Appendix

R code and outputs

```
library(pROC) # ROC
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(glmnet) # LASSO and Ridge
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-4
```

```
library(MVN) # multivariateQQPlot
library(car) # univariateQQPlot
```

```
## Loading required package: carData
```

```
library(MASS) # lda, qda
library(e1071) # Naive Bayes
library(class) # knn
library(tree) # tree
library(randomForest) # bagging and Random Forest
```

```
## randomForest 4.7-1.1

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(gbm) # Boosting
```

```
## Loaded gbm 2.1.8.1
```

```
library(huxtable)
```

```
##
## Attaching package: 'huxtable'

## The following object is masked from 'package:ggplot2':
##
##     theme_grey
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## v purrr   0.3.5
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::add_rownames()  masks huxtable::add_rownames()
## x dplyr::combine()       masks randomForest::combine()
## x tidyr::expand()        masks Matrix::expand()
## x dplyr::filter()        masks stats::filter()
## x dplyr::lag()           masks stats::lag()
## x purrr::lift()          masks caret::lift()
## x randomForest::margin() masks ggplot2::margin()
## x tidyr::pack()          masks Matrix::pack()
## x dplyr::recode()        masks car::recode()
## x dplyr::select()        masks MASS::select()
## x purrr::some()          masks car::some()
## x huxtable::theme_grey() masks ggplot2::theme_grey()
## x tidyr::unpack()        masks Matrix::unpack()
```

```r
draw_confusion_matrix <- function(cm) {

  layout(matrix(c(1,1,2)))
  par(mar=c(2,2,2,2))
  plot(c(100, 345), c(300, 450), type = "n", xlab="", ylab="", xaxt='n', yaxt='n')
  title('CONFUSION MATRIX', cex.main=2)

  # create the matrix
  rect(150, 430, 240, 370, col='#3F97D0')
  text(195, 435, 'Not Purchased', cex=1.2)
  rect(250, 430, 340, 370, col='#F7AD50')
  text(295, 435, 'Purchased', cex=1.2)
  text(125, 370, 'Predicted', cex=1.3, srt=90, font=2)
  text(245, 450, 'Actual', cex=1.3, font=2)
  rect(150, 305, 240, 365, col='#F7AD50')
  rect(250, 305, 340, 365, col='#3F97D0')
  text(140, 400, 'Not Purchased', cex=1.2, srt=90)
  text(140, 335, 'Purchased', cex=1.2, srt=90)

  # add in the cm results
  res <- as.numeric(cm$table)
  text(195, 400, res[1], cex=1.6, font=2, col='white')
  text(195, 335, res[2], cex=1.6, font=2, col='white')
  text(295, 400, res[3], cex=1.6, font=2, col='white')
  text(295, 335, res[4], cex=1.6, font=2, col='white')
```

```
  # add in the specifics
  plot(c(100, 0), c(100, 0), type = "n", xlab="", ylab="", main = "DETAILS", xaxt='n', yaxt='n')
  text(10, 85, names(cm$byClass[1]), cex=1.2, font=2)
  text(10, 70, round(as.numeric(cm$byClass[1]), 3), cex=1.2)
  text(30, 85, names(cm$byClass[2]), cex=1.2, font=2)
  text(30, 70, round(as.numeric(cm$byClass[2]), 3), cex=1.2)
  text(50, 85, names(cm$byClass[5]), cex=1.2, font=2)
  text(50, 70, round(as.numeric(cm$byClass[5]), 3), cex=1.2)
  text(70, 85, names(cm$byClass[6]), cex=1.2, font=2)
  text(70, 70, round(as.numeric(cm$byClass[6]), 3), cex=1.2)
  text(90, 85, names(cm$byClass[7]), cex=1.2, font=2)
  text(90, 70, round(as.numeric(cm$byClass[7]), 3), cex=1.2)

  # add in the accuracy information
  text(30, 35, names(cm$overall[1]), cex=1.5, font=2)
  text(30, 20, round(as.numeric(cm$overall[1]), 3), cex=1.4)
  text(70, 35, names(cm$overall[2]), cex=1.5, font=2)
  text(70, 20, round(as.numeric(cm$overall[2]), 3), cex=1.4)
}
```

```
socialnetwork <- read.csv("Social_Network_Ads.csv")
```

```
set.seed(1)
socialnetwork$Gender <- ifelse(socialnetwork$Gender=="Female",0,1)
socialnetwork <- socialnetwork[,-1]
train.i <- sample(dim(socialnetwork)[1],320)
```

```
mean_gender <- mean(socialnetwork$Gender)
mean_purchased <-mean(socialnetwork$Purchased)
knitr:::kable(summary(socialnetwork)[,c(-1,-4)])
```

Age	EstimatedSalary
Min. :18.00	Min. : 15000
1st Qu.:29.75	1st Qu.: 43000
Median :37.00	Median : 70000
Mean :37.66	Mean : 69743
3rd Qu.:46.00	3rd Qu.: 88000
Max. :60.00	Max. :150000

```
print(c(mean_gender,mean_purchased))
```

```
## [1] 0.4900 0.3575
```

