```r
###Install and load the required packages and libraries
library(pROC)
```

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

##
## Attaching package: 'pROC'

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

```r
library(ROCR)
library(readxl)
library(tree)
library(magrittr)
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:magrittr':
##
##     extract
```

```r
library(partykit)
```

```
## Loading required package: grid

## Loading required package: libcoin

## Loading required package: mvtnorm
```

```r
library(MASS)
library(dplyr)
```

```
##
## 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
```

```r
###Data preparation
#Prepare training data
bag_data <- read_excel('auction_clean version.xlsx') #Choose auction_clean version.xlsx
head(bag_data)
```

```
## # A tibble: 6 x 14
##   brand  bag                ‘Production Ye~‘ status Location ‘Auction year‘ color
##   <chr>  <chr>              <chr>            <chr>  <chr>              <dbl> <chr>
## 1 Hermès Nata Clemence Mi~ 2021              Biddi~ Hong Ko~            2022 non-~
## 2 Hermès Nata Swift In & ~ 2021              Biddi~ Hong Ko~            2022 non-~
## 3 Hermès White Matte Nilo~ 2014              Biddi~ Hong Ko~            2022 non-~
## 4 Hermès Nata Chèvre Myso~ 2021              Biddi~ Hong Ko~            2022 non-~
## 5 Hermès Metallic Silver ~ 2005              Biddi~ Hong Ko~            2022 non-~
## 6 Hermès Noir Swift and C~ 2021              Biddi~ Hong Ko~            2022 non-~
## # ... with 7 more variables: special_leather <dbl>, birkin <dbl>,
## #   over_sold <dbl>, lowerestimate_USD <dbl>, upperestimate_USD <dbl>,
## #   soldprice_USD <dbl>, amount_oversold <dbl>
```

```r
colnames(bag_data)
```

```
##  [1] "brand"             "bag"               "Production Year"
##  [4] "status"            "Location"          "Auction year"
##  [7] "color"             "special_leather"   "birkin"
## [10] "over_sold"         "lowerestimate_USD" "upperestimate_USD"
## [13] "soldprice_USD"     "amount_oversold"
```

```r
class(bag_data$soldprice_USD)
```

```
## [1] "numeric"
```

```r
class(bag_data$brand)
```

```
## [1] "character"
```

```r
bag_data$special_leather[is.na(bag_data$special_leather)] <- 0
Hermes <- ifelse(bag_data$brand == "Hermès", 1, 0)
Chanel <- ifelse(bag_data$brand == "Chanel", 1, 0)
Black <- ifelse(bag_data$color == "black", 1, 0)
bag_data <- cbind(bag_data, Hermes, Chanel, Black)
bag_data <- as.data.frame(bag_data)
head(bag_data)
```

```
##    brand
## 1 Hermès
## 2 Hermès
## 3 Hermès
## 4 Hermès
## 5 Hermès
## 6 Hermès
##                                                                            bag
```

```
## 1                   Nata Clemence Mini Amazon Evelyne TPM Gold Hardware, 2021
## 2              Nata Swift In & Out Kelly 25 Retourne Palladium Hardware, 2021
## 3 White Matte Niloticus Crocodile Himalaya Birkin 30 Palladium Hardware, 2014
## 4                            Nata Chèvre Mysore Geta Palladium Hardware, 2021
## 5        Metallic Silver and Bronze Chevre Birkin 25 Palladium Hardware, 2005
## 6                                  Noir Swift and Canvas Birkin Fray 35, 2021
##   Production Year            status  Location Auction year     color
## 1            2021 Bidding is closed Hong Kong         2022 non-black
## 2            2021 Bidding is closed Hong Kong         2022 non-black
## 3            2014 Bidding is closed Hong Kong         2022 non-black
## 4            2021 Bidding is closed Hong Kong         2022 non-black
## 5            2005 Bidding is closed Hong Kong         2022 non-black
## 6            2021 Bidding is closed Hong Kong         2022 non-black
##   special_leather birkin over_sold lowerestimate_USD upperestimate_USD
## 1               0      0         1           1911.45           2803.46
## 2               0      0         1          20388.80          25486.00
## 3               1      1         1         114687.00         152916.00
## 4               0      0         0           7008.65          10194.40
## 5               0      1         1          50972.00          76458.00
## 6               0      1         0          19114.50          31857.50
##   soldprice_USD amount_oversold Hermes Chanel Black
## 1      4495.730        1692.270      1      0     0
## 2     35323.596        9837.596      1      0     0
## 3    208730.340       55814.340      1      0     0
## 4      7706.966       -2487.434      1      0     0
## 5    136477.530       60019.530      1      0     0
## 6     28901.124       -2956.376      1      0     0
```

```
bag_data <- bag_data[bag_data$brand == "Hermès" | bag_data$brand == "Chanel", ]
bag_data$production_year <- as.numeric(bag_data$`Production Year`)
bag_data$auction_year <- as.numeric(bag_data$`Auction year`)
summary(bag_data)
```

```
##     brand               bag            Production Year       status
##  Length:915         Length:915         Length:915         Length:915
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##    Location          Auction year     color           special_leather
##  Length:915         Min.   :2020   Length:915         Min.   :0.0000
##  Class :character   1st Qu.:2022   Class :character   1st Qu.:0.0000
##  Mode  :character   Median :2022   Mode  :character   Median :0.0000
##                     Mean   :2022                      Mean   :0.1322
##                     3rd Qu.:2022                      3rd Qu.:0.0000
##                     Max.   :2022                      Max.   :1.0000
##
##      birkin          over_sold      lowerestimate_USD upperestimate_USD
##  Min.   :0.0000   Min.   :0.0000   Min.   :   100    Min.   :   200
##  1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:  2242    1st Qu.:  3020
##  Median :0.0000   Median :1.0000   Median :  6000    Median :  8000
##  Mean   :0.1989   Mean   :0.7648   Mean   :  8979    Mean   : 11781
```

```
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.: 12000    3rd Qu.: 15292
##  Max.   :1.0000   Max.   :1.0000   Max.   :120000    Max.   :152916
##                   NA's   :35
##  soldprice_USD    amount_oversold       Hermes           Chanel
##  Min.   :   126   Min.   :-7697.7   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:  3780   1st Qu.:   37.8   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :  8831   Median :  868.7   Median :1.0000   Median :0.0000
##  Mean   : 14156   Mean   : 2325.9   Mean   :0.7082   Mean   :0.2918
##  3rd Qu.: 18900   3rd Qu.: 2851.1   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :208730   Max.   :60019.5   Max.   :1.0000   Max.   :1.0000
##  NA's   :35       NA's   :35
##      Black        production_year  auction_year
##  Min.   :0.0000   Min.   :1956    Min.   :2020
##  1st Qu.:0.0000   1st Qu.:2009    1st Qu.:2022
##  Median :0.0000   Median :2016    Median :2022
##  Mean   :0.1137   Mean   :2013    Mean   :2022
##  3rd Qu.:0.0000   3rd Qu.:2021    3rd Qu.:2022
##  Max.   :1.0000   Max.   :2022    Max.   :2022
##                   NA's   :121
```

```
#Prepare testing data
bag_prediction <- read_excel('auction_prediction.xlsx') #Choose auction_prediction.xlsx
head(bag_prediction)
```

```
## # A tibble: 6 x 15
##   brand bag   production_year status location auction_year color special_leather
##   <chr> <chr>           <dbl> <chr>  <chr>           <dbl> <chr>           <dbl>
## 1 Herm~ Whit~            2020 Biddi~ Hong Ko~         2022 non-~               1
## 2 Herm~ Whit~            2021 Biddi~ Hong Ko~         2022 non-~               1
## 3 Herm~ Gris~            2021 Biddi~ Hong Ko~         2022 non-~               1
## 4 Herm~ Beto~            2020 Biddi~ Hong Ko~         2022 non-~              NA
## 5 Herm~ Nata~            2022 Biddi~ Hong Ko~         2022 non-~              NA
## 6 Herm~ Limi~            2021 Biddi~ Hong Ko~         2022 non-~              NA
## # ... with 7 more variables: birkin <dbl>, oversold <dbl>,
## #   lowerestimate_USD <dbl>, upperestimate_USD <dbl>, soldprice_USD <dbl>,
## #   amount_oversold <dbl>, exchange_rate <dbl>
```

```
colnames(bag_prediction)
```

```
##  [1] "brand"             "bag"               "production_year"
##  [4] "status"            "location"          "auction_year"
##  [7] "color"             "special_leather"   "birkin"
## [10] "oversold"          "lowerestimate_USD" "upperestimate_USD"
## [13] "soldprice_USD"     "amount_oversold"   "exchange_rate"
```

```
class(bag_prediction$soldprice_USD)
```

```
## [1] "numeric"
```

```
class(bag_prediction$brand)
```

```
## [1] "character"
```

```r
bag_prediction$special_leather[is.na(bag_prediction$special_leather)] <- 0
Hermes <- ifelse(bag_prediction$brand == "Hermès", 1, 0)
Chanel <- ifelse(bag_prediction$brand == "Chanel", 1, 0)
Black <- ifelse(bag_prediction$color == "black", 1, 0)
bag_prediction <- cbind(bag_prediction, Hermes, Chanel, Black)
bag_prediction <- as.data.frame(bag_prediction)
head(bag_prediction)
```

```
##    brand
## 1 Hermès
## 2 Hermès
## 3 Hermès
## 4 Hermès
## 5 Hermès
## 6 Hermès
##                                                                                  bag
## 1          White Matte Niloticus Crocodile Himalaya Birkin 30 Palladium Hardware, 2020
## 2 White Matte Niloticus Crocodile Himalaya Kelly 25 Retourné Palladium Hardware, 2021
## 3        Gris Perle and Kraft Matte Alligator Mini Kelly II 20 HSS Gold Hardware, 2021
## 4                Beton and Abricot Clemence Birkin 25 HSS Brushed Gold Hardware, 2020
## 5                                    Nata Epsom Kelly 28 Sellier Gold Hardware, 2022
## 6       Limited Edition Nata Swift In & Out Kelly 25 Retourne Palladium Hardware, 2021
##   production_year            status  location auction_year     color
## 1            2020 Bidding is closed Hong Kong         2022 non-black
## 2            2021 Bidding is closed Hong Kong         2022 non-black
## 3            2021 Bidding is closed Hong Kong         2022 non-black
## 4            2020 Bidding is closed Hong Kong         2022 non-black
## 5            2022 Bidding is closed Hong Kong         2022 non-black
## 6            2021 Bidding is closed Hong Kong         2022 non-black
##   special_leather birkin oversold lowerestimate_USD upperestimate_USD
## 1               1      1        0         114655.52         152874.03
## 2               1      0        1         127395.03         191092.54
## 3               1      0        1          50958.01          63697.51
## 4               0      1        0          19109.25          31848.76
## 5               0      0        1          11465.55          19109.25
## 6               0      0        0          22931.10          33122.71
##   soldprice_USD amount_oversold exchange_rate Hermes Chanel Black
## 1     152491.85       -382.1851        7.8496      1      0     0
## 2     240776.60      49684.0603            NA      1      0     0
## 3     192621.28     128923.7668            NA      1      0     0
## 4      24077.66      -7771.0966            NA      1      0     0
## 5      22472.48       3363.2287            NA      1      0     0
## 6      28893.19      -4229.5149            NA      1      0     0
```

```r
bag_prediction <- bag_prediction[bag_prediction$brand == "Hermès" | bag_prediction$brand == "Chanel", ]
bag_prediction$production_year <- as.numeric(bag_prediction$production_year)
bag_prediction$auction_year <- as.numeric(bag_prediction$auction_year)
summary(bag_prediction)
```

```
##     brand                bag             production_year    status
##  Length:94          Length:94          Min.   :2001    Length:94
##  Class :character   Class :character   1st Qu.:2017    Class :character
```

```
##  Mode  :character   Mode  :character   Median :2020    Mode  :character
##                                        Mean   :2018
##                                        3rd Qu.:2022
##                                        Max.   :2022
##                                        NA's   :2
##    location          auction_year     color            special_leather
##  Length:94          Min.   :2022   Length:94          Min.   :0.0000
##  Class :character   1st Qu.:2022   Class :character   1st Qu.:0.0000
##  Mode  :character   Median :2022   Mode  :character   Median :0.0000
##                     Mean   :2022                      Mean   :0.1915
##                     3rd Qu.:2022                      3rd Qu.:0.0000
##                     Max.   :2022                      Max.   :1.0000
##
##      birkin          oversold      lowerestimate_USD  upperestimate_USD
##  Min.   :0.0000   Min.   :0.0000   Min.   :   573.3   Min.   :   828.1
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:  4458.8   1st Qu.:  7006.7
##  Median :0.0000   Median :0.0000   Median :  7643.7   Median : 10191.6
##  Mean   :0.2447   Mean   :0.4681   Mean   : 13304.0   Mean   : 19032.7
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.: 14013.5   3rd Qu.: 20383.2
##  Max.   :1.0000   Max.   :1.0000   Max.   :127395.0   Max.   :191092.5
##
##  soldprice_USD      amount_oversold    exchange_rate      Hermes
##  Min.   :   963.1   Min.   : -7771.1   Min.   :7.85   Min.   :0.0000
##  1st Qu.:  6179.9   1st Qu.: -1220.4   1st Qu.:7.85   1st Qu.:1.0000
##  Median : 11236.2   Median :  -108.9   Median :7.85   Median :1.0000
##  Mean   : 20877.5   Mean   :  1844.9   Mean   :7.85   Mean   :0.9574
##  3rd Qu.: 20466.0   3rd Qu.:  1031.9   3rd Qu.:7.85   3rd Qu.:1.0000
##  Max.   :240776.6   Max.   :128923.8   Max.   :7.85   Max.   :1.0000
##                                        NA's   :93
##      Chanel             Black
##  Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000
##  Mean   :0.04255   Mean   :0.01064
##  3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000
##
```

```
###Linear regression

#lm for sold_price
bag_data$production_year <- as.numeric(bag_data$production_year)
bag_data$auction_year <- as.numeric(bag_data$auction_year)
price_predict_model <- lm(soldprice_USD ~ auction_year + special_leather +
                            birkin + upperestimate_USD + Hermes + Black,
                          data = bag_data)
summary(price_predict_model)
```

```
##
## Call:
## lm(formula = soldprice_USD ~ auction_year + special_leather +
##     birkin + upperestimate_USD + Hermes + Black, data = bag_data)
##
## Residuals:
```

```
##    Min     1Q Median     3Q    Max
## -14002  -1608   -406    989  51568
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -1.613e+06  5.057e+05  -3.190  0.00147 **
## auction_year         7.983e+02  2.502e+02   3.191  0.00147 **
## special_leather     -1.548e+03  5.214e+02  -2.970  0.00306 **
## birkin               1.073e+03  4.451e+02   2.410  0.01616 *
## upperestimate_USD    1.173e+00  1.342e-02  87.381  < 2e-16 ***
## Hermes              -5.353e+02  4.389e+02  -1.220  0.22288
## Black               -2.355e+02  5.646e+02  -0.417  0.67668
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4769 on 873 degrees of freedom
##   (35 observations deleted due to missingness)
## Multiple R-squared:  0.9267, Adjusted R-squared:  0.9262
## F-statistic:  1839 on 6 and 873 DF,  p-value: < 2.2e-16
```

