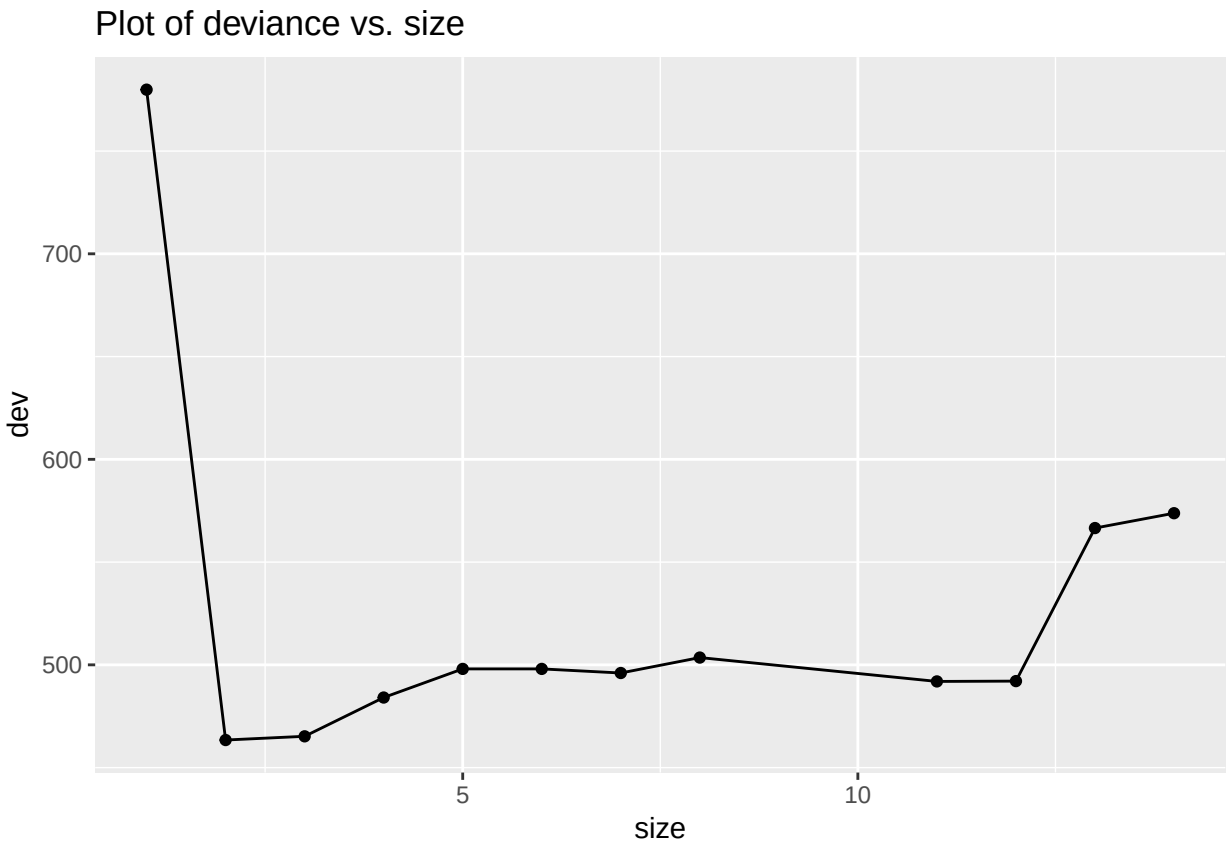
```
decision_tree_rpart <- rpart::rpart(data = train, formula = Class~., method = "class")
rpart.plot::rpart.plot(decision_tree_rpart, main= "Original decision tree")
```