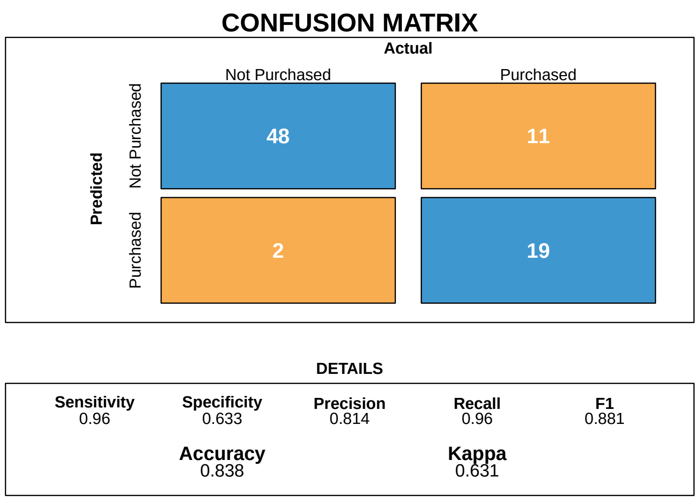
```
glm1 <- glm(Purchased~., data = socialnetwork,subset=train.i, family = "binomial")
glm1pre <- exp(predict(glm1,newdata=socialnetwork[-train.i,1:3]))

glm2 <- glm(Purchased~Age+EstimatedSalary, data = socialnetwork,subset=train.i, family = "binomial")
glm2pre <- exp(predict(glm2,newdata=socialnetwork[-train.i,1:3]))
huxreg(glm1, glm2) %>%
  set_tb_padding(0)
```

```
pre1 <- ifelse(glm1pre >= 1, 1, 0)
pre2 <- ifelse(glm2pre >= 1, 1, 0)
```

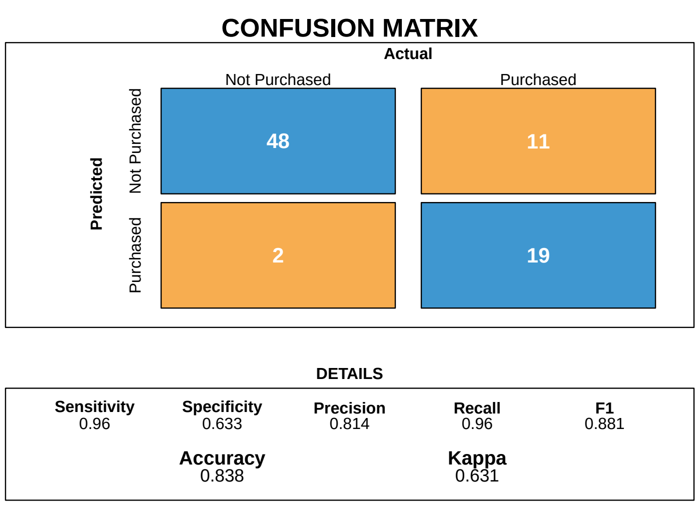
	(1)	(2)
(Intercept)	-12.625 ***	-12.499 ***
	(1.491)	(1.446)
Gender	0.128	
	(0.341)	
Age	0.243 ***	0.242 ***
	(0.029)	(0.029)
EstimatedSalary	0.000 ***	0.000 ***
	(0.000)	(0.000)
N	320	320
logLik	-110.111	-110.181
AIC	228.222	226.363

*** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

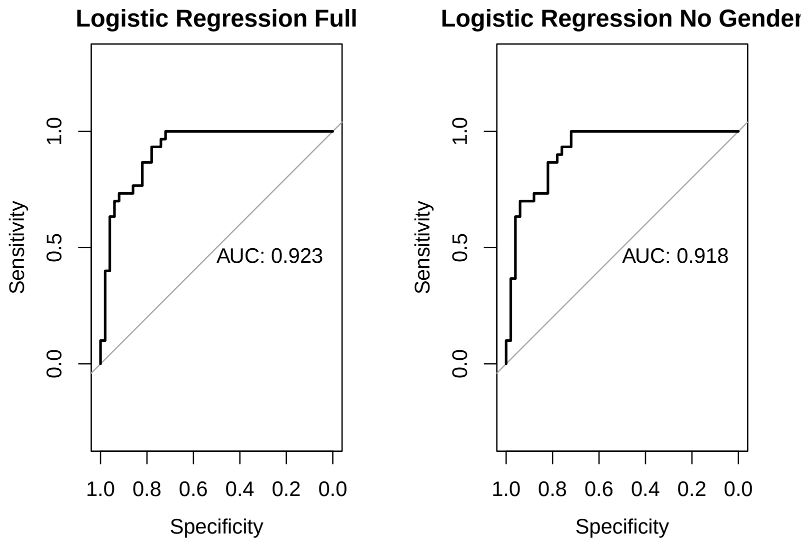
```
draw_confusion_matrix(confusionMatrix(factor(pre1), factor(socialnetwork$Purchased[-train.i])))
```




```
draw_confusion_matrix(confusionMatrix(factor(pre2), factor(socialnetwork$Purchased[-train.i])))
```



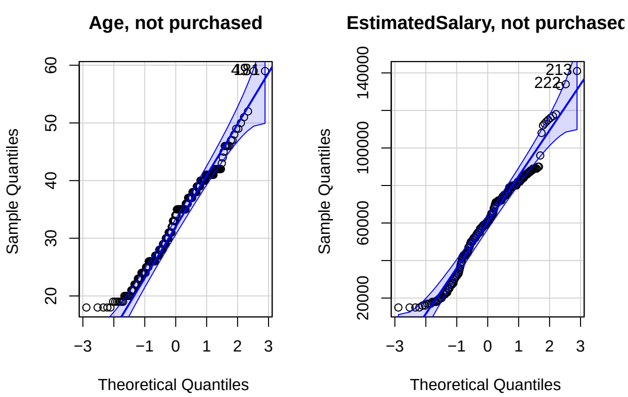
```r
par(mfrow=c(1,2))
plot.roc(socialnetwork$Purchased[-train.i], glm1pre, print.auc= T,
    main = "Logistic Regression Full")
plot.roc(socialnetwork$Purchased[-train.i], glm2pre, print.auc= T,
    main = "Logistic Regression No Gender")
```