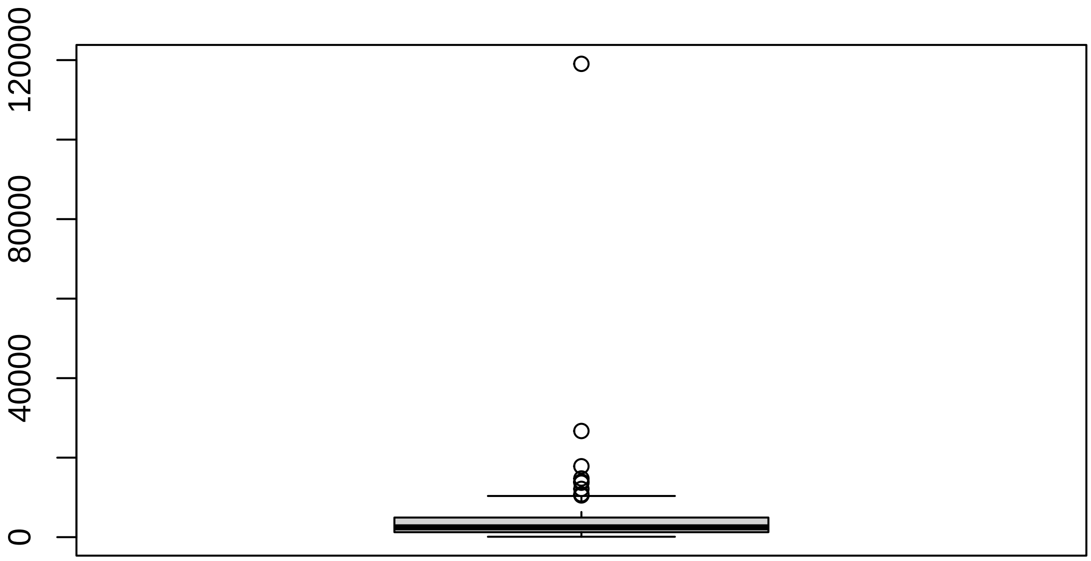
```r
predicted_sold_price <- predict(price_predict_model, newdata = bag_prediction)
bag_prediction <- cbind(bag_prediction, predicted_sold_price)
diff <- abs(bag_prediction$soldprice_USD - bag_prediction$predicted_sold_price)
bag_prediction <- cbind(bag_prediction, diff)
min(diff)
```

```
## [1] 106.0959
```

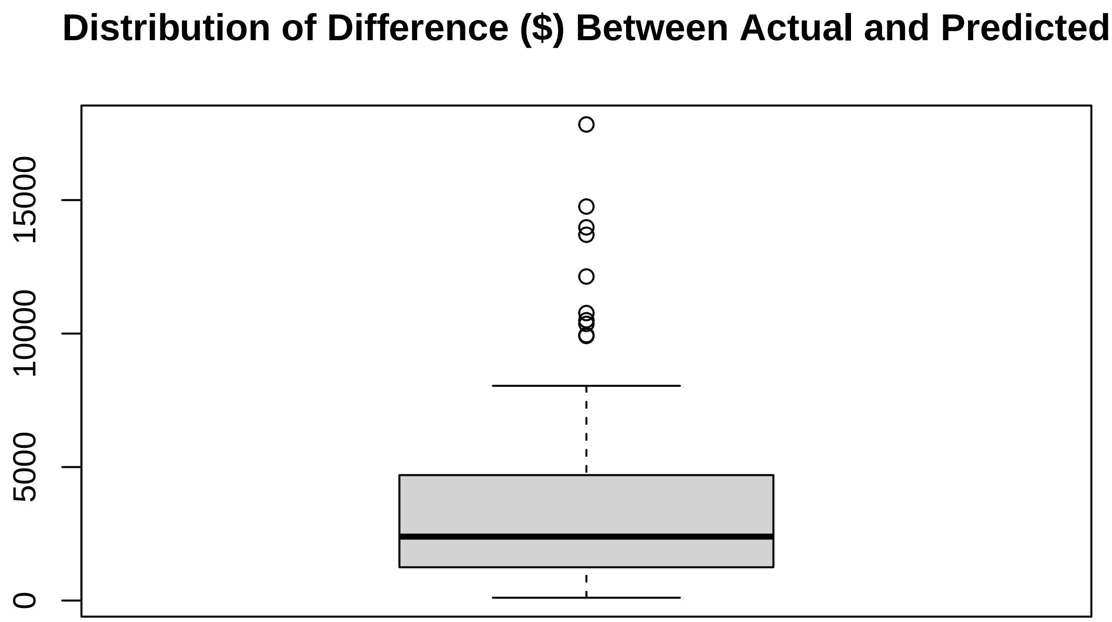
```r
max(diff)
```

```
## [1] 119060.8
```

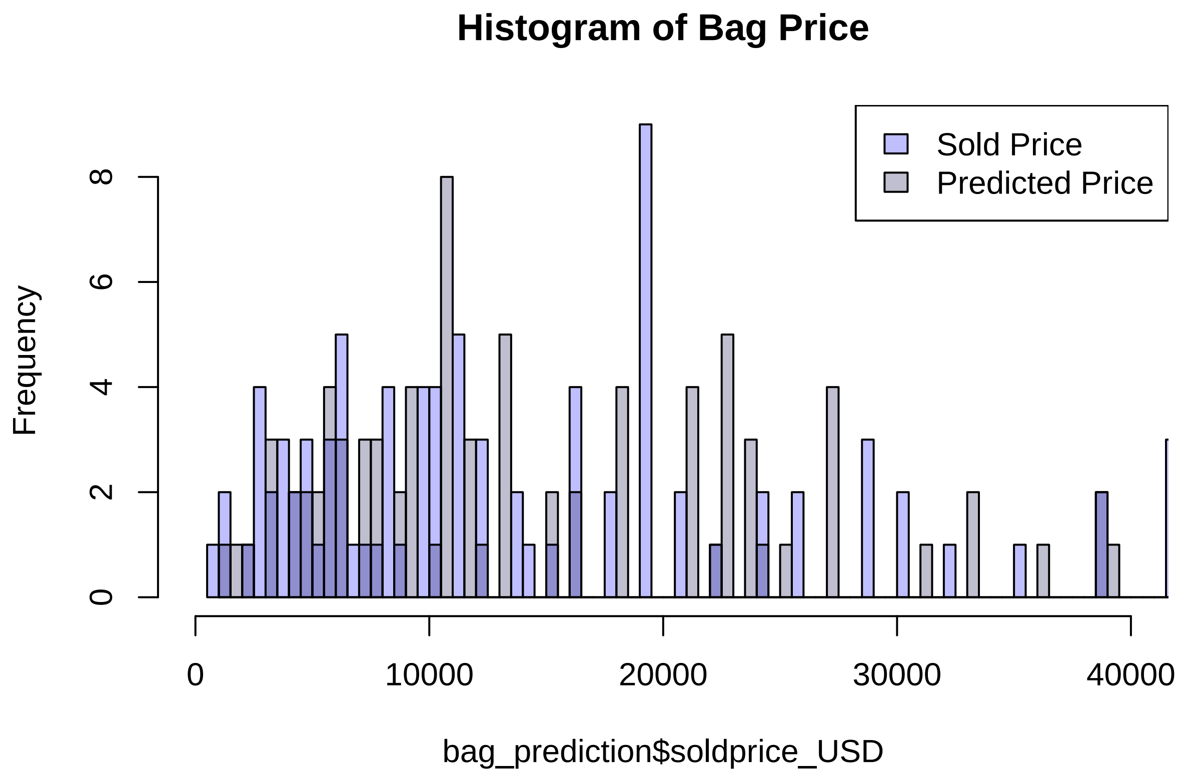
```r
boxplot(diff)
```




```r
#Drop outlier
bag_prediction <- bag_prediction[-c(1,3),]

#Plot the bloxplot of differences as shown in figure 4
boxplot(bag_prediction$diff, main = "Distribution of Difference ($) Between Actual and Predicted")
```




```
#Plot the histogram of bag prices as shown in figure 4
hgA <- hist(bag_prediction$soldprice_USD, breaks = 500,  plot = FALSE)
hgB <- hist(bag_prediction$predicted_sold_price, breaks = 500, plot = FALSE)
plot(hgA, xlim = c(0, 40000), col=rgb(0,0,1,1/4), main = "Histogram of Bag Price")
plot(hgB, xlim = c(0, 40000), col=rgb(0,0,1/4,1/4), add = TRUE)
legend('topright', c('Sold Price', 'Predicted Price'), fill=c(rgb(0,0,1,1/4), rgb(0,0,1/4,1/4)))
```




```
###5-fold Cross validation of linear regression models
nfold <- 5
n <- nrow(bag_data)
foldid <- rep(1:nfold,each=ceiling(n/nfold))[sample(1:n)]
### create an empty dataframe of results
OOS <- data.frame(linear1=rep(NA,nfold), linear2=rep(NA,nfold),linear3=rep(NA,nfold))
bag_data <- bag_data %>% drop_na()
deviance <- function(y, pred, family=c("gaussian","binomial")){
  family <- match.arg(family)
  if(family=="gaussian"){
    return( sum( (y-pred)^2 ) )
  }else{
    if(is.factor(y)) y <- as.numeric(y)>1
    return( -2*sum( y*log(pred) + (1-y)*log(1-pred) ) )
  }
}

R2 <- function(y, pred, family=c("gaussian","binomial")){
  fam <- match.arg(family)
  if(fam=="binomial"){
    if(is.factor(y)){ y <- as.numeric(y)>1 }
  }
  dev <- deviance(y, pred, family=fam)
  dev0 <- deviance(y, mean(y), family=fam)
  return(1-dev/dev0)
}
```

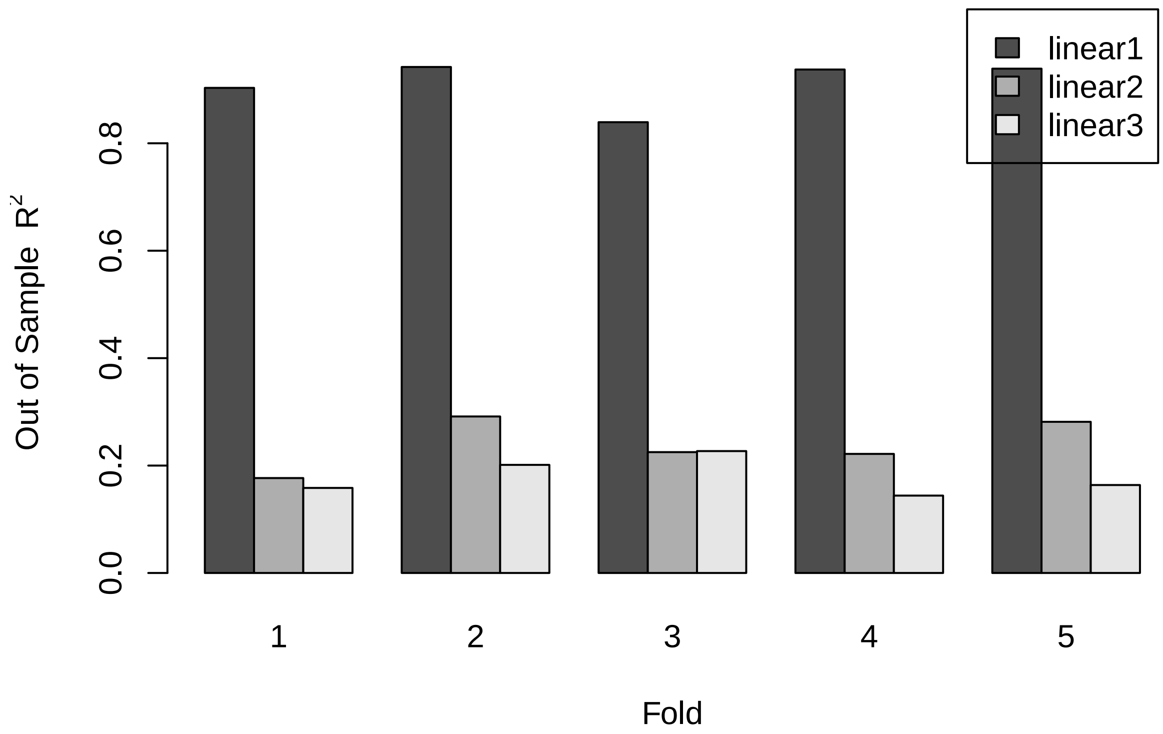
```r
for(k in 1:nfold){
  train <- which(foldid!=k) # train on all but fold `k'
  colnames(bag_data)
  model.linear1 <-lm(soldprice_USD ~ auction_year + special_leather +
                       birkin + upperestimate_USD + Hermes + Black,
                     data = bag_data,  subset=train)
  model.linear2 <-lm(soldprice_USD ~ special_leather +
                       birkin + Hermes + Black,
                     data = bag_data,  subset=train)
  model.linear3 <-lm(soldprice_USD ~ birkin + Hermes + Black,
                     data = bag_data,  subset=train)

  pred.linear1 <- predict(model.linear1, newdata=bag_data[-train,])
  pred.linear2 <- predict(model.linear2, newdata=bag_data[-train,])
  pred.linear3 <- predict(model.linear3, newdata=bag_data[-train,])

  ## calculate and log R2
  OOS$linear1[k] <- R2(y=bag_data$soldprice_USD[-train], pred=pred.linear1, family = "gaussian")
  OOS$linear1[k]
  OOS$linear2[k] <- R2(y=bag_data$soldprice_USD[-train], pred=pred.linear2, family = "gaussian")
  OOS$linear2[k]
  OOS$linear3[k] <- R2(y=bag_data$soldprice_USD[-train], pred=pred.linear3, family = "gaussian")
  OOS$linear3[k]
}
OOS
```

```
##     linear1   linear2   linear3
## 1 0.9031282 0.1767346 0.1583123
## 2 0.9420573 0.2913947 0.2012516
## 3 0.8392498 0.2249945 0.2269542
## 4 0.9373173 0.2216785 0.1440821
## 5 0.9389078 0.2814375 0.1637521
```

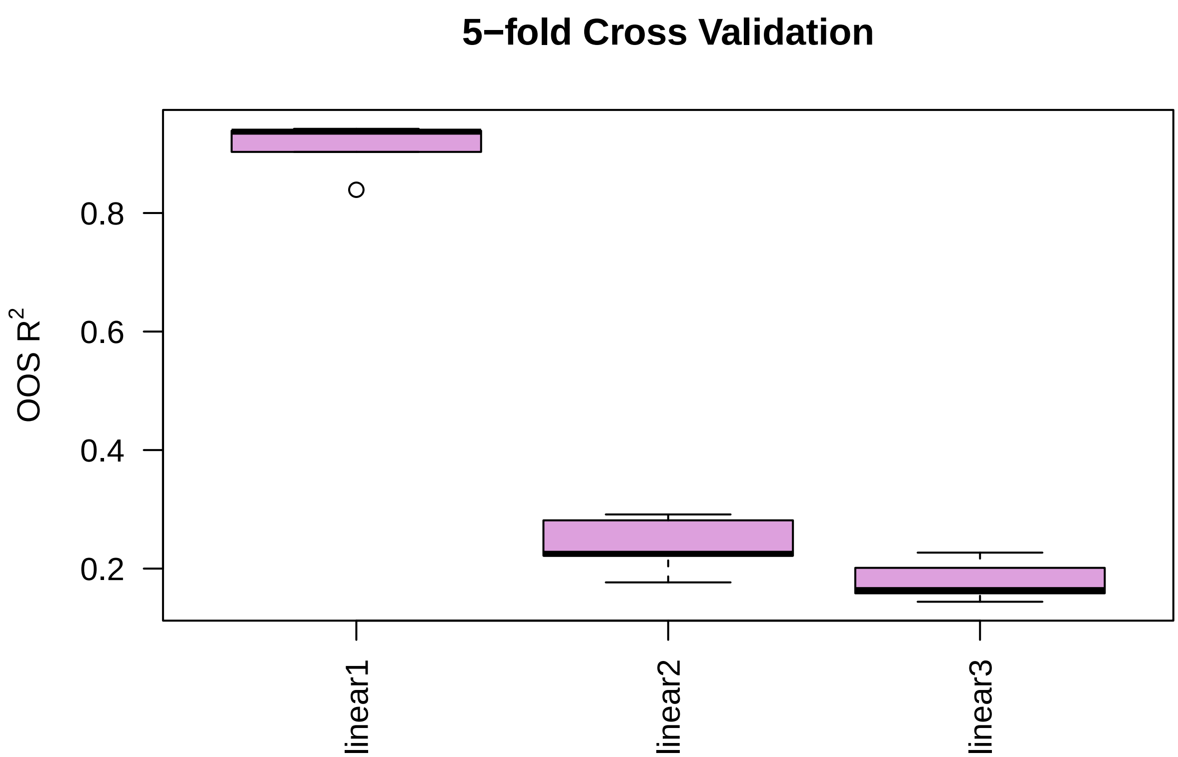
```r
#We have nfold values in OOS for each model, this computes the mean of them
colMeans(OOS)
```

```
##   linear1   linear2   linear3
## 0.9121321 0.2392480 0.1788705
```

```r
m.OOS <- as.matrix(OOS)
rownames(m.OOS) <- c(1:nfold)
barplot(t(as.matrix(OOS)), beside=TRUE, legend=TRUE, args.legend=c(xjust=1, yjust=0.5),
        ylab= bquote( "Out of Sample " ~ R^2), xlab="Fold", names.arg = c(1:5))
```




```
#We then plotted the boxplot and concluded that model.linear1 performs the best, as shown in Figure 9
if (nfold >= 5){
  boxplot(OOS, col="plum", las = 2, ylab=expression(paste("OOS ",R^2)), xlab=c(""), main="5-fold Cross V
}
```




```
###Logistic regression models:
bag_data <- rename(bag_data, oversold = over_sold)
bag_data <- rename(bag_data, location = Location)

model.logistic <- glm(oversold~Hermes + birkin + Black +special_leather + auction_year + location, data
summary(model.logistic)
```

```
##
## Call:
## glm(formula = oversold ~ Hermes + birkin + Black + special_leather +
##     auction_year + location, family = "binomial", data = bag_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1356   0.4644   0.5893   0.8208   1.2905
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -564.38897  510.50486  -1.106   0.2689
## Hermes                0.04730    0.27463   0.172   0.8633
## birkin                0.43961    0.23238   1.892   0.0585 .
## Black                -0.07019    0.32626  -0.215   0.8297
## special_leather      -0.58435    0.24703  -2.365   0.0180 *
## auction_year          0.27970    0.25258   1.107   0.2681
## locationHong Kong    -0.29443    0.32734  -0.899   0.3684
## locationLondon       -0.60716    0.35497  -1.710   0.0872 .
```

```
## locationNew York     0.52371    0.35030   1.495   0.1349
## locationParis       -0.09624    0.52881  -0.182   0.8556
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 838.47  on 758  degrees of freedom
## Residual deviance: 799.96  on 749  degrees of freedom
## AIC: 819.96
##
## Number of Fisher Scoring iterations: 4
```

