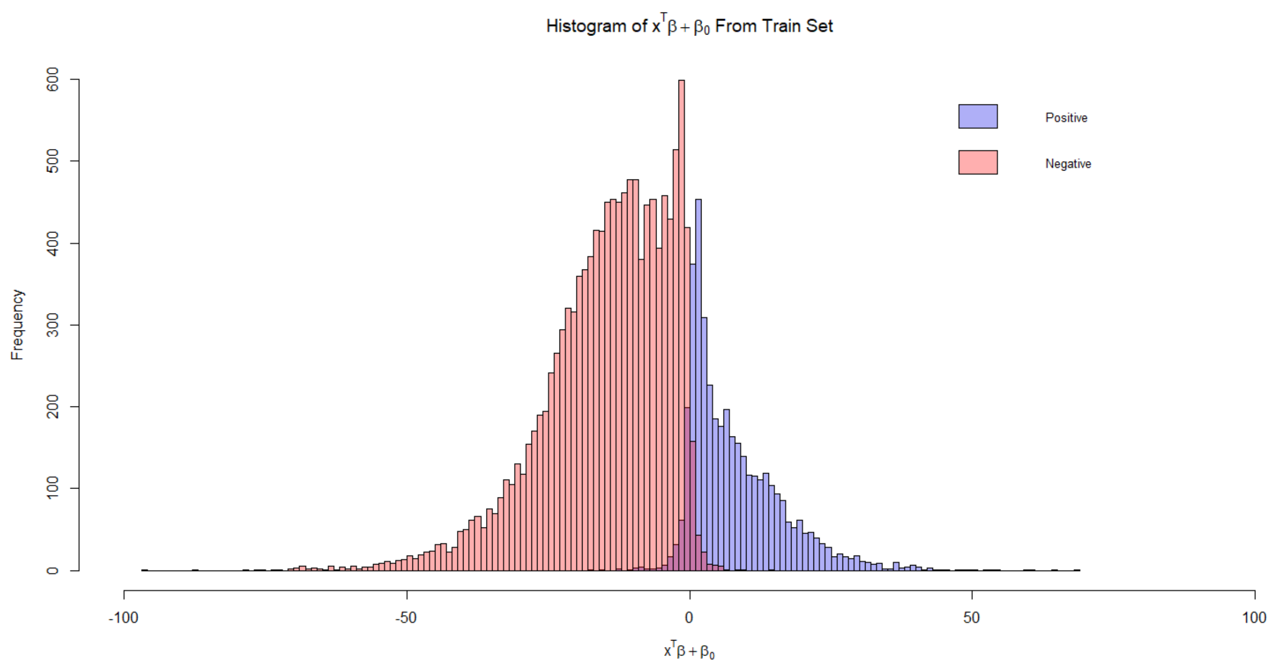
Name: Tanay Mukherjee

STA 9891

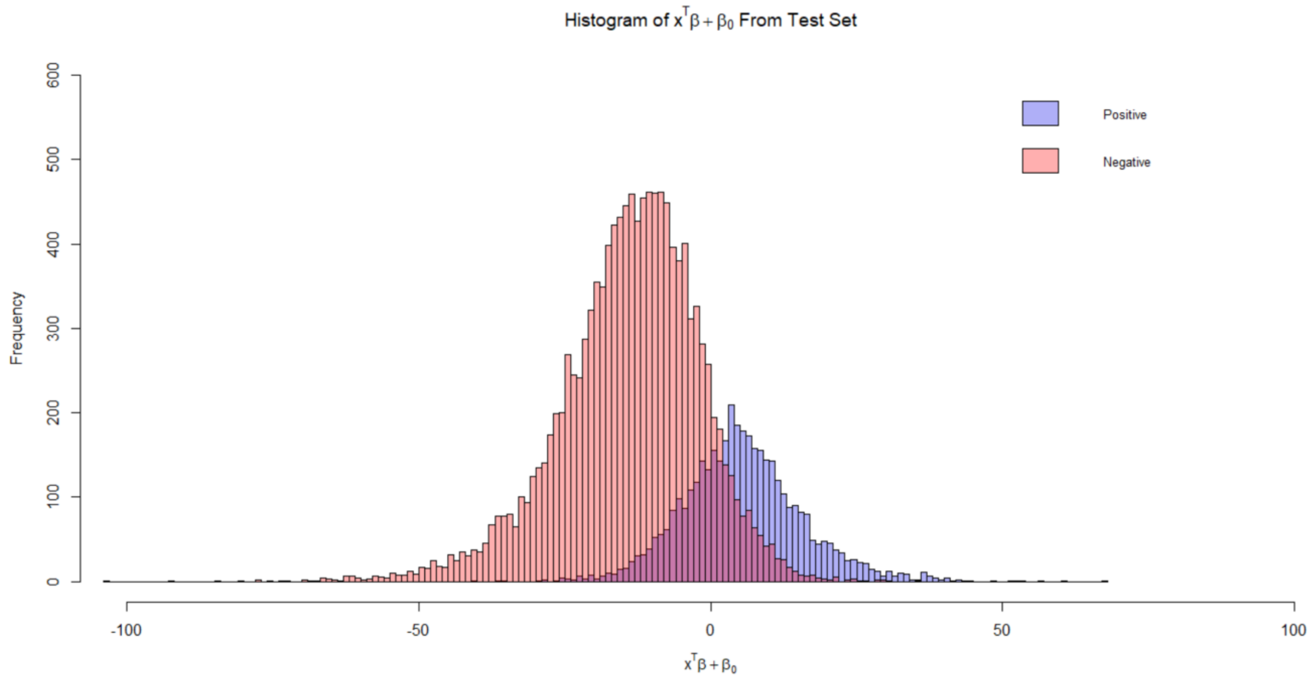
NOTE: Code for replication has been shared in the appendix section.