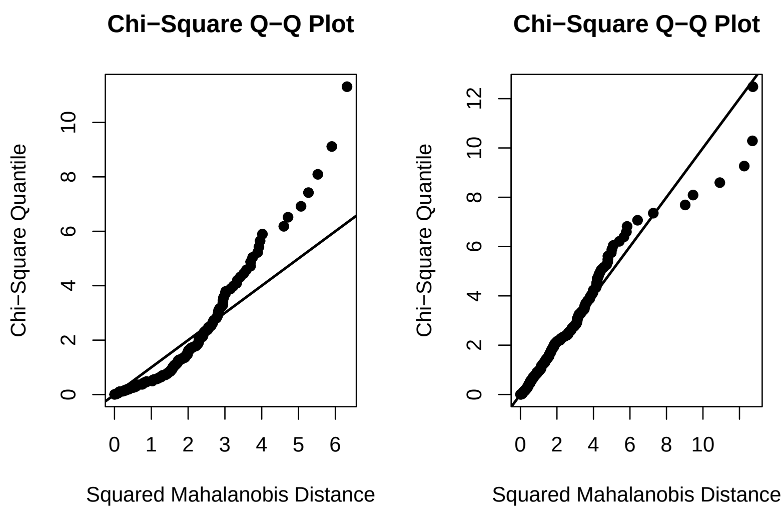

```r
decisionplot <- function(model, data ,class = NULL, predict_type = "class",
                         resolution = 100, showgrid = TRUE){
  if (!is.null(class)) cl <- data[,class] else cl <- 1
  data <- data[,c(2,3)]
  k <- length(unique(cl))
  plot(data, col = as.integer(cl)+1L, pch = as.integer(cl)+1L)

  r <- sapply(data, range, na.rm = T)
  xs <- seq(r[1,1], r[2,1], length.out = resolution)
  ys <- seq(r[1,2], r[2,2], length.out = resolution)
  g <- cbind(rep(xs, each = resolution), rep(ys, time = resolution))
  colnames(g) <- colnames(r)
  g <- as.data.frame(g)
  p <- predict(model, g, type = predict_type)
  if (is.list(p)) p <- p$class
  p <- as.factor(p)
  if (showgrid) points(g, col = as.integer(p)+1L, pch=".")
  z <- matrix(as.integer(p), nrow = resolution, byrow = T)
  contour(xs, ys, z, add = T, drawlabels = F, lwd= 2, levels = (1:(k-1))+0.5)
  invisible(z)
}
```

```r
par(mfrow=c(1,2))
invisible(capture.output(qqPlot(socialnetwork$Age[socialnetwork$Purchased==1], xlab = "Theoretical Quan
       ylab = "Sample Quantiles ", main ="Age, purchased")))

invisible(capture.output(qqPlot(socialnetwork$EstimatedSalary[socialnetwork$Purchased==1], xlab = "Theo
       ylab = "Sample Quantiles ", main ="EstimatedSalary, purchased")))
```

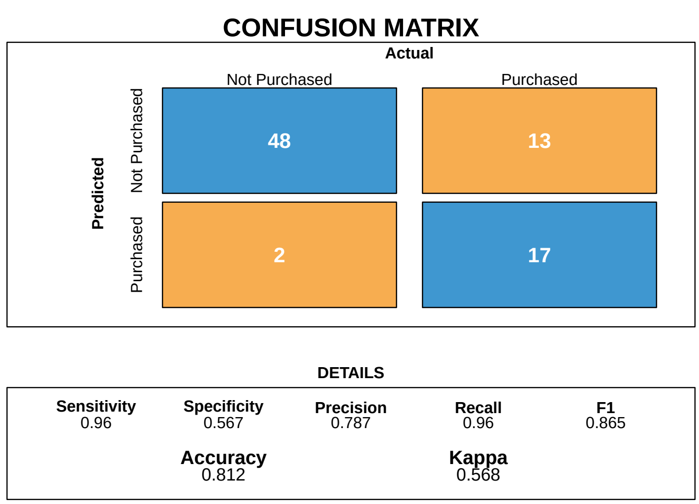

```
invisible(capture.output(qqPlot(socialnetwork$Age[socialnetwork$Purchased==0], xlab = "Theoretical Quan
       ylab = "Sample Quantiles ", main ="Age, not purchased")))

invisible(capture.output(qqPlot(socialnetwork$EstimatedSalary[socialnetwork$Purchased==0], xlab = "Theo
       ylab = "Sample Quantiles ", main ="EstimatedSalary, not purchased")))
```



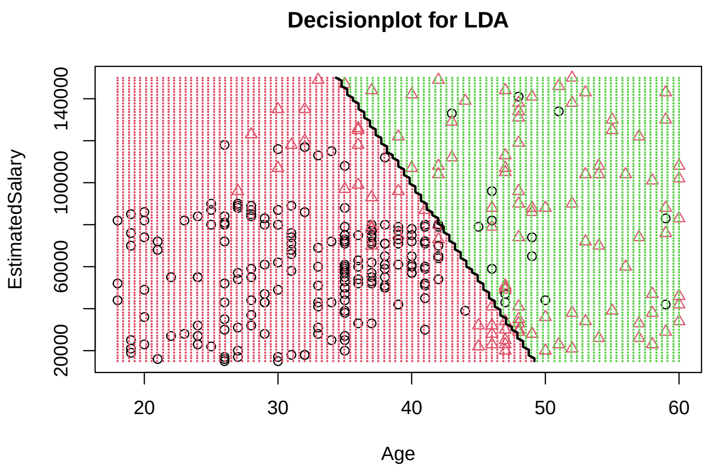

```
par(mfrow=c(1,2))
invisible(capture.output(mvn(cbind(socialnetwork$Age[socialnetwork$Purchased==1],
          socialnetwork$EstimatedSalary[socialnetwork$Purchased==1]),
    multivariatePlot="qq")))
invisible(capture.output(mvn(cbind(socialnetwork$Age[socialnetwork$Purchased==0],
          socialnetwork$EstimatedSalary[socialnetwork$Purchased==0]),
    multivariatePlot="qq")))
```