prune the tree using cross validation

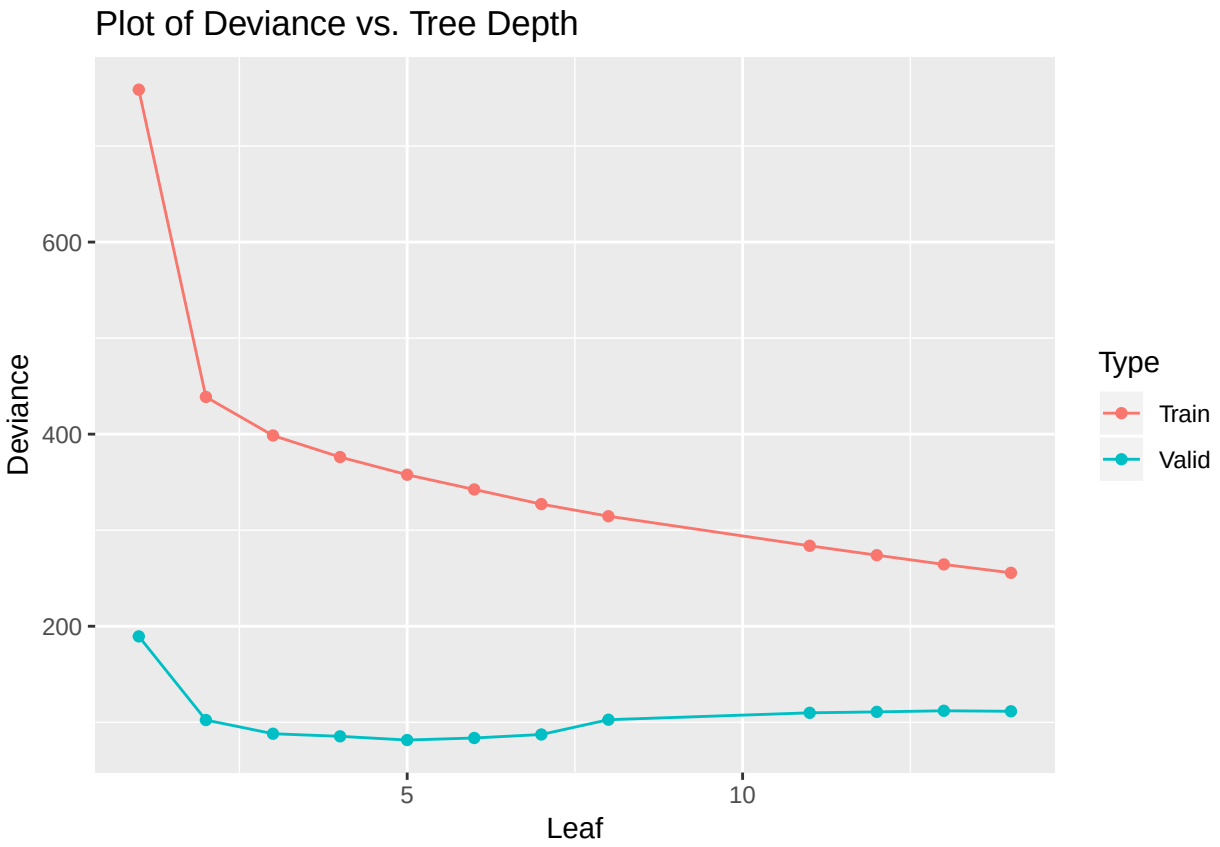
```
library(ggplot2)

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)
```


```
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")
```