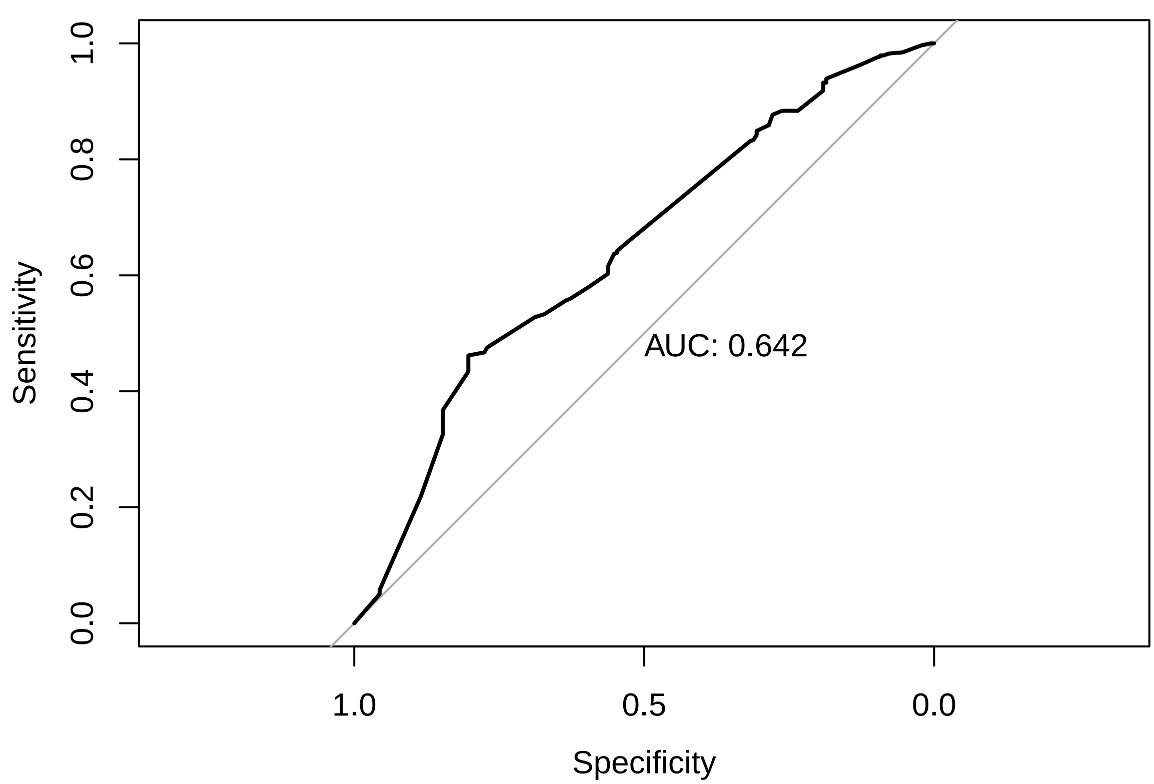
```
#We can compute the R squared
Rsq <- 1 - model.logistic$deviance/model.logistic$null.deviance
Rsq
```

```
## [1] 0.04592939
```

```
#Overall ROC, as shown in Figure 5
test_prob.all = predict(model.logistic, newdata = bag_data, type = "response")
test_roc.all = roc(bag_data$oversold ~ test_prob.all, plot = TRUE, print.auc = TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```r
#We also created two separate datasets for Hermes and Chanel (in Appendix)
hermes <- bag_data[bag_data$brand == "Hermès", ]
chanel <- bag_data[bag_data$brand == "Chanel", ]

#Hermes logistics regression:
model.logistic.hermes <- glm(oversold~ birkin + Black + special_leather + auction_year + location, data
summary(model.logistic.hermes)
```

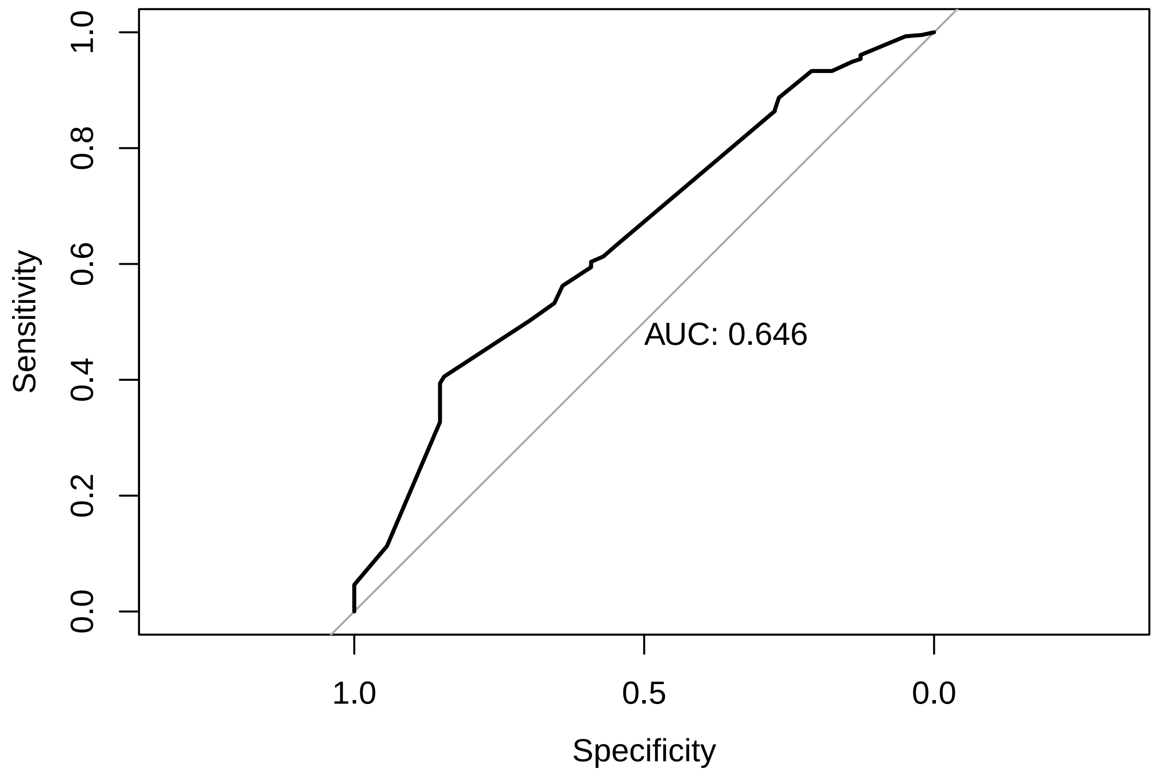
```
##
## Call:
## glm(formula = oversold ~ birkin + Black + special_leather + auction_year +
##     location, family = "binomial", data = hermes)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.12300   0.00026   0.61606   0.84708   1.23283
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -363.7427   785.5103  -0.463   0.6433
## birkin               0.4570     0.2330   1.961   0.0499 *
## Black               15.0945   522.0514   0.029   0.9769
## special_leather     -0.5770     0.2527  -2.283   0.0224 *
## auction_year         0.1804     0.3886   0.464   0.6424
## locationHong Kong   -0.2436     0.3575  -0.681   0.4956
## locationLondon      -0.4559     0.3812  -1.196   0.2317
## locationNew York     0.6016     0.3877   1.552   0.1207
## locationParis        0.0132     0.5714   0.023   0.9816
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 643.38  on 575  degrees of freedom
## Residual deviance: 606.81  on 567  degrees of freedom
## AIC: 624.81
##
## Number of Fisher Scoring iterations: 15
```

```r
#We can compute the R squared
Rsq2 <- 1 - model.logistic.hermes$deviance/model.logistic.hermes$null.deviance
Rsq2
```

```
## [1] 0.05683745
```

```r
#Hermes ROC
test_prob = predict(model.logistic.hermes, newdata = hermes, type = "response")
test_roc = roc(hermes$oversold ~ test_prob, plot = TRUE, print.auc = TRUE)
```

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




```r
###Logistic regression predictions:
#For all bags prediction:
prediction.all <- predict(model.logistic, newdata=bag_prediction, type="response")
prediction.all
```

```
##         2         4         5         6         7         8         9        10
## 0.5818854 0.7948612 0.7139958 0.7139958 0.7139958 0.7139958 0.7139958 0.7948612
##        11        12        13        14        15        16        17        18
## 0.7139958 0.7948612 0.7139958 0.7139958 0.7139958 0.7948612 0.7139958 0.7139958
##        19        20        21        22        23        24        25        26
## 0.7139958 0.7139958 0.7139958 0.7139958 0.7139958 0.7139958 0.7139958 0.7139958
##        27        28        29        30        31        32        33        34
## 0.7139958 0.7948612 0.7139958 0.7139958 0.7139958 0.7948612 0.7948612 0.7948612
##        35        36        37        38        39        40        41        42
## 0.7948612 0.7139958 0.5818854 0.7042402 0.7139958 0.7139958 0.7948612 0.5818854
##        43        44        45        46        47        48        49        50
## 0.7139958 0.7139958 0.7948612 0.5818854 0.7139958 0.6835487 0.7948612 0.7139958
##        51        52        53        54        55        56        57        58
## 0.7948612 0.7139958 0.6835487 0.5818854 0.6835487 0.5818854 0.5818854 0.7831799
##        59        60        61        62        63        64        65        66
## 0.5818854 0.7139958 0.7948612 0.7139958 0.7139958 0.7139958 0.5818854 0.7139958
##        67        68        69        70        71        72        73        74
## 0.5818854 0.7139958 0.7139958 0.5818854 0.7948612 0.7139958 0.7139958 0.7139958
##        75        76        77        78        79        80        81        82
## 0.7139958 0.7139958 0.7139958 0.7139958 0.7042402 0.7948612 0.7948612 0.7139958
##        83        84        85        86        87        88        89        90
```

```
## 0.7042402 0.7139958 0.7139958 0.7139958 0.7139958 0.7042402 0.7139958 0.7139958
##        91        92        93        94
## 0.7948612 0.7139958 0.5818854 0.5818854
```

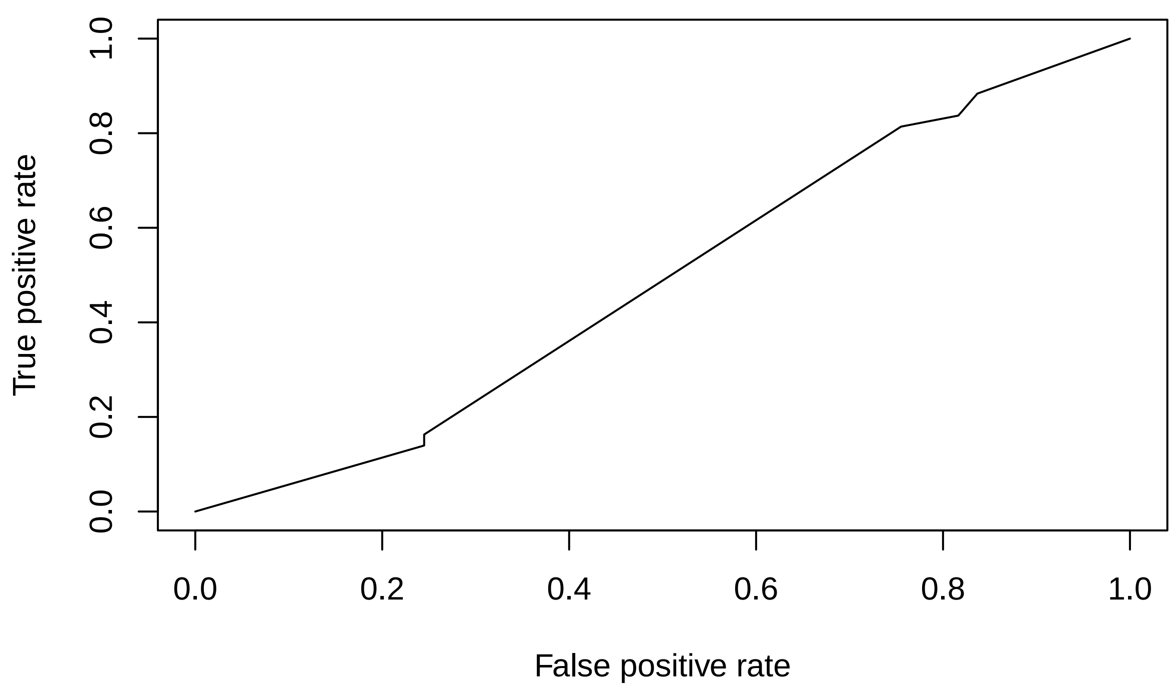
```r
#Combing the predictions with the actual oversold status of the 94 bags:
pred <- prediction(prediction.all,bag_prediction$oversold)
pred
```

```
## A prediction instance
##   with 92 data points
```

```r
#We then try to find the optimal cutoff that maxmized TPR while minimizing FPR
perf <- performance(pred,"tpr","fpr")
perf
```

```
## A performance instance
##   'False positive rate' vs. 'True positive rate' (alpha: 'Cutoff')
##   with 7 data points
```

```r
plot(perf)
```




```r
pred@cutoffs[[1]][which.min(perf@y.values[[1]])]
```

```
##
## Inf
```

```r
opt.cut = function(perf, pred){
  cut.ind = mapply(FUN=function(x, y, p){
    d = (x - 0)^2 + (y-1)^2
    ind = which(d == min(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf@x.values, perf@y.values, pred@cutoffs)
}
optimal.cutoff <- opt.cut(perf, pred)[3]
optimal.cutoff
```

```
## [1] 0.7139958
```

```r
#Using the optimal cutoff, we then proceeded to assign the oversold status to the 94 new bags
predicted_oversold <- ifelse(prediction.all >= optimal.cutoff,1,0)
predicted_oversold
```

```
##  2  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
##  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
##  1  1  1  1  1  1  1  1  0  0  1  1  1  0  1  1  1  0  1  0  1  1  1  1  0  0
## 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
##  0  0  0  1  0  1  1  1  1  1  0  1  0  1  1  0  1  1  1  1  1  1  1  1  0  1
## 81 82 83 84 85 86 87 88 89 90 91 92 93 94
##  1  1  0  1  1  1  1  0  1  1  1  1  0  0
```

```r
comparison <- cbind(predicted_oversold,bag_prediction$oversold)
colnames(comparison) <- c('predicted','actual')
comparison <- data.frame(comparison)

#Calculating accuracy of our prediction:
TP <- sum(comparison$predicted == 1 & comparison$actual == 1)
TP
```

```
## [1] 35
```

```r
TN <- sum(comparison$predicted == 0 & comparison$actual == 0)
TN
```

```
## [1] 12
```

```r
FP <- sum(comparison$predicted == 1 & comparison$actual == 0)
FP
```

```
## [1] 37
```

```r
FN <- sum(comparison$predicted == 0 & comparison$actual == 1)
FN
```

```
## [1] 8
```

```r
accuracy <- (TP + TN) / nrow(bag_prediction)
accuracy
```

```
## [1] 0.5108696
```

```r
#For Hermes bags prediction:
prediction.hermes <- predict(model.logistic.hermes, newdata=bag_prediction[bag_prediction$brand == "Herm
prediction.hermes
```

```
##         2         4         5         6         7         8         9        10
## 0.5654508 0.7853839 0.6985336 0.6985336 0.6985336 0.6985336 0.6985336 0.7853839
##        11        12        13        14        15        16        17        18
## 0.6985336 0.7853839 0.6985336 0.6985336 0.6985336 0.7853839 0.6985336 0.6985336
##        19        20        21        22        23        24        25        26
## 0.6985336 0.6985336 0.6985336 0.6985336 0.6985336 0.6985336 0.6985336 0.6985336
##        27        28        29        30        31        32        33        34
## 0.6985336 0.7853839 0.6985336 0.6985336 0.6985336 0.7853839 0.7853839 0.7853839
##        35        36        37        39        40        41        42        43
## 0.7853839 0.6985336 0.5654508 0.6985336 0.6985336 0.7853839 0.5654508 0.6985336
##        44        45        46        47        48        49        50        51
## 0.6985336 0.7853839 0.5654508 0.6985336 0.6726755 0.7853839 0.6985336 0.7853839
##        52        53        54        55        56        57        58        59
## 0.6985336 0.6726755 0.5654508 0.6726755 0.5654508 0.5654508 0.9999999 0.5654508
##        60        61        62        63        64        65        66        67
## 0.6985336 0.7853839 0.6985336 0.6985336 0.6985336 0.5654508 0.6985336 0.5654508
##        68        69        70        71        72        73        74        75
## 0.6985336 0.6985336 0.5654508 0.7853839 0.6985336 0.6985336 0.6985336 0.6985336
##        76        77        78        80        81        82        84        85
## 0.6985336 0.6985336 0.6985336 0.7853839 0.7853839 0.6985336 0.6985336 0.6985336
##        86        87        89        90        91        92        93        94
## 0.6985336 0.6985336 0.6985336 0.6985336 0.7853839 0.6985336 0.5654508 0.5654508
```