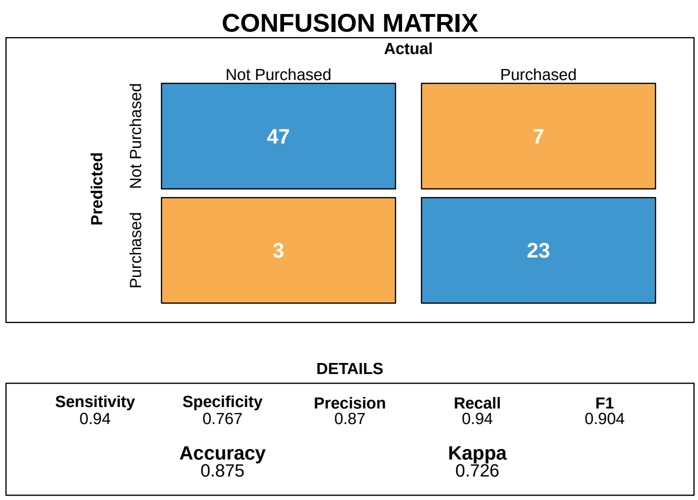

```
lda.fit = lda(Purchased ~ Age+EstimatedSalary, data=socialnetwork, subset = train.i)
lda.pred = predict(lda.fit,  socialnetwork[-train.i,][,-4])$class
lda.pred.prob = predict(lda.fit,  socialnetwork[-train.i,][,-4])$posterior[,2]
lda.error = mean(lda.pred != socialnetwork$Purchased[-train.i])
draw_confusion_matrix(confusionMatrix(lda.pred, factor(socialnetwork$Purchased[-train.i])))
```



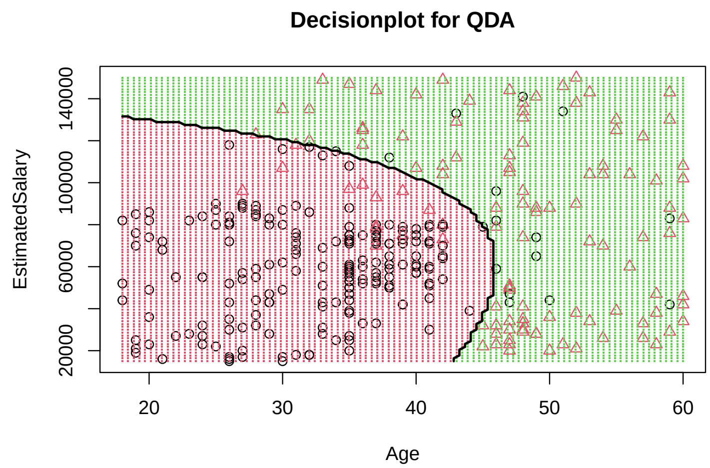

```
decisionplot(lda.fit, data = socialnetwork[train.i,], class = "Purchased")
title("Decisionplot for LDA")
```




```
qda.fit = qda(Purchased ~ Age+EstimatedSalary, data=socialnetwork, subset = train.i)
qda.pred = predict(qda.fit,  socialnetwork[-train.i,][,-4])$class
qda.pred.prob = predict(qda.fit,  socialnetwork[-train.i,][,-4])$posterior[,2]
qda.error = mean(qda.pred != socialnetwork$Purchased[-train.i])
draw_confusion_matrix(confusionMatrix(qda.pred, factor(socialnetwork$Purchased[-train.i])))
```

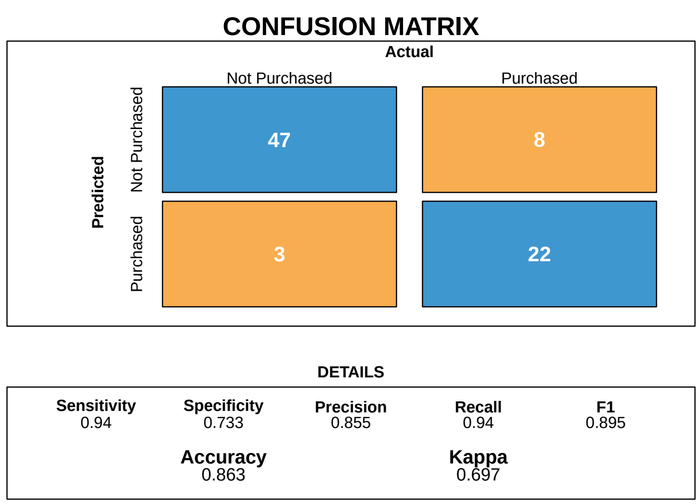



```
decisionplot(qda.fit, data = socialnetwork[train.i,], class = "Purchased")
title("Decisionplot for QDA")
```

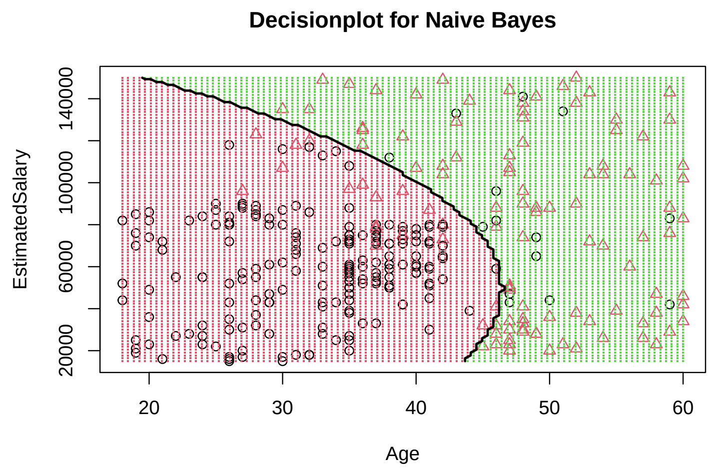



```r
nb.fit=naiveBayes(Purchased ~ Age+EstimatedSalary,data=socialnetwork,subset=train.i)
```