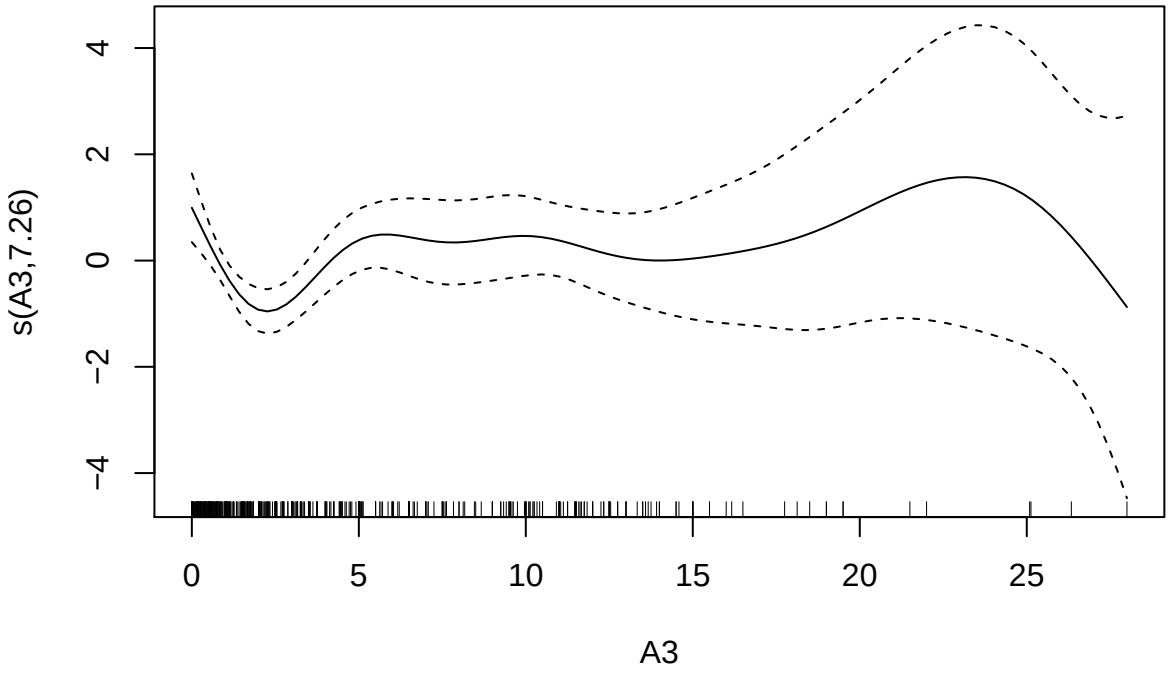
GAM Model or Spline for classification

```
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
```

```
plot(gam_model)
```



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

```
data(spam)
spam$type <- as.factor(spam$type)

## create test and training set
n=nrow(spam)
id=sample(1:n, floor(n*0.8))
spamtrain=spam[id,]
spamtest=spam[-id,]


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

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    2174   52
##     spam        112 1342
##
##                Accuracy : 0.9554
##                  95% CI : (0.9483, 0.9619)
##     No Information Rate : 0.6212
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9061
##  Mcnemar's Test P-Value : 4.083e-06
##
##             Sensitivity : 0.9510
##             Specificity : 0.9627
##          Pos Pred Value : 0.9766
##          Neg Pred Value : 0.9230
##              Prevalence : 0.6212
##          Detection Rate : 0.5908
##    Detection Prevalence : 0.6049
##       Balanced Accuracy : 0.9569
##
##        'Positive' Class : nonspam
##
```

ADA boost or ensemble for classification

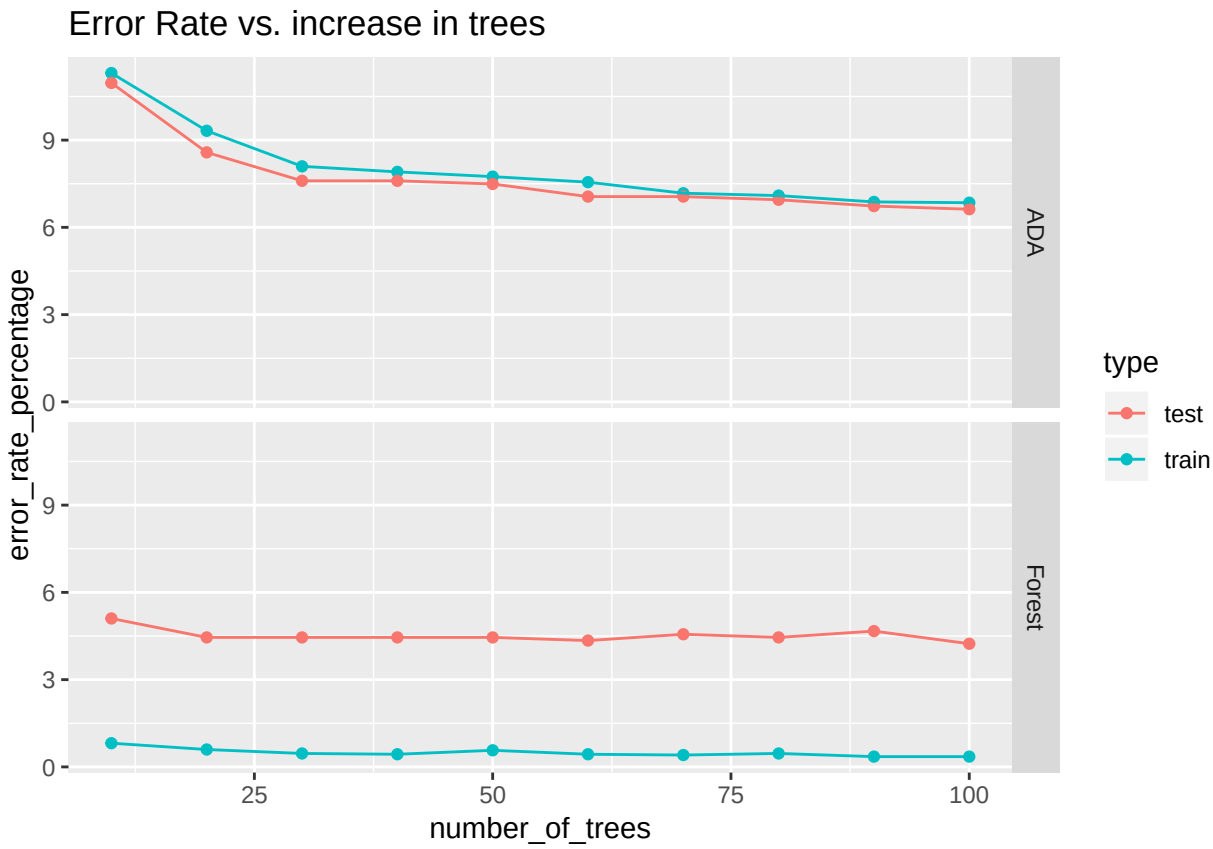
```
data(spam)
## create test and training set
n=nrow(spam)
id=sample(1:n, floor(n*0.8))
spamtrain=spam[id,]
spamtest=spam[-id,]

ada_model <- mboost::blackboost(type~., data = spamtrain, family = AdaExp(),
                                control=boost_control(mstop=15))
test_ada_model_predict <- predict(ada_model, newdata = spamtest, type = c("class"))
```