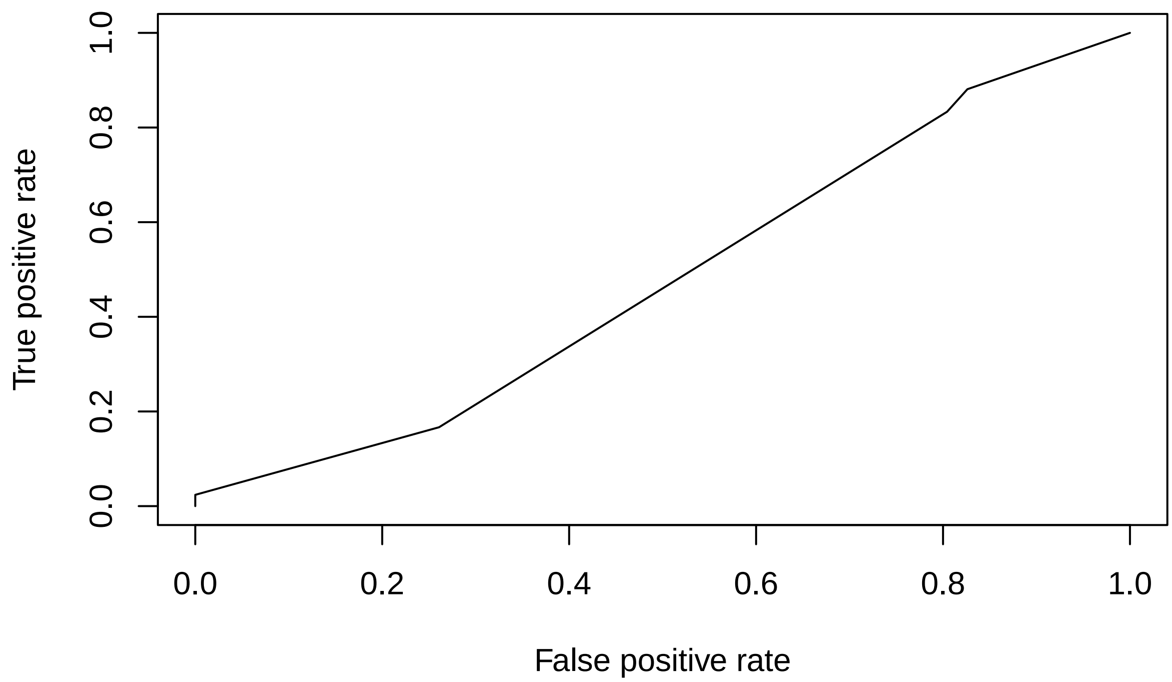
```r
#Combing the predictions with the actual oversold status of the new Hermes bags:
pred.hermes <- prediction(prediction.hermes,bag_prediction$oversold[bag_prediction$brand == "Hermès"])
pred.hermes
```

```
## A prediction instance
##   with 88 data points
```

```r
#We then try to find the optimal cutoff that maxmized TPR while minimizing FPR
perf.hermes <- performance(pred.hermes,"tpr","fpr")
perf.hermes
```

```
## A performance instance
##   'False positive rate' vs. 'True positive rate' (alpha: 'Cutoff')
##   with 6 data points
```

```r
plot(perf.hermes)
```




```
pred.hermes@cutoffs[[1]][which.min(perf.hermes@y.values[[1]])]
```

```
##
## Inf
```

```
opt.cut.hermes = function(perf.hermes, pred.hermes){
  cut.ind = mapply(FUN=function(x, y, p){
    d = (x - 0)^2 + (y-1)^2
    ind = which(d == min(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf.hermes@x.values, perf.hermes@y.values, pred.hermes@cutoffs)
}
optimal.cutoff.hermes <- opt.cut.hermes(perf.hermes, pred.hermes)[3]
optimal.cutoff.hermes
```

```
## [1] 0.6985336
```

```
#Using the optimal cutoff, we then proceeded to assign the oversold status to the 94 new bags
predicted_oversold.hermes <- ifelse(prediction.hermes >= optimal.cutoff.hermes,1,0)
predicted_oversold.hermes
```

```
##  2  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
##  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
```

```
## 29 30 31 32 33 34 35 36 37 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
##  1  1  1  1  1  1  1  1  0  1  1  1  0  1  1  1  0  1  0  1  1  1  1  0  0  0
## 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 80 81 82
##  0  0  1  0  1  1  1  1  1  0  1  0  1  1  0  1  1  1  1  1  1  1  1  1  1  1
## 84 85 86 87 89 90 91 92 93 94
##  1  1  1  1  1  1  1  1  0  0
```

```
comparison.hermes <- cbind(predicted_oversold.hermes,bag_prediction$oversold[bag_prediction$brand == "H
colnames(comparison.hermes) <- c('predicted','actual')
comparison.hermes <- data.frame(comparison.hermes)

#Calculating accuracy of our prediction:
TP.hermes <- sum(comparison.hermes$predicted == 1 & comparison.hermes$actual == 1)
TP.hermes
```

```
## [1] 35
```

```
TN.hermes <- sum(comparison.hermes$predicted == 0 & comparison.hermes$actual == 0)
TN.hermes
```

```
## [1] 9
```

```
FP.hermes <- sum(comparison.hermes$predicted == 1 & comparison.hermes$actual == 0)
FP.hermes
```

```
## [1] 37
```

```
FN.hermes <- sum(comparison.hermes$predicted == 0 & comparison.hermes$actual == 1)
FN.hermes
```

```
## [1] 7
```

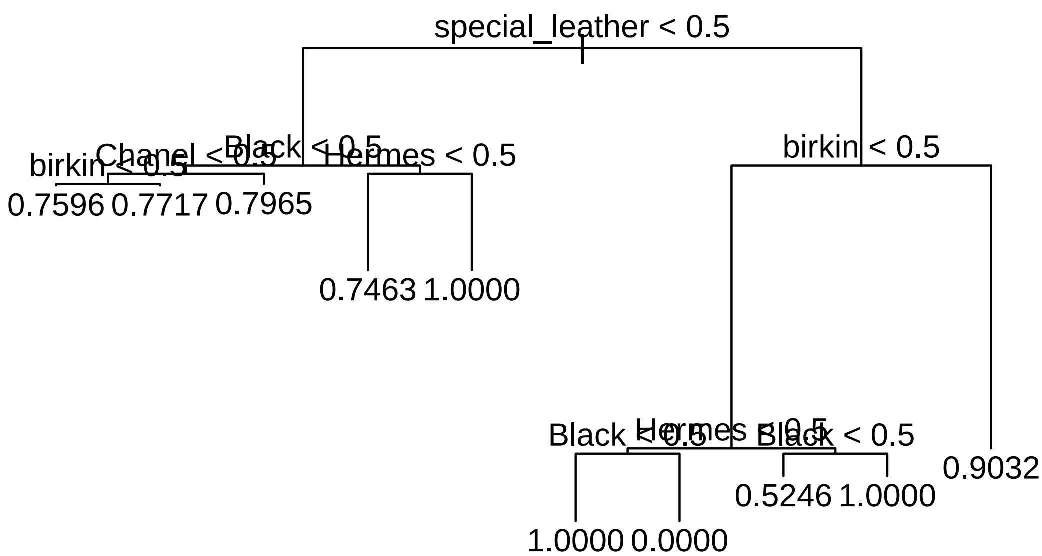
```
accuracy.hermes <- (TP.hermes + TN.hermes) / length(prediction.hermes)
accuracy.hermes
```

```
## [1] 0.5
```

```
###Classification Tree
bagtree <- tree(oversold ~ Hermes + Chanel + Black + birkin + special_leather, data=bag_data, mindev = 
summary(bagtree)
```

```
##
## Regression tree:
## tree(formula = oversold ~ Hermes + Chanel + Black + birkin +
##     special_leather, data = bag_data, mindev = 0, minsize = 2)
## Number of terminal nodes:  10
## Residual mean deviance:  0.1774 = 132.8 / 749
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.9032  0.0000  0.2283  0.0000  0.2404  0.4754
```

```r
#We plotted the tree
plot(bagtree)
#We also added the labels, as shown in Figure 6
text(bagtree, label="yval")
```

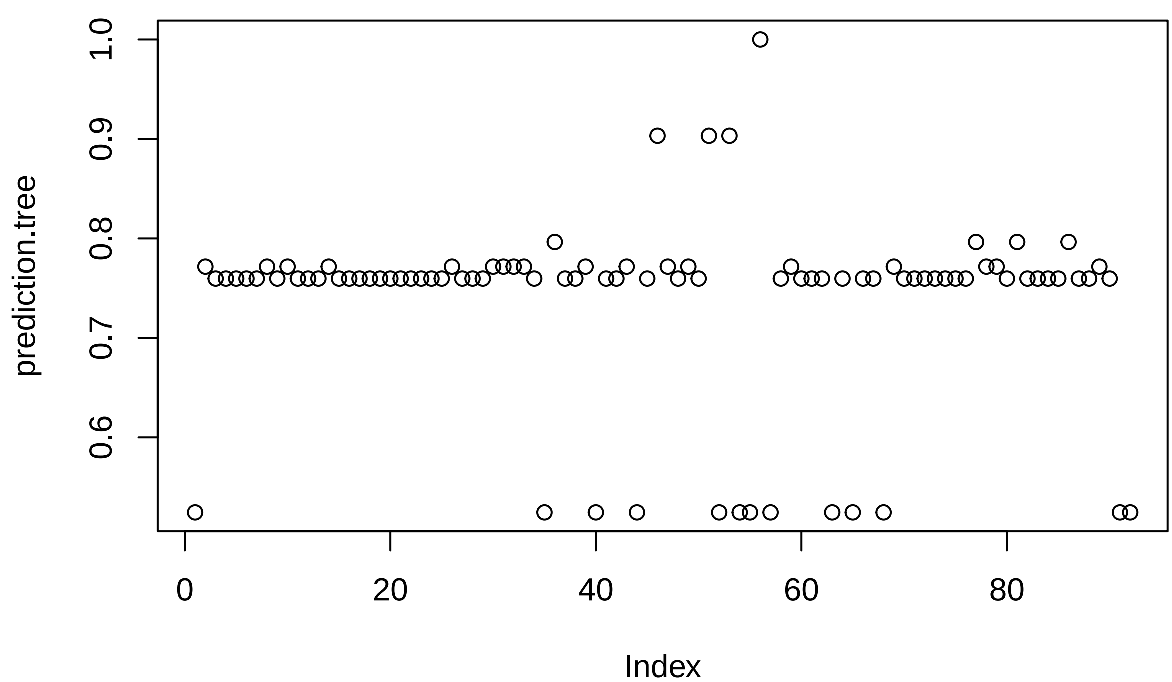



```r
###Classification Tree Prediction
prediction.tree <- predict(bagtree, newdata = bag_prediction, type = "vector")
prediction.tree
```

```
##         2         4         5         6         7         8         9        10
## 0.5245902 0.7716535 0.7596439 0.7596439 0.7596439 0.7596439 0.7596439 0.7716535
##        11        12        13        14        15        16        17        18
## 0.7596439 0.7716535 0.7596439 0.7596439 0.7596439 0.7716535 0.7596439 0.7596439
##        19        20        21        22        23        24        25        26
## 0.7596439 0.7596439 0.7596439 0.7596439 0.7596439 0.7596439 0.7596439 0.7596439
##        27        28        29        30        31        32        33        34
## 0.7596439 0.7716535 0.7596439 0.7596439 0.7596439 0.7716535 0.7716535 0.7716535
##        35        36        37        38        39        40        41        42
## 0.7716535 0.7596439 0.5245902 0.7964602 0.7596439 0.7596439 0.7716535 0.5245902
##        43        44        45        46        47        48        49        50
## 0.7596439 0.7596439 0.7716535 0.5245902 0.7596439 0.9032258 0.7716535 0.7596439
##        51        52        53        54        55        56        57        58
## 0.7716535 0.7596439 0.9032258 0.5245902 0.9032258 0.5245902 0.5245902 1.0000000
##        59        60        61        62        63        64        65        66
## 0.5245902 0.7596439 0.7716535 0.7596439 0.7596439 0.7596439 0.5245902 0.7596439
```

```
##         67         68         69         70         71         72         73         74
## 0.5245902 0.7596439 0.7596439 0.5245902 0.7716535 0.7596439 0.7596439 0.7596439
##         75         76         77         78         79         80         81         82
## 0.7596439 0.7596439 0.7596439 0.7596439 0.7964602 0.7716535 0.7716535 0.7596439
##         83         84         85         86         87         88         89         90
## 0.7964602 0.7596439 0.7596439 0.7596439 0.7596439 0.7964602 0.7596439 0.7596439
##         91         92         93         94
## 0.7716535 0.7596439 0.5245902 0.5245902
```

```
plot(prediction.tree)
```