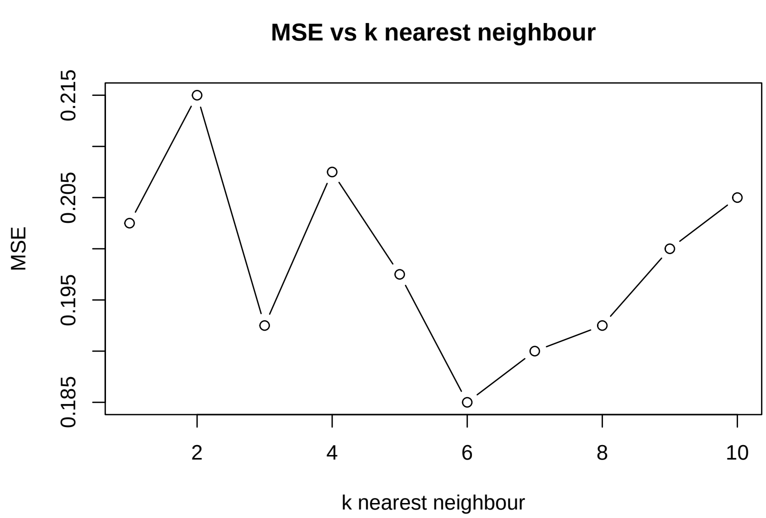
```r
nb.class=predict(nb.fit,socialnetwork)[-train.i]
nb.class.prob=predict(nb.fit,socialnetwork, type= c("raw"))[-train.i,2]
nb.error = mean(nb.class != socialnetwork$Purchased[-train.i])
draw_confusion_matrix(confusionMatrix(nb.class, factor(socialnetwork$Purchased[-train.i])))
```



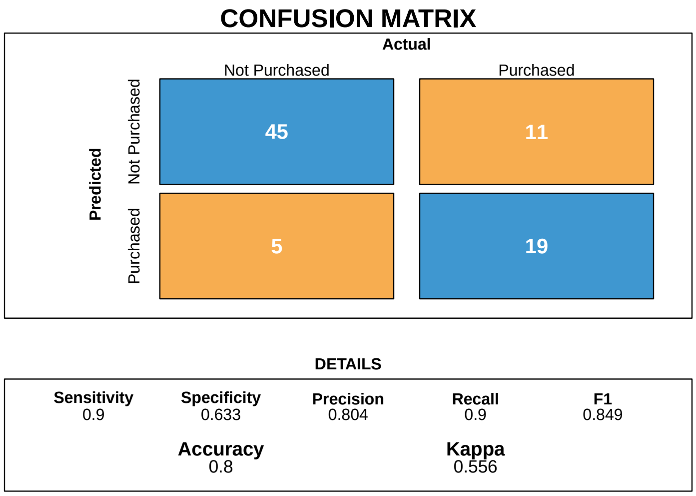

```r
decisionplot(nb.fit, data = socialnetwork[train.i,], class = "Purchased")
title("Decisionplot for Naive Bayes")
```



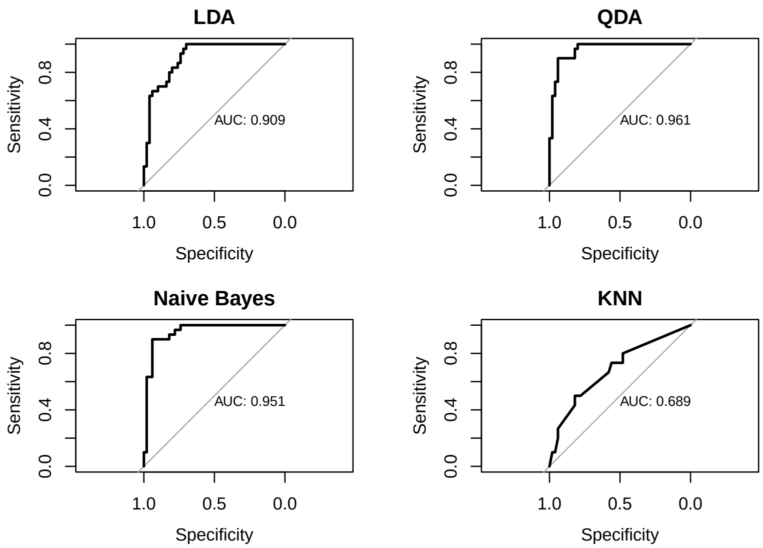

```
set.seed(441)
fivefoldcv = matrix(NA, 5, 10)
randomseq = c(1:nrow(socialnetwork))[order(runif(nrow(socialnetwork)))]
fold = c(1:nrow(socialnetwork) %% 5 + 1)
for (i in 1:5){
  train = randomseq[fold!=i]
  train.X = socialnetwork[,1:3][train,]
  test.X = socialnetwork[,1:3][-train,]
  train.y = socialnetwork[,4][train]
  test.y = socialnetwork[,4][-train]
  for (j in 1:10){
    knn.pred = knn(train.X, test.X, train.y, k=j)
    fivefoldcv[i,j]=mean(knn.pred!=test.y)
  }
}
fivefold=apply(fivefoldcv, MARGIN=2, FUN=mean)
plot(fivefold,type ="b",main="MSE vs k nearest neighbour",
     xlab = "k nearest neighbour", ylab = "MSE")
```