Random Forest for classification

```
forest_model <- randomForest(type~., data = spamtrain, ntree = 15)
test_forest_model_predict <- predict(forest_model, newdata = spamtest, type = c("class"))
```

Comparing ADA boost and Randomforest

```
# using warnings = FALSE

final_result <- NULL

for(i in seq(from = 10, to = 100, by = 10)){
ada_model <- mboost::blackboost(type~.,
data = spamtrain,
family = AdaExp(),
control=boost_control(mstop=i))

forest_model <- randomForest(type~., data = spamtrain, ntree = i)

prediction_function <- function(model, data){
predicted <- predict(model, newdata = data, type = c("class"))
predict_correct <- ifelse(data$type == predicted, 1, 0)
score <- sum(predict_correct)/NROW(data)
return(score)
}

train_ada_model_predict <- predict(ada_model, newdata = spamtrain, type = c("class"))
test_ada_model_predict <- predict(ada_model, newdata = spamtest, type = c("class"))
train_forest_model_predict <- predict(forest_model, newdata = spamtrain, type = c("class"))
test_forest_model_predict <- predict(forest_model, newdata = spamtest, type = c("class"))

test_predict_correct <- ifelse(spamtest$type == test_forest_model_predict, 1, 0)
train_predict_correct <- ifelse(spamtest$type == train_forest_model_predict, 1, 0)

train_ada_score <- prediction_function(ada_model, spamtrain)
test_ada_score <- prediction_function(ada_model, spamtest)
train_forest_score <- prediction_function(forest_model, spamtrain)
test_forest_score <- prediction_function(forest_model, spamtest)

iteration_result <- data.frame(number_of_trees = i,
accuracy = c(train_ada_score,
test_ada_score,
train_forest_score,
test_forest_score),
type = c("train", "test", "train", "test"),
model = c("ADA", "ADA", "Forest", "Forest"))
final_result <- rbind(iteration_result, final_result)
}

final_result$error_rate_percentage <- 100*(1 - final_result$accuracy)

ggplot(data = final_result, aes(x = number_of_trees,
y = error_rate_percentage,
group = type, color = type)) +
geom_point() +
geom_line() +
ggtitle("Error Rate vs. increase in trees") +
facet_grid(rows = vars(model))
```


Error Rate vs. increase in trees
error_rate_percentage
number_of_trees
ADA
Forest
type
test
train
0
3
6
9
25
50
75
100