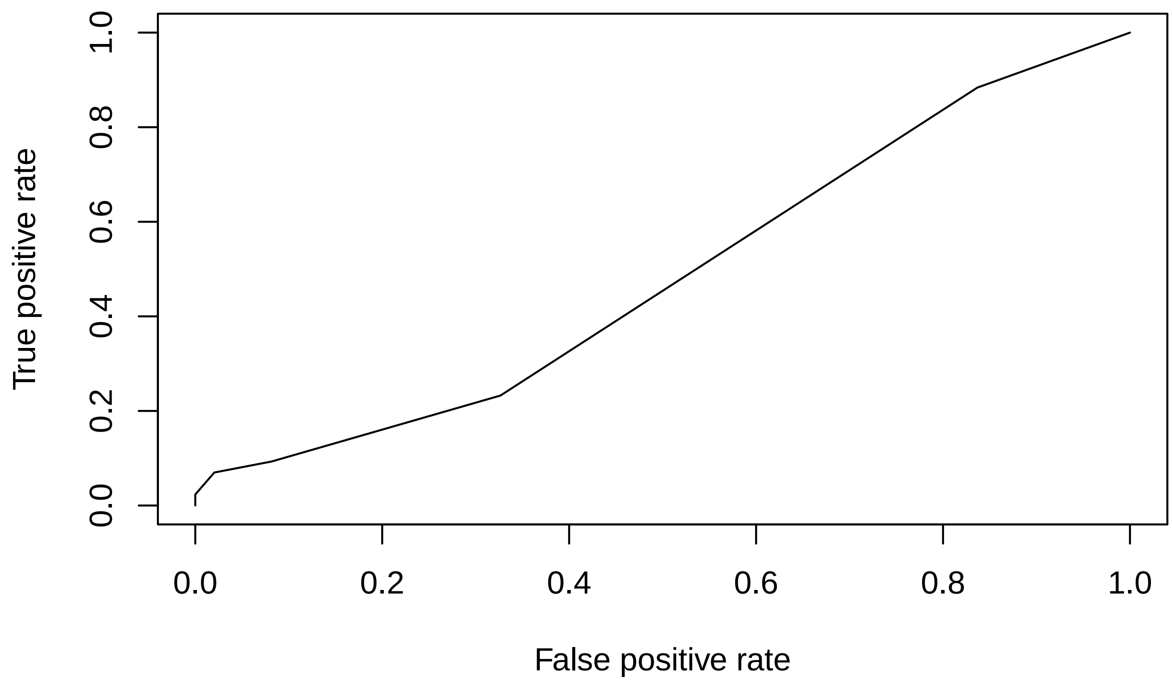

```
#Combing the predictions with the actual oversold status of the 94 bags:
pred.tree <- prediction(prediction.tree,bag_prediction$oversold)
pred.tree
```

```
## A prediction instance
##   with 92 data points
```

```
perf.tree <- performance(pred.tree,"tpr","fpr")
perf.tree
```

```
## A performance instance
##   'False positive rate' vs. 'True positive rate' (alpha: 'Cutoff')
##   with 7 data points
```

```
plot(perf.tree)
```




```
pred.tree@cutoffs[[1]][which.min(perf.tree@y.values[[1]])]
```

```
##
## Inf
```

```
opt.cut.tree = function(perf.tree, pred.tree){
  cut.ind = mapply(FUN=function(x, y, p){
    d = (x - 0)^2 + (y-1)^2
    ind = which(d == min(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf.tree@x.values, perf.tree@y.values, pred.tree@cutoffs)
}
optimal.cutoff.tree <- opt.cut.tree(perf.tree, pred.tree)[3]
optimal.cutoff.tree
```

```
## [1] 0.7716535
```

```
#Using the optimal cutoff, we then proceeded to assign the oversold status to the 94 new bags
predicted_oversold.tree <- ifelse(prediction.tree >= optimal.cutoff.tree,1,0)
predicted_oversold.tree
```

```
##  2  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
##  0  1  0  0  0  0  0  1  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  1
## 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
##  0  0  0  1  1  1  1  0  0  1  0  0  1  0  0  0  1  0  0  1  1  0  1  0  1  0
## 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
##  1  0  0  1  0  0  1  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1  1
## 81 82 83 84 85 86 87 88 89 90 91 92 93 94
##  1  0  1  0  0  0  0  1  0  0  1  0  0  0
```

```
comparison.tree <- cbind(predicted_oversold.tree,bag_prediction$oversold)
colnames(comparison.tree) <- c('predicted','actual')
comparison.tree <- data.frame(comparison.tree)

#Calculating accuracy of our prediction:
TP.tree <- sum(comparison.tree$predicted == 1 & comparison.tree$actual == 1)
TP.tree
```

```
## [1] 10
```

```
TN.tree <- sum(comparison.tree$predicted == 0 & comparison.tree$actual == 0)
TN.tree
```

```
## [1] 33
```

```
FP.tree <- sum(comparison.tree$predicted == 1 & comparison.tree$actual == 0)
FP.tree
```

```
## [1] 16
```

```
FN.tree <- sum(comparison.tree$predicted == 0 & comparison.tree$actual == 1)
FN.tree
```

```
## [1] 33
```

```
accuracy.tree <- (TP.tree + TN.tree) / nrow(bag_prediction)
accuracy.tree
```

```
## [1] 0.4673913
```

```
###Linear Discriminant Analysis

set.seed(555)

#Train & test
temp <- bag_data[,c(8:12,15,16,17)]
temp <- temp %>% drop_na()
temp <- scale(temp)
train_size <- 0.8*nrow(temp)
train_index <- sample(x = 1:nrow(temp), size = train_size, replace = F)
train_set <- as.data.frame(temp[train_index, ])
test_set <- as.data.frame(temp[-train_index, ])

LDA <- lda(oversold ~ lowerestimate_USD + upperestimate_USD + special_leather + birkin + Hermes + Black
LDA
```

```
## Call:
## lda(oversold ~ lowerestimate_USD + upperestimate_USD + special_leather +
##     birkin + Hermes + Black, data = train_set)
##
## Prior probabilities of groups:
## -1.77296193579809  0.56328478168585
##         0.2421746         0.7578254
##
## Group means:
##                   lowerestimate_USD upperestimate_USD special_leather
## -1.77296193579809       -0.13319494       -0.08228658      0.11058885
## 0.56328478168585         0.04309862        0.02521216     -0.04040373
##                        birkin       Hermes       Black
## -1.77296193579809 -0.12237272  0.007035563 -0.06462577
## 0.56328478168585   0.03084894 -0.010619303  0.01809522
##
## Coefficients of linear discriminants:
##                         LD1
## lowerestimate_USD  8.8178495
## upperestimate_USD -8.5052362
## special_leather   -0.4154099
## birkin             0.1321538
## Hermes             0.1108569
## Black              0.1399917
```

```
#In-sample test
LDA_training_pred = predict(LDA, train_set)
LDA_training_pred
```

```
## $class
##   [1] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##   [5] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##   [9] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [13] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [17] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [21] 0.56328478168585  0.56328478168585  -1.77296193579809 0.56328478168585
##  [25] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [29] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [33] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [37] 0.56328478168585  -1.77296193579809 0.56328478168585  0.56328478168585
##  [41] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [45] -1.77296193579809 0.56328478168585  0.56328478168585  0.56328478168585
##  [49] 0.56328478168585  0.56328478168585  0.56328478168585  -1.77296193579809
##  [53] 0.56328478168585  0.56328478168585  0.56328478168585  -1.77296193579809
##  [57] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [61] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [65] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [69] 0.56328478168585  0.56328478168585  0.56328478168585  -1.77296193579809
##  [73] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [77] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [81] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [85] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [89] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
##  [93] 0.56328478168585  0.56328478168585  0.56328478168585  -1.77296193579809
```

```
##  [97] 0.56328478168585  0.56328478168585  -1.77296193579809 0.56328478168585
## [101] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [105] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [109] -1.77296193579809 0.56328478168585  0.56328478168585  0.56328478168585
## [113] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [117] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [121] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [125] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [129] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [133] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [137] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [141] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [145] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [149] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [153] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [157] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [161] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [165] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [169] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [173] 0.56328478168585  0.56328478168585  0.56328478168585  -1.77296193579809
## [177] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [181] -1.77296193579809 0.56328478168585  0.56328478168585  0.56328478168585
## [185] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [189] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [193] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [197] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [201] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [205] 0.56328478168585  0.56328478168585  0.56328478168585  -1.77296193579809
## [209] 0.56328478168585  0.56328478168585  0.56328478168585  -1.77296193579809
## [213] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [217] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [221] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [225] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [229] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [233] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [237] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [241] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [245] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [249] 0.56328478168585  0.56328478168585  -1.77296193579809 0.56328478168585
## [253] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [257] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [261] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [265] 0.56328478168585  0.56328478168585  -1.77296193579809 0.56328478168585
## [269] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [273] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [277] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [281] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [285] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [289] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [293] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [297] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [301] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [305] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [309] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
```

```
## [313] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [317] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [321] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [325] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [329] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [333] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [337] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [341] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [345] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [349] 0.56328478168585  0.56328478168585  -1.77296193579809 0.56328478168585
## [353] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [357] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [361] -1.77296193579809 0.56328478168585  0.56328478168585  -1.77296193579809
## [365] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [369] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [373] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [377] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [381] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [385] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [389] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [393] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [397] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [401] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [405] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [409] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [413] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [417] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [421] 0.56328478168585  -1.77296193579809 0.56328478168585  -1.77296193579809
## [425] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [429] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [433] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [437] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [441] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [445] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [449] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [453] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [457] 0.56328478168585  -1.77296193579809 0.56328478168585  0.56328478168585
## [461] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [465] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [469] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [473] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [477] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [481] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [485] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [489] -1.77296193579809 0.56328478168585  0.56328478168585  0.56328478168585
## [493] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [497] -1.77296193579809 0.56328478168585  0.56328478168585  0.56328478168585
## [501] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [505] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [509] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [513] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [517] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [521] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [525] 0.56328478168585  0.56328478168585  -1.77296193579809 0.56328478168585
```

```
## [529] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [533] -1.77296193579809 0.56328478168585  0.56328478168585  0.56328478168585
## [537] 0.56328478168585  0.56328478168585  0.56328478168585  -1.77296193579809
## [541] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [545] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [549] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [553] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [557] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [561] 0.56328478168585  -1.77296193579809 0.56328478168585  0.56328478168585
## [565] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [569] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [573] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [577] 0.56328478168585  0.56328478168585  -1.77296193579809 0.56328478168585
## [581] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [585] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [589] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [593] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [597] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [601] 0.56328478168585  0.56328478168585  0.56328478168585  0.56328478168585
## [605] 0.56328478168585  0.56328478168585  0.56328478168585
## Levels: -1.77296193579809 0.56328478168585
##
## $posterior
##     -1.77296193579809 0.56328478168585
## 1         0.212600415        0.7873996
## 2         0.237762232        0.7622378
## 3         0.209821685        0.7901783
## 4         0.193029720        0.8069703
## 5         0.175989282        0.8240107
## 6         0.246485804        0.7535142
## 7         0.446861055        0.5531389
## 8         0.223662324        0.7763377
## 9         0.210221272        0.7897787
## 10        0.403776339        0.5962237
## 11        0.217476070        0.7825239
## 12        0.196226666        0.8037733
## 13        0.238636298        0.7613637
## 14        0.297555493        0.7024445
## 15        0.214051931        0.7859481
## 16        0.154172825        0.8458272
## 17        0.178498354        0.8215016
## 18        0.156244495        0.8437555
## 19        0.197467558        0.8025324
## 20        0.193029720        0.8069703
## 21        0.200603220        0.7993968
## 22        0.242476504        0.7575235
## 23        0.795766273        0.2042337
## 24        0.155709004        0.8442910
## 25        0.169139888        0.8308601
## 26        0.032086782        0.9679132
## 27        0.240790140        0.7592099
## 28        0.213536066        0.7864639
## 29        0.212949611        0.7870504
## 30        0.203278092        0.7967219
```

```
## 31          0.227145622        0.7728544
## 32          0.176035444        0.8239646
## 33          0.209821685        0.7901783
## 34          0.175342567        0.8246574
## 35          0.439073972        0.5609260
## 36          0.198914589        0.8010854
## 37          0.302950357        0.6970496
## 38          0.595599527        0.4044005
## 39          0.183749553        0.8162504
## 40          0.212507055        0.7874929
## 41          0.464229135        0.5357709
## 42          0.099393858        0.9006061
## 43          0.138483768        0.8615162
## 44          0.114332712        0.8856673
## 45          0.797210192        0.2027898
## 46          0.031395239        0.9686048
## 47          0.354159092        0.6458409
## 48          0.216700234        0.7832998
## 49          0.136682949        0.8633171
## 50          0.065482532        0.9345175
## 51          0.010447160        0.9895528
## 52          0.754699377        0.2453006
## 53          0.243002289        0.7569977
## 54          0.066673313        0.9333267
## 55          0.221857234        0.7781428
## 56          0.620882901        0.3791171
## 57          0.209491148        0.7905089
## 58          0.282381404        0.7176186
## 59          0.481665117        0.5183349
## 60          0.169139888        0.8308601
## 61          0.195557888        0.8044421
## 62          0.274805305        0.7251947
## 63          0.342849353        0.6571506
## 64          0.229151770        0.7708482
## 65          0.212507055        0.7874929
## 66          0.307637284        0.6923627
## 67          0.196420361        0.8035796
## 68          0.210123067        0.7898769
## 69          0.186615422        0.8133846
## 70          0.067275306        0.9327247
## 71          0.156880562        0.8431194
## 72          0.869631812        0.1303682
## 73          0.183749553        0.8162504
## 74          0.213536066        0.7864639
## 75          0.175989282        0.8240107
## 76          0.258680733        0.7413193
## 77          0.216700234        0.7832998
## 78          0.170807201        0.8291928
## 79          0.214051931        0.7859481
## 80          0.185072201        0.8149278
## 81          0.299280278        0.7007197
## 82          0.115581708        0.8844183
## 83          0.193179827        0.8068202
## 84          0.175989282        0.8240107
```

```
## 85         0.209413998         0.7905860
## 86         0.116778183         0.8832218
## 87         0.211481670         0.7885183
## 88         0.367876008         0.6321240
## 89         0.121852109         0.8781479
## 90         0.197734660         0.8022653
## 91         0.129420610         0.8705794
## 92         0.174554236         0.8254458
## 93         0.216893922         0.7831061
## 94         0.355011517         0.6449885
## 95         0.124624027         0.8753760
## 96         0.554160047         0.4458400
## 97         0.103006777         0.8969932
## 98         0.175849621         0.8241504
## 99         0.792203297         0.2077967
## 100        0.396049741         0.6039503
## 101        0.156880562         0.8431194
## 102        0.237762232         0.7622378
## 103        0.094532682         0.9054673
## 104        0.161225808         0.8387742
## 105        0.184210314         0.8157897
## 106        0.213758040         0.7862420
## 107        0.274805305         0.7251947
## 108        0.211481670         0.7885183
## 109        0.563921288         0.4360787
## 110        0.212949611         0.7870504
## 111        0.100465829         0.8995342
## 112        0.212600415         0.7873996
## 113        0.149465232         0.8505348
## 114        0.285213898         0.7147861
## 115        0.216700234         0.7832998
## 116        0.210221272         0.7897787
## 117        0.169907693         0.8300923
## 118        0.193029720         0.8069703
## 119        0.151592306         0.8484077
## 120        0.200603220         0.7993968
## 121        0.175733265         0.8242667
## 122        0.113143529         0.8868565
## 123        0.317328921         0.6826711
## 124        0.417658198         0.5823418
## 125        0.200226499         0.7997735
## 126        0.242179589         0.7578204
## 127        0.171490503         0.8285095
## 128        0.265283464         0.7347165
## 129        0.355011517         0.6449885
## 130        0.152523904         0.8474761
## 131        0.353532694         0.6464673
## 132        0.212507055         0.7874929
## 133        0.057298934         0.9427011
## 134        0.240790140         0.7592099
## 135        0.363596766         0.6364032
## 136        0.257744268         0.7422557
## 137        0.276681114         0.7233189
## 138        0.202205655         0.7977943
```

```
## 139        0.202937315        0.7970627
## 140        0.154316112        0.8456839
## 141        0.246121896        0.7538781
## 142        0.276226503        0.7237735
## 143        0.129135824        0.8708642
## 144        0.135491903        0.8645081
## 145        0.197508113        0.8024919
## 146        0.056088347        0.9439117
## 147        0.309447794        0.6905522
## 148        0.150926477        0.8490735
## 149        0.175989282        0.8240107
## 150        0.180673825        0.8193262
## 151        0.152056874        0.8479431
## 152        0.312120309        0.6878797
## 153        0.210221272        0.7897787
## 154        0.275056636        0.7249434
## 155        0.202217070        0.7977829
## 156        0.214406391        0.7855936
## 157        0.247127191        0.7528728
## 158        0.469913720        0.5300863
## 159        0.390749909        0.6092501
## 160        0.102443350        0.8975566
## 161        0.192654839        0.8073452
## 162        0.339229996        0.6607700
## 163        0.298027499        0.7019725
## 164        0.102763562        0.8972364
## 165        0.183749553        0.8162504
## 166        0.399766570        0.6002334
## 167        0.197467558        0.8025324
## 168        0.230607817        0.7693922
## 169        0.132822048        0.8671780
## 170        0.238636298        0.7613637
## 171        0.238636298        0.7613637
## 172        0.094532682        0.9054673
## 173        0.304041043        0.6959590
## 174        0.193029720        0.8069703
## 175        0.237048477        0.7629515
## 176        0.563921288        0.4360787
## 177        0.212143415        0.7878566
## 178        0.398253873        0.6017461
## 179        0.469114873        0.5308851
## 180        0.110148942        0.8898511
## 181        0.766928511        0.2330715
## 182        0.131731893        0.8682681
## 183        0.021267779        0.9787322
## 184        0.216648579        0.7833514
## 185        0.203458631        0.7965414
## 186        0.209339534        0.7906605
## 187        0.136682949        0.8633171
## 188        0.377992029        0.6220080
## 189        0.326561379        0.6734386
## 190        0.341153629        0.6588464
## 191        0.203478452        0.7965215
## 192        0.316000709        0.6839993
```

```
## 193        0.129135824        0.8708642
## 194        0.329736673        0.6702633
## 195        0.212507055        0.7874929
## 196        0.355011517        0.6449885
## 197        0.167889367        0.8321106
## 198        0.234695687        0.7653043
## 199        0.237421104        0.7625789
## 200        0.260395467        0.7396045
## 201        0.104502797        0.8954972
## 202        0.097328185        0.9026718
## 203        0.041629945        0.9583701
## 204        0.114332712        0.8856673
## 205        0.192449415        0.8075506
## 206        0.217476070        0.7825239
## 207        0.202205655        0.7977943
## 208        0.615883816        0.3841162
## 209        0.165751221        0.8342488
## 210        0.085819361        0.9141806
## 211        0.446861055        0.5531389
## 212        0.719617828        0.2803822
## 213        0.179205813        0.8207942
## 214        0.171413344        0.8285867
## 215        0.298742551        0.7012574
## 216        0.063725180        0.9362748
## 217        0.341153629        0.6588464
## 218        0.396049741        0.6039503
## 219        0.121852109        0.8781479
## 220        0.300136885        0.6998631
## 221        0.247830213        0.7521698
## 222        0.195557888        0.8044421
## 223        0.247381531        0.7526185
## 224        0.280657100        0.7193429
## 225        0.240136807        0.7598632
## 226        0.289769135        0.7102309
## 227        0.163810452        0.8361895
## 228        0.221857234        0.7781428
## 229        0.247830213        0.7521698
## 230        0.396049741        0.6039503
## 231        0.094532682        0.9054673
## 232        0.210537719        0.7894623
## 233        0.466856948        0.5331431
## 234        0.171413344        0.8285867
## 235        0.422553061        0.5774469
## 236        0.302950357        0.6970496
## 237        0.149465232        0.8505348
## 238        0.160089824        0.8399102
## 239        0.007709259        0.9922907
## 240        0.247707085        0.7522929
## 241        0.161946400        0.8380536
## 242        0.312120309        0.6878797
## 243        0.214163092        0.7858369
## 244        0.181450875        0.8185491
## 245        0.163518533        0.8364815
## 246        0.341153629        0.6588464
```

```
## 247         0.264494400         0.7355056
## 248         0.210221272         0.7897787
## 249         0.224602124         0.7753979
## 250         0.109044080         0.8909559
## 251         0.563921288         0.4360787
## 252         0.302950357         0.6970496
## 253         0.212507055         0.7874929
## 254         0.202205655         0.7977943
## 255         0.217476070         0.7825239
## 256         0.212866308         0.7871337
## 257         0.175337866         0.8246621
## 258         0.155709004         0.8442910
## 259         0.121852109         0.8781479
## 260         0.150926477         0.8490735
## 261         0.035285476         0.9647145
## 262         0.240993908         0.7590061
## 263         0.003895612         0.9961044
## 264         0.200603220         0.7993968
## 265         0.124405073         0.8755949
## 266         0.116778183         0.8832218
## 267         0.626173578         0.3738264
## 268         0.192449415         0.8075506
## 269         0.212949611         0.7870504
## 270         0.282381404         0.7176186
## 271         0.174765462         0.8252345
## 272         0.059063385         0.9409366
## 273         0.149465232         0.8505348
## 274         0.267995420         0.7320046
## 275         0.044783656         0.9552163
## 276         0.212949611         0.7870504
## 277         0.266990070         0.7330099
## 278         0.257744268         0.7422557
## 279         0.350420090         0.6495799
## 280         0.234695687         0.7653043
## 281         0.060965352         0.9390346
## 282         0.214163092         0.7858369
## 283         0.182622660         0.8173773
## 284         0.261802190         0.7381978
## 285         0.209413998         0.7905860
## 286         0.103006777         0.8969932
## 287         0.035285476         0.9647145
## 288         0.228069331         0.7719307
## 289         0.331420147         0.6685799
## 290         0.237762232         0.7622378
## 291         0.191724829         0.8082752
## 292         0.214163092         0.7858369
## 293         0.278802039         0.7211980
## 294         0.222197837         0.7778022
## 295         0.135008726         0.8649913
## 296         0.200603220         0.7993968
## 297         0.366844339         0.6331557
## 298         0.091685748         0.9083143
## 299         0.396049741         0.6039503
## 300         0.192654839         0.8073452
```

```
## 301        0.203458631        0.7965414
## 302        0.214051931        0.7859481
## 303        0.300136885        0.6998631
## 304        0.212143415        0.7878566
## 305        0.176035444        0.8239646
## 306        0.401127852        0.5988721
## 307        0.278185359        0.7218146
## 308        0.354770263        0.6452297
## 309        0.116778183        0.8832218
## 310        0.186615422        0.8133846
## 311        0.170543302        0.8294567
## 312        0.214738024        0.7852620
## 313        0.293076506        0.7069235
## 314        0.293076506        0.7069235
## 315        0.398254135        0.6017459
## 316        0.202205655        0.7977943
## 317        0.193010362        0.8069896
## 318        0.238636298        0.7613637
## 319        0.305010172        0.6949898
## 320        0.214163092        0.7858369
## 321        0.186615422        0.8133846
## 322        0.175989282        0.8240107
## 323        0.170369046        0.8296310
## 324        0.240268151        0.7597318
## 325        0.247297150        0.7527028
## 326        0.209441775        0.7905582
## 327        0.244984771        0.7550152
## 328        0.304041043        0.6959590
## 329        0.341153629        0.6588464
## 330        0.188858036        0.8111420
## 331        0.216700234        0.7832998
## 332        0.212143415        0.7878566
## 333        0.212143415        0.7878566
## 334        0.170807201        0.8291928
## 335        0.393408388        0.6065916
## 336        0.310717976        0.6892820
## 337        0.212673392        0.7873266
## 338        0.088682316        0.9113177
## 339        0.175652483        0.8243475
## 340        0.316997453        0.6830025
## 341        0.187054271        0.8129457
## 342        0.193029720        0.8069703
## 343        0.158102353        0.8418976
## 344        0.175652483        0.8243475
## 345        0.121852109        0.8781479
## 346        0.175337866        0.8246621
## 347        0.063006576        0.9369934
## 348        0.214014282        0.7859857
## 349        0.210537719        0.7894623
## 350        0.293076506        0.7069235
## 351        0.620882901        0.3791171
## 352        0.175652483        0.8243475
## 353        0.195709485        0.8042905
## 354        0.149465232        0.8505348
```

```
## 355        0.113143529       0.8868565
## 356        0.200226499       0.7997735
## 357        0.173949241       0.8260508
## 358        0.247830213       0.7521698
## 359        0.061410568       0.9385894
## 360        0.102763562       0.8972364
## 361        0.756344737       0.2436553
## 362        0.223555761       0.7764442
## 363        0.173949241       0.8260508
## 364        0.531215392       0.4687846
## 365        0.232698051       0.7673019
## 366        0.216893922       0.7831061
## 367        0.174381281       0.8256187
## 368        0.219687661       0.7803123
## 369        0.132822048       0.8671780
## 370        0.491977384       0.5080226
## 371        0.246121896       0.7538781
## 372        0.209413998       0.7905860
## 373        0.189340235       0.8106598
## 374        0.274805305       0.7251947
## 375        0.143135257       0.8568647
## 376        0.248099684       0.7519003
## 377        0.246485804       0.7535142
## 378        0.218376445       0.7816236
## 379        0.244417541       0.7555825
## 380        0.221815641       0.7781844
## 381        0.129135824       0.8708642
## 382        0.212949611       0.7870504
## 383        0.355011517       0.6449885
## 384        0.362244016       0.6377560
## 385        0.076446548       0.9235535
## 386        0.144819930       0.8551801
## 387        0.180153099       0.8198469
## 388        0.310717976       0.6892820
## 389        0.192926835       0.8070732
## 390        0.280470171       0.7195298
## 391        0.067275306       0.9327247
## 392        0.439073972       0.5609260
## 393        0.214075392       0.7859246
## 394        0.224695258       0.7753047
## 395        0.175989282       0.8240107
## 396        0.079083516       0.9209165
## 397        0.234810834       0.7651892
## 398        0.200226499       0.7997735
## 399        0.240136807       0.7598632
## 400        0.202217070       0.7977829
## 401        0.346828436       0.6531716
## 402        0.203478452       0.7965215
## 403        0.276226503       0.7237735
## 404        0.152056874       0.8479431
## 405        0.157694272       0.8423057
## 406        0.132067339       0.8679327
## 407        0.202217070       0.7977829
## 408        0.370652228       0.6293478
```

```
## 409        0.298027499        0.7019725
## 410        0.149465232        0.8505348
## 411        0.206155487        0.7938445
## 412        0.067275306        0.9327247
## 413        0.155709004        0.8442910
## 414        0.207914972        0.7920850
## 415        0.173077581        0.8269224
## 416        0.184671968        0.8153280
## 417        0.210537719        0.7894623
## 418        0.381537531        0.6184625
## 419        0.121852109        0.8781479
## 420        0.175849621        0.8241504
## 421        0.248099684        0.7519003
## 422        0.743241724        0.2567583
## 423        0.186615422        0.8133846
## 424        0.615883816        0.3841162
## 425        0.176035444        0.8239646
## 426        0.213021108        0.7869789
## 427        0.045658446        0.9543416
## 428        0.313746485        0.6862535
## 429        0.167017706        0.8329823
## 430        0.330284223        0.6697158
## 431        0.064307538        0.9356925
## 432        0.209413998        0.7905860
## 433        0.246121896        0.7538781
## 434        0.283717047        0.7162830
## 435        0.139127166        0.8608728
## 436        0.223978523        0.7760215
## 437        0.247830213        0.7521698
## 438        0.214163092        0.7858369
## 439        0.215750970        0.7842490
## 440        0.210123067        0.7898769
## 441        0.198272689        0.8017273
## 442        0.222648208        0.7773518
## 443        0.226376325        0.7736237
## 444        0.199199309        0.8008007
## 445        0.169139888        0.8308601
## 446        0.193029720        0.8069703
## 447        0.222197837        0.7778022
## 448        0.156979689        0.8430203
## 449        0.187575512        0.8124245
## 450        0.116395018        0.8836050
## 451        0.134276248        0.8657238
## 452        0.261540566        0.7384594
## 453        0.227314786        0.7726852
## 454        0.390749909        0.6092501
## 455        0.227314786        0.7726852
## 456        0.198272689        0.8017273
## 457        0.390749909        0.6092501
## 458        0.620882901        0.3791171
## 459        0.402080421        0.5979196
## 460        0.221857234        0.7781428
## 461        0.221857234        0.7781428
## 462        0.226376325        0.7736237
```

```
## 463        0.493511586        0.5064884
## 464        0.096419639        0.9035804
## 465        0.131731893        0.8682681
## 466        0.444963910        0.5550361
## 467        0.396049741        0.6039503
## 468        0.156244495        0.8437555
## 469        0.247127191        0.7528728
## 470        0.210221272        0.7897787
## 471        0.197626894        0.8023731
## 472        0.212686471        0.7873135
## 473        0.200072274        0.7999277
## 474        0.149465232        0.8505348
## 475        0.102443350        0.8975566
## 476        0.396049741        0.6039503
## 477        0.006492824        0.9935072
## 478        0.244417541        0.7555825
## 479        0.216700234        0.7832998
## 480        0.231126304        0.7688737
## 481        0.216700234        0.7832998
## 482        0.242830586        0.7571694
## 483        0.240031238        0.7599688
## 484        0.164054090        0.8359459
## 485        0.247830213        0.7521698
## 486        0.211481670        0.7885183
## 487        0.186615422        0.8133846
## 488        0.169139888        0.8308601
## 489        0.661150976        0.3388490
## 490        0.171413344        0.8285867
## 491        0.396049741        0.6039503
## 492        0.214163092        0.7858369
## 493        0.216700234        0.7832998
## 494        0.221728067        0.7782719
## 495        0.216700234        0.7832998
## 496        0.175652483        0.8243475
## 497        0.766928511        0.2330715
## 498        0.156880562        0.8431194
## 499        0.174765462        0.8252345
## 500        0.098375987        0.9016240
## 501        0.245679365        0.7543206
## 502        0.359876982        0.6401230
## 503        0.175989282        0.8240107
## 504        0.061350297        0.9386497
## 505        0.102873856        0.8971261
## 506        0.221857234        0.7781428
## 507        0.203458631        0.7965414
## 508        0.121852109        0.8781479
## 509        0.159519245        0.8404808
## 510        0.238062977        0.7619370
## 511        0.234695687        0.7653043
## 512        0.094396979        0.9056030
## 513        0.078458652        0.9215413
## 514        0.293076506        0.7069235
## 515        0.197508113        0.8024919
## 516        0.211993909        0.7880061
```

```
## 517         0.187575512         0.8124245
## 518         0.169139888         0.8308601
## 519         0.350420090         0.6495799
## 520         0.277399442         0.7226006
## 521         0.175652483         0.8243475
## 522         0.238636298         0.7613637
## 523         0.214406391         0.7855936
## 524         0.093539990         0.9064600
## 525         0.314675494         0.6853245
## 526         0.193029720         0.8069703
## 527         0.873378596         0.1266214
## 528         0.149360042         0.8506400
## 529         0.146178503         0.8538215
## 530         0.338682175         0.6613178
## 531         0.491977384         0.5080226
## 532         0.093548820         0.9064512
## 533         0.527366788         0.4726332
## 534         0.208224971         0.7917750
## 535         0.212949611         0.7870504
## 536         0.197652805         0.8023472
## 537         0.211481670         0.7885183
## 538         0.216700234         0.7832998
## 539         0.200226499         0.7997735
## 540         0.507521207         0.4924788
## 541         0.444963910         0.5550361
## 542         0.160089824         0.8399102
## 543         0.487375801         0.5126242
## 544         0.211985984         0.7880140
## 545         0.331414609         0.6685854
## 546         0.204978167         0.7950218
## 547         0.248009839         0.7519902
## 548         0.198272689         0.8017273
## 549         0.341153629         0.6588464
## 550         0.355011517         0.6449885
## 551         0.207477431         0.7925226
## 552         0.242097638         0.7579024
## 553         0.317328921         0.6826711
## 554         0.208427265         0.7915727
## 555         0.244417541         0.7555825
## 556         0.212507055         0.7874929
## 557         0.367876008         0.6321240
## 558         0.195482369         0.8045176
## 559         0.138483768         0.8615162
## 560         0.239345481         0.7606545
## 561         0.261324935         0.7386751
## 562         0.572886425         0.4271136
## 563         0.149465232         0.8505348
## 564         0.210537719         0.7894623
## 565         0.226825579         0.7731744
## 566         0.221854584         0.7781454
## 567         0.221681943         0.7783181
## 568         0.143259871         0.8567401
## 569         0.246121896         0.7538781
## 570         0.243002289         0.7569977
```

```
## 571        0.180224253        0.8197757
## 572        0.279977580        0.7200224
## 573        0.247830213        0.7521698
## 574        0.221857234        0.7781428
## 575        0.204351164        0.7956488
## 576        0.193179827        0.8068202
## 577        0.155709004        0.8442910
## 578        0.275727043        0.7242730
## 579        0.578877514        0.4211225
## 580        0.209821685        0.7901783
## 581        0.279419556        0.7205804
## 582        0.216700234        0.7832998
## 583        0.248087967        0.7519120
## 584        0.226610357        0.7733896
## 585        0.072210258        0.9277897
## 586        0.236569935        0.7634301
## 587        0.247830213        0.7521698
## 588        0.243002289        0.7569977
## 589        0.198272689        0.8017273
## 590        0.302061759        0.6979382
## 591        0.158102353        0.8418976
## 592        0.190823342        0.8091767
## 593        0.402080421        0.5979196
## 594        0.156880562        0.8431194
## 595        0.199166642        0.8008334
## 596        0.341153629        0.6588464
## 597        0.423533027        0.5764670
## 598        0.174280209        0.8257198
## 599        0.276226503        0.7237735
## 600        0.243285750        0.7567142
## 601        0.212949611        0.7870504
## 602        0.176035444        0.8239646
## 603        0.232698051        0.7673019
## 604        0.204405470        0.7955945
## 605        0.074111200        0.9258888
## 606        0.244984771        0.7550152
## 607        0.341153629        0.6588464
##
## $x
##               LD1
## 1    0.041029589
## 2   -0.155926458
## 3    0.063789708
## 4    0.206309803
## 5    0.360941509
## 6   -0.220804186
## 7   -1.454500692
## 8   -0.047492785
## 9    0.060503212
## 10  -1.213797151
## 11   0.001612536
## 12   0.178477978
## 13  -0.162499450
## 14  -0.573519062
```

```
## 15   0.029226797
## 16   0.577201456
## 17   0.337462620
## 18   0.555640842
## 19   0.167767512
## 20   0.206309803
## 21   0.140926830
## 22  -0.191184123
## 23  -3.601541168
## 24   0.561191423
## 25   0.426407312
## 26   2.903078213
## 27  -0.178626215
## 28   0.033414778
## 29   0.038184799
## 30   0.118277799
## 31  -0.074717369
## 32   0.360507174
## 33   0.063789708
## 34   0.367035799
## 35  -1.411434909
## 36   0.155341695
## 37  -0.608558854
## 38  -2.273967138
## 39   0.289146716
## 40   0.041790742
## 41  -1.550066926
## 42   1.261863973
## 43   0.748729688
## 44   1.047966567
## 45  -3.613696755
## 46   2.933784267
## 47  -0.925798493
## 48   0.007841585
## 49   0.769440302
## 50   1.881752818
## 51   4.464469934
## 52  -3.279218793
## 53  -0.195087343
## 54   1.855421277
## 55  -0.033265921
## 56  -2.418796699
## 57   0.066511764
## 58  -0.472930180
## 59  -1.645527678
## 60   0.426407312
## 61   0.184271615
## 62  -0.421488434
## 63  -0.857821473
## 64  -0.090263209
## 65   0.041790742
## 66  -0.638715007
## 67   0.176802785
## 68   0.061310493
```

```
## 69   0.263228582
## 70   1.842275293
## 71   0.549067850
## 72  -4.335233128
## 73   0.289146716
## 74   0.033414778
## 75   0.360941509
## 76  -0.308965475
## 77   0.007841585
## 78   0.410280548
## 79   0.029226797
## 80   0.277146522
## 81  -0.584760791
## 82   1.031214640
## 83   0.204995205
## 84   0.360941509
## 85   0.067147562
## 86   1.015314074
## 87   0.050166705
## 88  -1.006946750
## 89   0.949413936
## 90   0.165468729
## 91   0.855372722
## 92   0.374488585
## 93   0.006284989
## 94  -0.930881229
## 95   0.914405677
## 96  -2.042438920
## 97   1.207656960
## 98   0.362256108
## 99  -3.571816938
## 100 -1.169861240
## 101  0.549067850
## 102 -0.155926458
## 103  1.337636450
## 104  0.504734868
## 105  0.284958735
## 106  0.031611807
## 107 -0.421488434
## 108  0.050166705
## 109 -2.096474324
## 110  0.038184799
## 111  1.245600415
## 112  0.041029589
## 113  0.627091561
## 114 -0.491946560
## 115  0.007841585
## 116  0.060503212
## 117  0.418965238
## 118  0.206309803
## 119  0.604391804
## 120  0.140926830
## 121  0.363351965
## 122  1.064065037
```

```
## 123 -0.700277182
## 124 -1.292070535
## 125  0.144134766
## 126 -0.188977395
## 127  0.403707556
## 128 -0.355567348
## 129 -0.930881229
## 130  0.594532317
## 131 -0.922059962
## 132  0.041790742
## 133  2.075824501
## 134 -0.178626215
## 135 -0.981773857
## 136 -0.302293787
## 137 -0.434305634
## 138  0.127331590
## 139  0.121150857
## 140  0.575702622
## 141 -0.218129178
## 142 -0.431204276
## 143  0.858825066
## 144  0.783264619
## 145  0.167418328
## 146  2.106715024
## 147 -0.650295365
## 148  0.611469092
## 149  0.360941509
## 150  0.317313670
## 151  0.599468898
## 152 -0.667321209
## 153  0.060503212
## 154 -0.423208889
## 155  0.127235032
## 156  0.026353413
## 157 -0.225512420
## 158 -1.581230825
## 159 -1.139555074
## 160  1.215998323
## 161  0.209596299
## 162 -0.835844351
## 163 -0.576599194
## 164  1.211252715
## 165  0.289146716
## 166 -1.191031685
## 167  0.167767512
## 168 -0.101486432
## 169  0.814629826
## 170 -0.162499450
## 171 -0.162499450
## 172  1.337636450
## 173 -0.615599691
## 174  0.206309803
## 175 -0.150546662
## 176 -2.096474324
```

```
## 177  0.044757790
## 178 -1.182423695
## 179 -1.576854174
## 180  1.105268217
## 181 -3.370937121
## 182  0.827590443
## 183  3.479433266
## 184  0.008256886
## 185  0.116757143
## 186  0.067761392
## 187  0.769440302
## 188 -1.065978442
## 189 -0.757992823
## 190 -0.847538865
## 191  0.116590249
## 192 -0.691901218
## 193  0.858825066
## 194 -0.777646459
## 195  0.041790742
## 196 -0.930881229
## 197  0.438586012
## 198 -0.132733377
## 199 -0.153356658
## 200 -0.321141197
## 201  1.185703164
## 202  1.293648883
## 203  2.534252747
## 204  1.047966567
## 205  0.211399271
## 206  0.001612536
## 207  0.127331590
## 208 -2.389889186
## 209  0.459578089
## 210  1.482661774
## 211 -1.454500692
## 212 -3.031862818
## 213  0.330889276
## 214  0.404448740
## 215 -0.581260015
## 216  1.921438201
## 217 -0.847538865
## 218 -1.169861240
## 219  0.949413936
## 220 -0.590330134
## 221 -0.230663673
## 222  0.184271615
## 223 -0.227377177
## 224 -0.461297171
## 225 -0.173744895
## 226 -0.522292689
## 227  0.478821026
## 228 -0.033265921
## 229 -0.230663673
## 230 -1.169861240
```

```
## 231  1.337636450
## 232  0.057903774
## 233 -1.564478898
## 234  0.404448740
## 235 -1.319488649
## 236 -0.608558854
## 237  0.627091561
## 238  0.516230515
## 239  4.882948706
## 240 -0.229762187
## 241  0.497476688
## 242 -0.667321209
## 243  0.028325311
## 244  0.310162601
## 245  0.481731267
## 246 -0.847538865
## 247 -0.350037715
## 248  0.060503212
## 249 -0.054867489
## 250  1.120718200
## 251 -2.096474324
## 252 -0.608558854
## 253  0.041790742
## 254  0.127331590
## 255  0.001612536
## 256  0.038863131
## 257  0.367080166
## 258  0.561191423
## 259  0.949413936
## 260  0.611469092
## 261  2.768888377
## 262 -0.180146801
## 263  5.819609454
## 264  0.140926830
## 265  0.917146525
## 266  1.015314074
## 267 -2.449552615
## 268  0.211399271
## 269  0.038184799
## 270 -0.472930180
## 271  0.372489081
## 272  2.031881301
## 273  0.627091561
## 274 -0.374492705
## 275  2.430108116
## 276  0.038184799
## 277 -0.367491154
## 278 -0.302293787
## 279 -0.903438142
## 280 -0.132733377
## 281  1.985870358
## 282  0.028325311
## 283  0.299423771
## 284 -0.331091092
```

```
## 285  0.067147562
## 286  1.207656960
## 287  2.768888377
## 288 -0.081887245
## 289 -0.788027326
## 290 -0.155926458
## 291  0.217770571
## 292  0.028325311
## 293 -0.448733225
## 294 -0.035956704
## 295  0.788902015
## 296  0.140926830
## 297 -1.000889264
## 298  1.383647392
## 299 -1.169861240
## 300  0.209596299
## 301  0.116757143
## 302  0.029226797
## 303 -0.590330134
## 304  0.044757790
## 305  0.360507174
## 306 -1.198768754
## 307 -0.444545244
## 308 -0.929443263
## 309  1.015314074
## 310  0.263228582
## 311  0.412824704
## 312  0.023668201
## 313 -0.544148972
## 314 -0.544148972
## 315 -1.182425185
## 316  0.127331590
## 317  0.206479386
## 318 -0.162499450
## 319 -0.621843898
## 320  0.028325311
## 321  0.263228582
## 322  0.360941509
## 323  0.414506351
## 324 -0.174726945
## 325 -0.226758659
## 326  0.066918632
## 327 -0.209753214
## 328 -0.615599691
## 329 -0.847538865
## 330  0.243160157
## 331  0.007841585
## 332  0.044757790
## 333  0.044757790
## 334  0.410280548
## 335 -1.154775063
## 336 -0.658397364
## 337  0.040434787
## 338  1.433601544
```

```
## 339  0.364113117
## 340 -0.698188647
## 341  0.259286921
## 342  0.206309803
## 343  0.536502691
## 344  0.364113117
## 345  0.949413936
## 346  0.367080166
## 347  1.937960439
## 348  0.029532194
## 349  0.057903774
## 350 -0.544148972
## 351 -2.418796699
## 352  0.364113117
## 353  0.182957017
## 354  0.627091561
## 355  1.064065037
## 356  0.144134766
## 357  0.380226150
## 358 -0.230663673
## 359  1.975294175
## 360  1.211252715
## 361 -3.291374379
## 362 -0.046655188
## 363  0.380226150
## 364 -1.916256668
## 365 -0.117511595
## 366  0.006284989
## 367  0.376127232
## 368 -0.016056416
## 369  0.814629826
## 370 -1.701856666
## 371 -0.218129178
## 372  0.067147562
## 373  0.238869047
## 374 -0.421488434
## 375  0.696260398
## 376 -0.232635571
## 377 -0.220804186
## 378 -0.005596339
## 379 -0.205565233
## 380 -0.032937136
## 381  0.858825066
## 382  0.038184799
## 383 -0.930881229
## 384 -0.973789984
## 385  1.654399345
## 386  0.677607753
## 387  0.322119258
## 388 -0.658397364
## 389  0.207211289
## 390 -0.460033444
## 391  1.842275293
## 392 -1.411434909
```

```
## 393  0.029036509
## 394 -0.055597129
## 395  0.360941509
## 396  1.604220836
## 397 -0.133608045
## 398  0.144134766
## 399 -0.173744895
## 400  0.127235032
## 401 -0.881855293
## 402  0.116590249
## 403 -0.431204276
## 404  0.599468898
## 405  0.540690673
## 406  0.823592750
## 407  0.127235032
## 408 -1.023212396
## 409 -0.576599194
## 410  0.627091561
## 411  0.094160397
## 412  1.842275293
## 413  0.561191423
## 414  0.079535588
## 415  0.388520451
## 416  0.280770753
## 417  0.057903774
## 418 -1.086518876
## 419  0.949413936
## 420  0.362256108
## 421 -0.232635571
## 422 -3.196049039
## 423  0.263228582
## 424 -2.389889186
## 425  0.360507174
## 426  0.037602760
## 427  2.402459483
## 428 -0.677642120
## 429  0.447117927
## 430 -0.781025775
## 431  1.908175431
## 432  0.067147562
## 433 -0.218129178
## 434 -0.481911502
## 435  0.741385000
## 436 -0.049976491
## 437 -0.230663673
## 438  0.028325311
## 439  0.015485042
## 440  0.061310493
## 441  0.160845336
## 442 -0.039510128
## 443 -0.068730317
## 444  0.152904784
## 445  0.426407312
## 446  0.206309803
```

```
## 447 -0.035956704
## 448  0.548045439
## 449  0.254614469
## 450  1.020390682
## 451  0.797480651
## 452 -0.329243220
## 453 -0.076031967
## 454 -1.139555074
## 455 -0.076031967
## 456  0.160845336
## 457 -1.139555074
## 458 -2.418796699
## 459 -1.204177669
## 460 -0.033265921
## 461 -0.033265921
## 462 -0.068730317
## 463 -1.710232630
## 464  1.307819696
## 465  0.827590443
## 466 -1.444022802
## 467 -1.169861240
## 468  0.555640842
## 469 -0.225512420
## 470  0.060503212
## 471  0.166395917
## 472  0.040328201
## 473  0.145449364
## 474  0.627091561
## 475  1.215998323
## 476 -1.169861240
## 477  5.118953188
## 478 -0.205565233
## 479  0.007841585
## 480 -0.105470941
## 481  0.007841585
## 482 -0.193813319
## 483 -0.172955290
## 484  0.476395312
## 485 -0.230663673
## 486  0.050166705
## 487  0.263228582
## 488  0.426407312
## 489 -2.657776711
## 490  0.404448740
## 491 -1.169861240
## 492  0.028325311
## 493  0.007841585
## 494 -0.032244727
## 495  0.007841585
## 496  0.364113117
## 497 -3.370937121
## 498  0.549067850
## 499  0.372489081
## 500  1.277451833
```

```
## 501 -0.214872647
## 502 -0.959788743
## 503  0.360941509
## 504  1.976721721
## 505  1.209621172
## 506 -0.033265921
## 507  0.116757143
## 508  0.949413936
## 509  0.522029441
## 510 -0.158189937
## 511 -0.132733377
## 512  1.339801245
## 513  1.615971258
## 514 -0.544148972
## 515  0.167418328
## 516  0.045978724
## 517  0.254614469
## 518  0.426407312
## 519 -0.903438142
## 520 -0.439199663
## 521  0.364113117
## 522 -0.162499450
## 523  0.026353413
## 524  1.353537015
## 525 -0.683525255
## 526  0.206309803
## 527 -4.380892569
## 528  0.628221019
## 529  0.662696540
## 530 -0.832508046
## 531 -1.701856666
## 532  1.353394909
## 533 -1.895177237
## 534  0.076968357
## 535  0.038184799
## 536  0.166172953
## 537  0.050166705
## 538  0.007841585
## 539  0.144134766
## 540 -1.786706982
## 541 -1.444022802
## 542  0.516230515
## 543 -1.676728776
## 544  0.046043468
## 545 -0.787993216
## 546  0.103997916
## 547 -0.231978272
## 548  0.160845336
## 549 -0.847538865
## 550 -0.930881229
## 551  0.083163851
## 552 -0.188368002
## 553 -0.700277182
## 554  0.075294596
```

```
## 555 -0.205565233
## 556  0.041790742
## 557 -1.006946750
## 558  0.184926780
## 559  0.748729688
## 560 -0.167820387
## 561 -0.327719302
## 562 -2.146343900
## 563  0.627091561
## 564  0.057903774
## 565 -0.072228382
## 566 -0.033244972
## 567 -0.031879964
## 568  0.694874441
## 569 -0.218129178
## 570 -0.195087343
## 571  0.321461959
## 572 -0.456700830
## 573 -0.230663673
## 574 -0.033265921
## 575  0.109254175
## 576  0.204995205
## 577  0.561191423
## 578 -0.427793317
## 579 -2.179816688
## 580  0.063789708
## 581 -0.452921207
## 582  0.007841585
## 583 -0.232549859
## 584 -0.070553189
## 585  1.738438844
## 586 -0.146933489
## 587 -0.230663673
## 588 -0.195087343
## 589  0.160845336
## 590 -0.602811981
## 591  0.536502691
## 592  0.225723064
## 593 -1.204177669
## 594  0.549067850
## 595  0.153184242
## 596 -0.847538865
## 597 -1.324967426
## 598  0.377085421
## 599 -0.431204276
## 600 -0.197189269
## 601  0.038184799
## 602  0.360507174
## 603 -0.117511595
## 604  0.108798449
## 605  1.700181953
## 606 -0.209753214
## 607 -0.847538865
```

```r
names(LDA_training_pred)
```

```
## [1] "class"     "posterior" "x"
```

```r
head(LDA_training_pred$class)
```

```
## [1] 0.56328478168585 0.56328478168585 0.56328478168585 0.56328478168585
## [5] 0.56328478168585 0.56328478168585
## Levels: -1.77296193579809 0.56328478168585
```

```r
head(LDA_training_pred$posterior)
```

```
##   -1.77296193579809 0.56328478168585
## 1         0.2126004        0.7873996
## 2         0.2377622        0.7622378
## 3         0.2098217        0.7901783
## 4         0.1930297        0.8069703
## 5         0.1759893        0.8240107
## 6         0.2464858        0.7535142
```

```r
head(LDA_training_pred$x)
```

```
##           LD1
## 1  0.04102959
## 2 -0.15592646
## 3  0.06378971
## 4  0.20630980
## 5  0.36094151
## 6 -0.22080419
```

```r
#Find accuracy of model, 78% correctly predicted
mean(LDA_training_pred$class == train_set$oversold)
```

```
## [1] 0.7545305
```

```r
table(predicted_LDA = LDA_training_pred$class, actual = train_set$oversold)
```

```
##                    actual
## predicted_LDA       -1.77296193579809 0.56328478168585
##   -1.77296193579809                13               15
##   0.56328478168585                134              445
```

```r
#out-sample test
LDA_testing_pred = predict(LDA, test_set)
head(LDA_testing_pred$class)
```

```
## [1] -1.77296193579809 0.56328478168585  0.56328478168585  0.56328478168585
## [5] 0.56328478168585  0.56328478168585
## Levels: -1.77296193579809 0.56328478168585
```

```r
head(LDA_testing_pred$posterior)
```

```
##   -1.77296193579809 0.56328478168585
## 1         0.8774817        0.1225183
## 2         0.4386059        0.5613941
## 3         0.2461219        0.7538781
## 4         0.1756525        0.8243475
## 5         0.2034586        0.7965414
## 6         0.3564183        0.6435817
```

```r
head(LDA_testing_pred$x)
```

```
##          LD1
## 1 -4.4322390
## 2 -1.4088413
## 3 -0.2181292
## 4  0.3641131
## 5  0.1167571
## 6 -0.9392572
```