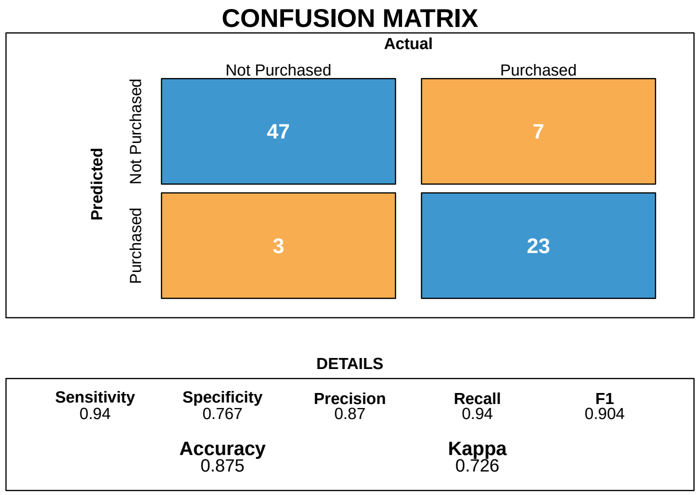
```
knn.pred = knn(socialnetwork[train.i,-4], socialnetwork[-train.i,-4], socialnetwork$Purchased[train.i],
knn.pred.prob = knn(socialnetwork[train.i,-4], socialnetwork[-train.i,-4], socialnetwork$Purchased[trai
knn.error = mean(knn.pred != socialnetwork$Purchased[-train.i])
draw_confusion_matrix(confusionMatrix(knn.pred, factor(socialnetwork$Purchased[-train.i])))
```



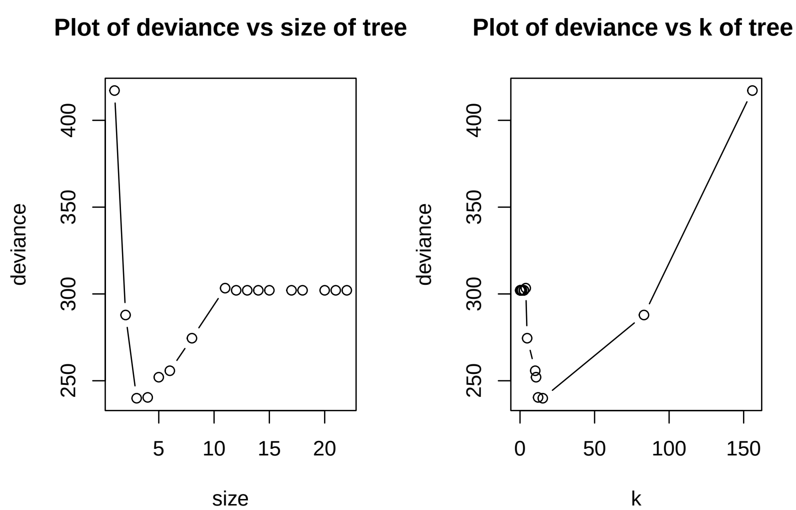

```
par(mfrow=c(2,2))
plot.roc(socialnetwork$Purchased[-train.i], lda.pred.prob, print.auc= T,main = "LDA")
plot.roc(socialnetwork$Purchased[-train.i], qda.pred.prob, print.auc= T,main = "QDA")
plot.roc(socialnetwork$Purchased[-train.i], nb.class.prob, print.auc= T,main = "Naive Bayes")
plot.roc(socialnetwork$Purchased[-train.i], attr(knn.pred.prob,"prob"), print.auc= T,main = "KNN")
```




```
CART <- tree(as.factor(Purchased)~., data = socialnetwork, subset = train.i, split = "gini")
tree.pre  <- predict(CART,socialnetwork[-train.i,], type = "class")

draw_confusion_matrix(confusionMatrix(tree.pre, factor(socialnetwork$Purchased[-train.i])))
```

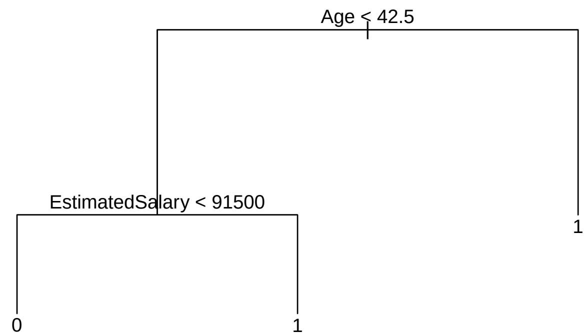



```r
set.seed(441)
result <- cv.tree(CART, FUN = prune.tree, K=5)
par(mfrow=c(1,2))
plot(result$size, result$dev, type="b", xlab="size", ylab="deviance",
     main ="Plot of deviance vs size of tree")
plot(result$k, result$dev, type="b", xlab="k", ylab="deviance" ,
     main = "Plot of deviance vs k of tree ")
```

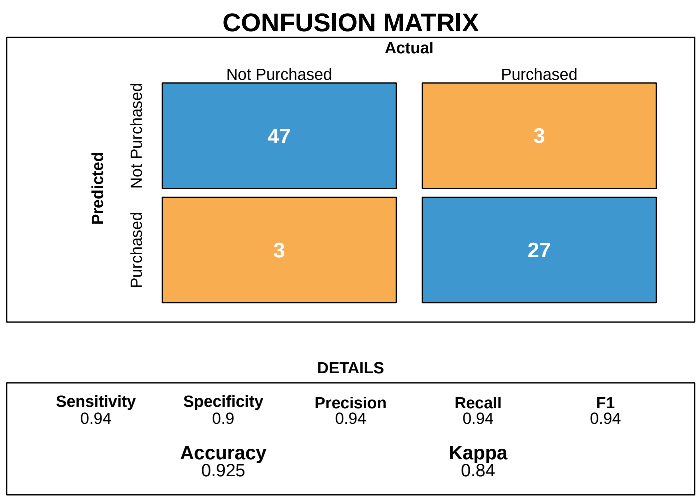



```r
new.tree <- prune.tree(CART, best=result$size[which.min(result$dev)])
par(mfrow=c(1,1))
```

```r
plot(new.tree)
text(new.tree)
```