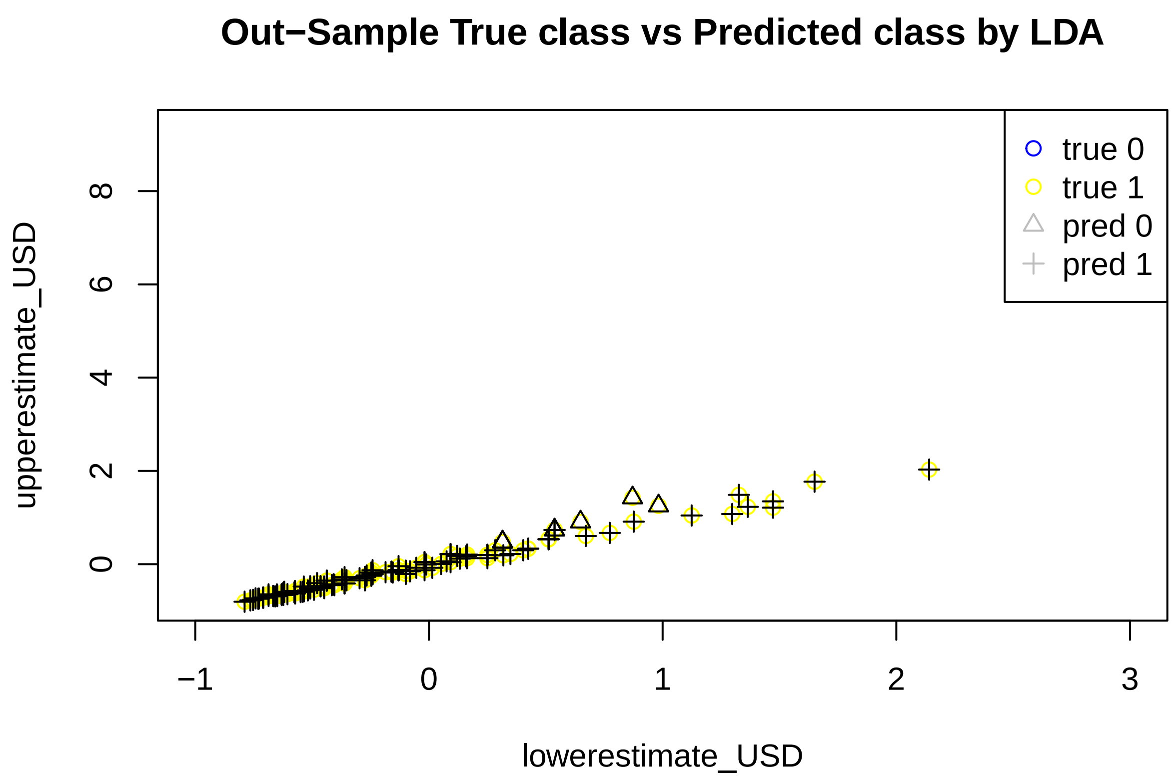
```r
#find accuracy of model, 77% correctly predicted
mean(LDA_testing_pred$class == test_set$oversold)
```

```
## [1] 0.75
```

```r
table(predicted_LDA = LDA_testing_pred$class, actual = test_set$oversold)
```

```
##                      actual
## predicted_LDA          -1.77296193579809 0.56328478168585
##   -1.77296193579809                    4                6
##   0.56328478168585                    32              110
```

```r
plot(test_set$lowerestimate_USD, test_set$upperestimate_USD,
     col = c("blue", "yellow")[test_set$oversold], xlim = c(-1,3),
     xlab = "lowerestimate_USD", ylab = "upperestimate_USD",
     main = "Out-Sample True class vs Predicted class by LDA")
points(test_set$lowerestimate_USD, test_set$upperestimate_USD, pch = c(2,3)[LDA_testing_pred$class])
legend("topright", c("true 0", "true 1",
                     "pred 0", "pred 1"),
       col = c("blue", "yellow", "grey", "grey"), pch = c(1, 1, 2, 3))
```




```r
#Predict new dataset
#Clean data
bag_prediction_test <- bag_prediction[,c(8:12,16,17,18)]
bag_prediction_test <- bag_prediction_test %>% drop_na()
bag_prediction_scale <- as.data.frame(scale(bag_prediction_test))

LDA_new_testing_pred = predict(LDA, bag_prediction_scale)
names(LDA_new_testing_pred)
```

```
## [1] "class"     "posterior" "x"
```

```r
head(LDA_new_testing_pred$class)
```

```
## [1] 0.56328478168585 0.56328478168585 0.56328478168585 0.56328478168585
## [5] 0.56328478168585 0.56328478168585
## Levels: -1.77296193579809 0.56328478168585
```

```r
head(LDA_new_testing_pred$posterior)
```

```
##   -1.77296193579809 0.56328478168585
## 1         0.1911567        0.8088433
## 2         0.3575822        0.6424178
## 3         0.3427573        0.6572427
## 4         0.1603872        0.8396128
```

```
## 5         0.2731471        0.7268529
## 6         0.2268944        0.7731056
```

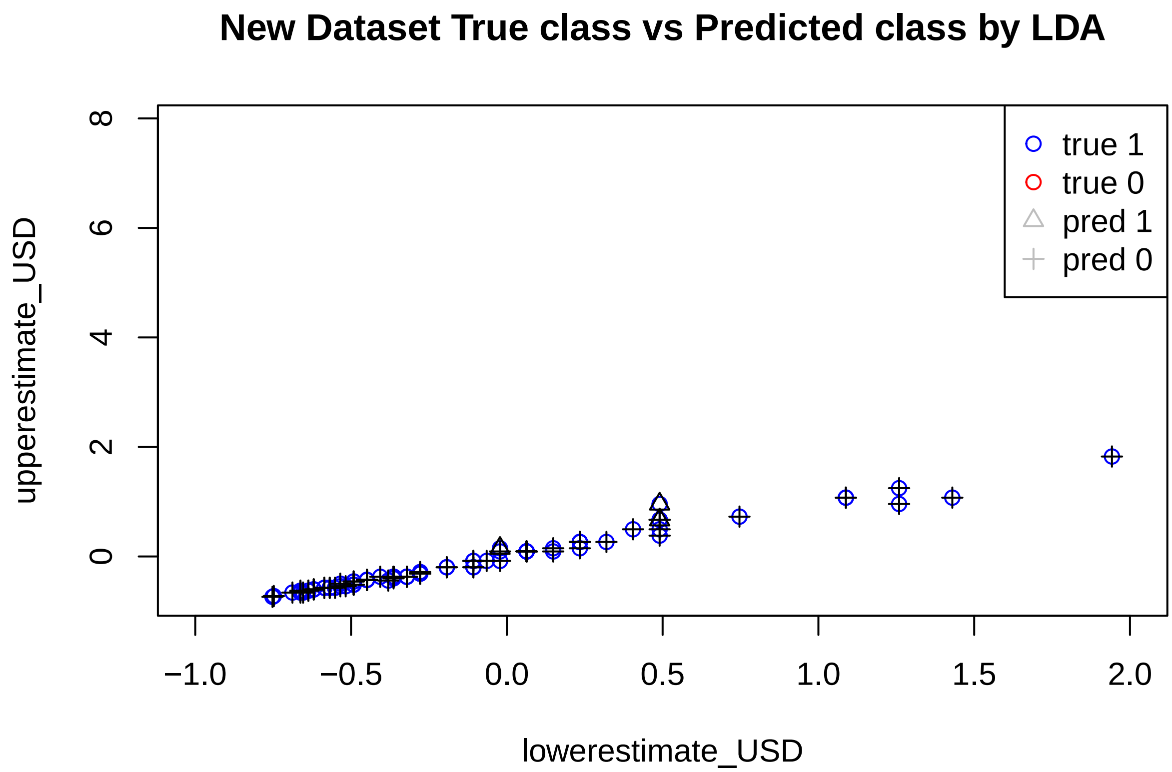
```r
head(LDA_new_testing_pred$x)
```

```
##            LD1
## 1  0.22277902
## 2 -0.94617628
## 3 -0.85726419
## 4  0.51321503
## 5 -0.41011272
## 6 -0.07276407
```