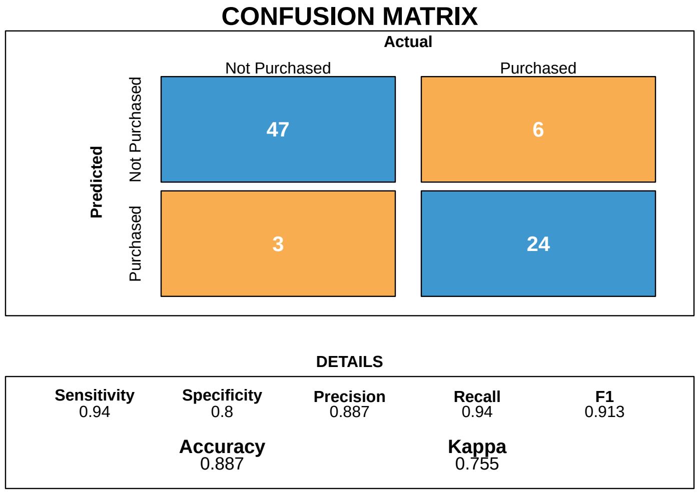

```
tree.pre  <- predict(new.tree,socialnetwork[-train.i,], type = "class")
draw_confusion_matrix(confusionMatrix(tree.pre, factor(socialnetwork$Purchased[-train.i])))
```




```
mean(tree.pre != socialnetwork[-train.i,]$Purchased)
```

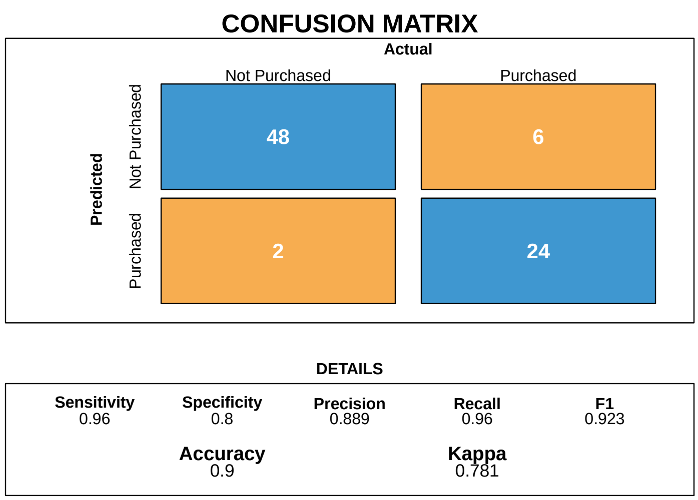
```
## [1] 0.075
```

```
set.seed(441)
bagging <- randomForest(as.factor(Purchased)~., data=socialnetwork[train.i,], mtry=3, ntree = 200,
             importance = T, xtest=socialnetwork[-train.i,][,-4],
             ytest=factor(socialnetwork$Purchased[-train.i]), keep.forest=T)
draw_confusion_matrix(confusionMatrix(bagging$test$predicted, factor(socialnetwork$Purchased[-train.i])
```



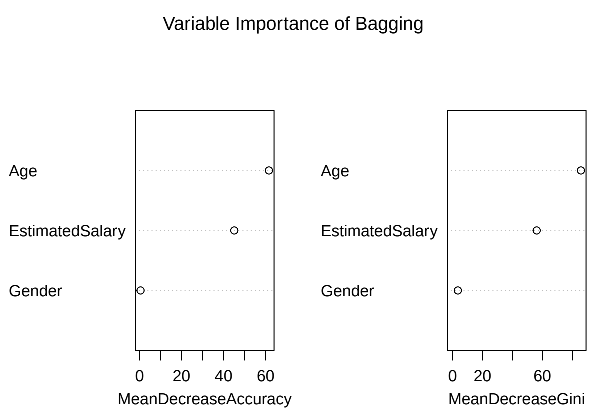

```
#bagging
```

```
set.seed(441)
rf2 <- randomForest(as.factor(Purchased)~., data=socialnetwork[train.i,], mtry=2, ntree = 200,
             importance = T, xtest=socialnetwork[-train.i,][,-4],
             ytest=factor(socialnetwork$Purchased[-train.i]), keep.forest=T)
draw_confusion_matrix(confusionMatrix(rf2$test$predicted, factor(socialnetwork$Purchased[-train.i])))
```

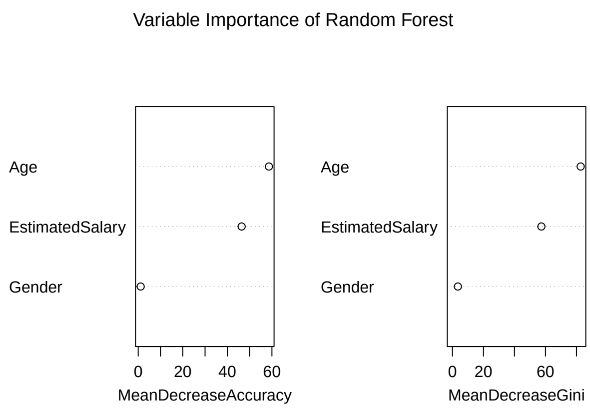



```
#rf2
```

```
varImpPlot(bagging, main = "Variable Importance of Bagging")
```