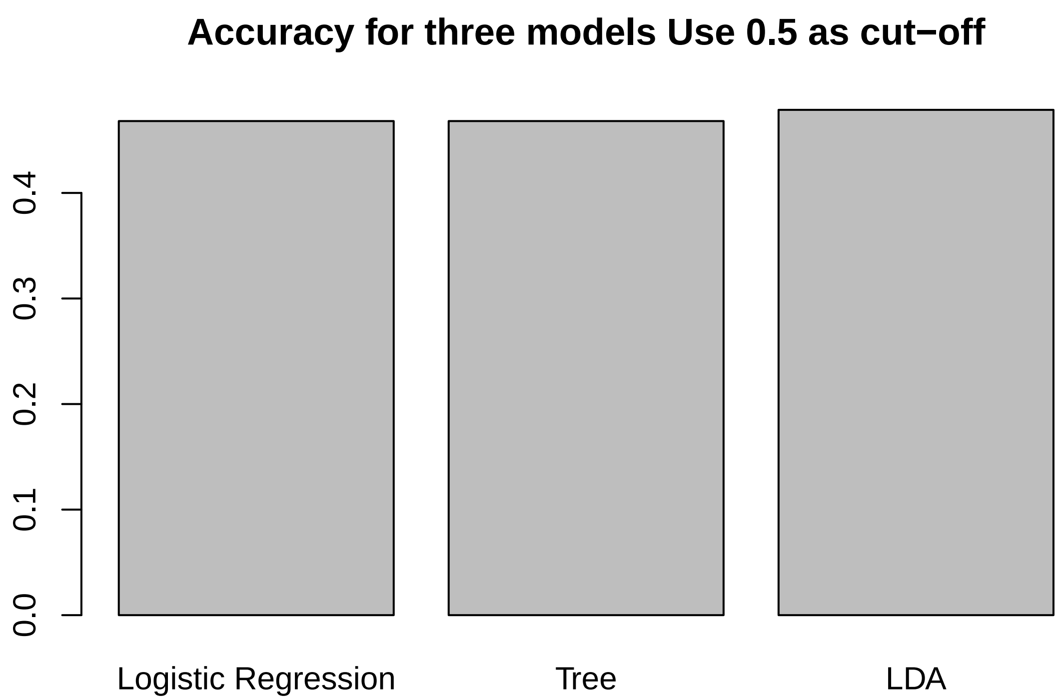
```r
#find accuracy of model

table(predicted_LDA = LDA_new_testing_pred$class, actual = bag_prediction_test$oversold)
```

```
##                     actual
## predicted_LDA         0  1
##   -1.77296193579809  2  1
##   0.56328478168585  47 42
```

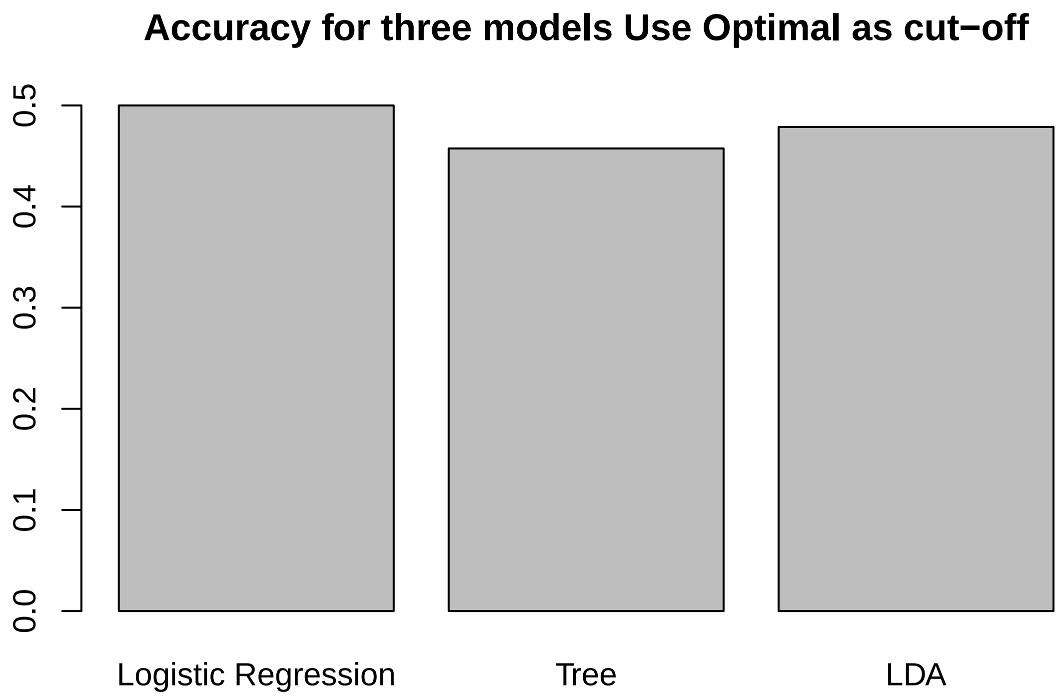
```r
plot(bag_prediction_scale$lowerestimate_USD, bag_prediction_scale$upperestimate_USD,
     col = c("blue", "red")[bag_prediction_test$oversold],
     xlab = "lowerestimate_USD", ylab = "upperestimate_USD", xlim = c(-1,2),
     main = "New Dataset True class vs Predicted class by LDA")
points(bag_prediction_scale$lowerestimate_USD, bag_prediction_scale$upperestimate_USD, pch = c(2,3)[LDA_
legend("topright", c("true 1", "true 0",
                     "pred 1", "pred 0"),
       col = c("blue", "red", "grey", "grey"), pch = c(1, 1, 2, 3))
```




```r
#accuracy table
log_acc <- 44/94
class_acc <- 44/94
lda_acc <- 45/94
barplot(c(log_acc, class_acc,lda_acc), main="Accuracy for three models Use 0.5 as cut-off",
        names.arg=c("Logistic Regression", "Tree", "LDA"))
```




```
log_acc2 <- 47/94
class_acc2 <- 43/94
lda_acc2 <- 45/94
barplot(c(log_acc2, class_acc2,lda_acc2), main="Accuracy for three models Use Optimal as cut-off",
        names.arg=c("Logistic Regression", "Tree", "LDA"))
```


Accuracy for three models Use Optimal as cut−off
0.0
0.1
0.2
0.3
0.4
0.5
Logistic Regression
Tree
LDA