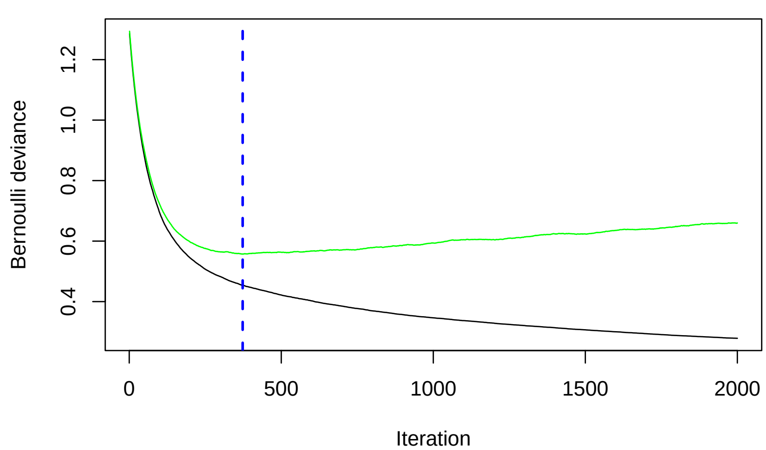



```
varImpPlot(rf2, main = "Variable Importance of Random Forest")
```



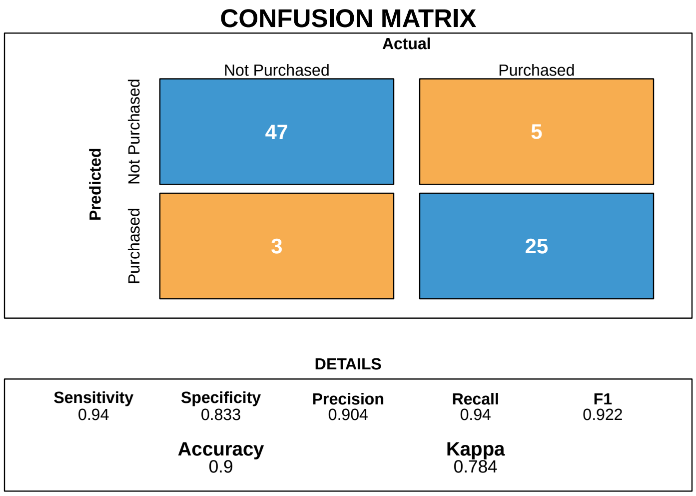

```
set.seed(441)
boosting <- gbm(as.numeric(Purchased)~., data= socialnetwork[train.i, ], distribution="bernoulli",
    n.trees=2000, interaction.depth = 2, shrinkage = 0.01, cv.folds = 5 )
```

```
ntree <- gbm.perf(boosting, method = "cv")
```



```r
preboosting <- predict(boosting, n.tree = ntree, newdata=socialnetwork[-train.i, ], type = "response")
pred.class.boosting <- ifelse(preboosting >= 0.5, 1,0)
```

```r
draw_confusion_matrix(confusionMatrix(factor(pred.class.boosting), factor(socialnetwork$Purchased[-trai
```

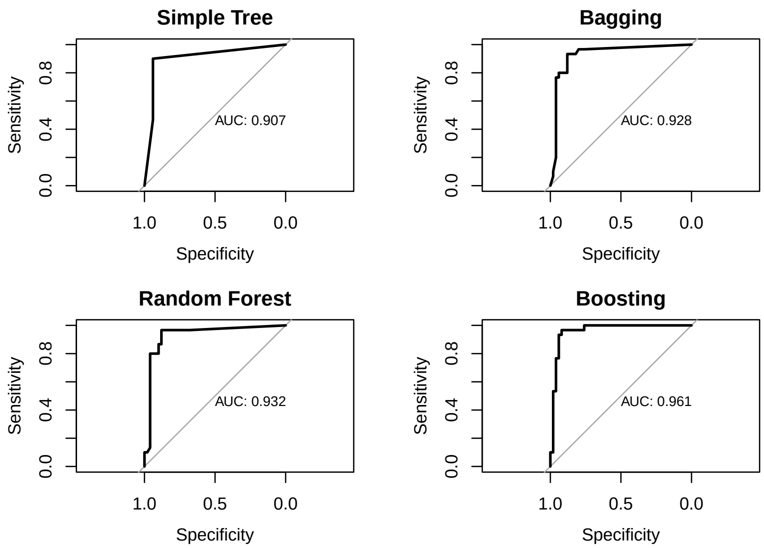



```r
mean(pred.class.boosting != socialnetwork[-train.i,]$Purchased)
```

```
## [1] 0.1
```

```r
pretree <- predict(new.tree, socialnetwork[-train.i,])[,2]
predbag <- predict(bagging, newdata = socialnetwork[-train.i,],type="prob")[,2]
prerf2 <- predict(rf2, newdata = socialnetwork[-train.i,],type="prob")[,2]
```

```r
par(mfrow=c(2,2))
plot.roc(socialnetwork$Purchased[-train.i], pretree, print.auc= T,main="Simple Tree")
plot.roc(socialnetwork$Purchased[-train.i], predbag, print.auc= T,main="Bagging")
plot.roc(socialnetwork$Purchased[-train.i], prerf2, print.auc= T,main="Random Forest")
plot.roc(socialnetwork$Purchased[-train.i], preboosting, print.auc= T,main="Boosting")
```



```r
name <- c("Logistic Regression", "LDA", "QDA", "Naive Bayes", "KNN", "Single Tree with Pruning", "Baggi
test_error <- c("16.25%", "18.75%","12.5%", "13.75%","20%","7.5%","11.25%","10%","**10%**")
AUC <- c("0.918", "0.909", "0.961", "0.951", "0.689", "0.907", "0.928", "0.932", "**0.961**")
con_ds <- data.frame(Model_Names = name, Test_Error = test_error, AUC = AUC)
```

```r
knitr::kable(con_ds)
```

Model_Names	Test_Error	AUC
Logistic Regression	16.25%	0.918
LDA	18.75%	0.909
QDA	12.5%	0.961
Naive Bayes	13.75%	0.951
KNN	20%	0.689
Single Tree with Pruning	7.5%	0.907
Bagging	11.25%	0.928
Random Forest	10%	0.932
Boosting	**10%**	**0.961**