1. Imbalanced data refers to a classification problem where the number of observations per class is not equally distributed. In this question we subsample the IMDB data to create imbalanced data. To subsample the data, keep all the negative observations, but only keep the first 4000 (out of the 12500) of the positive observations. Do this separately for both train and test. We end with 12500 +4000 = 16500 observations separately for training and testing. Use the $p = 2500$ most frequent words as predictors.

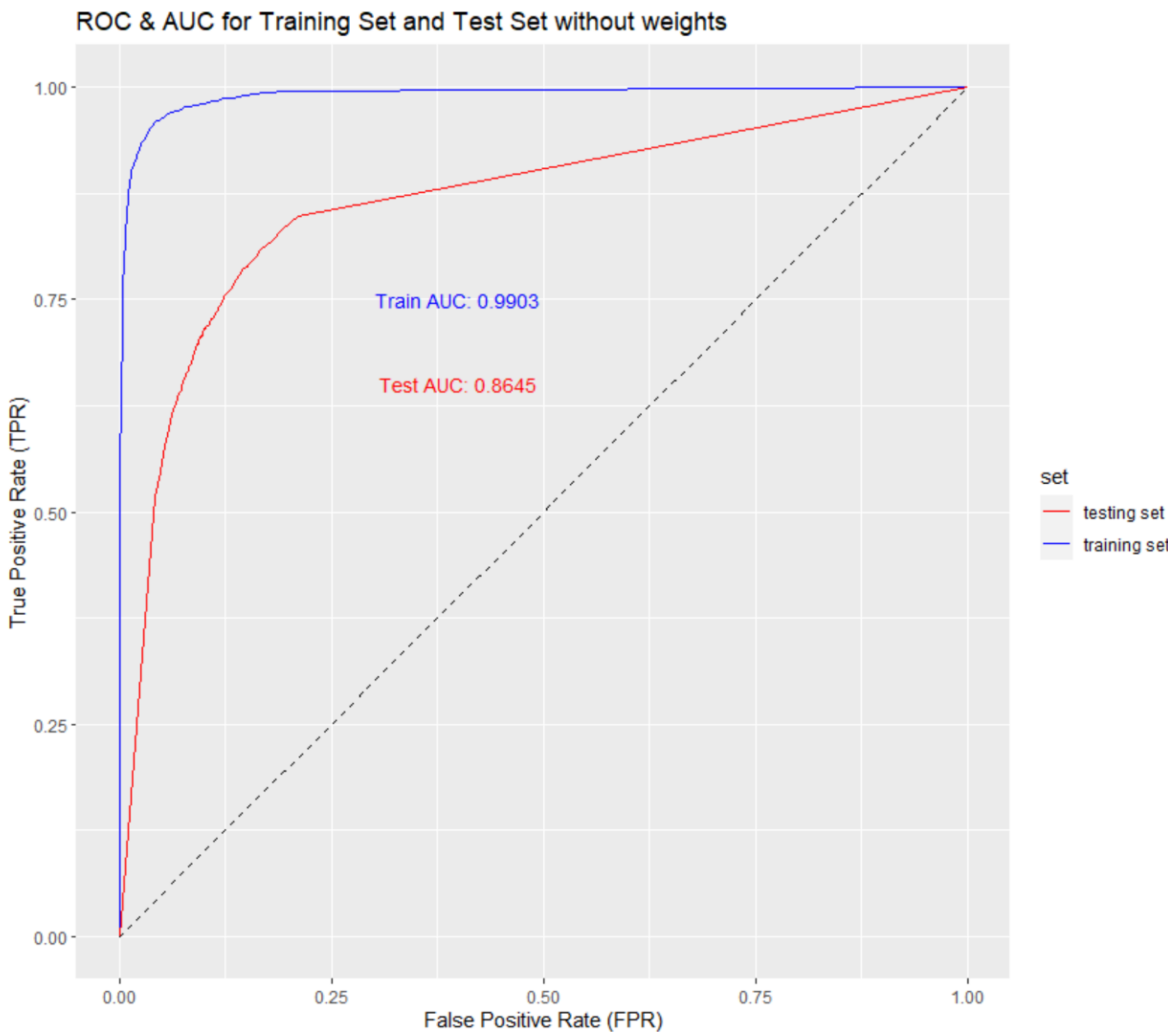
i. For each review in the training set calculate $x_i^\top \hat{\beta} + \hat{\beta}_0$ and two histograms on top of each other with different colors. A histogram of $x_i^\top \hat{\beta} + \hat{\beta}_0$ for positive reviews, and another for negative reviews. (2 points)



ii. For each review in the test set calculate $x_i^\top \hat{\beta} + \hat{\beta}_0$ and two histograms on top of each other with different colors. A histogram of $x_i^\top \hat{\beta} + \hat{\beta}_0$ for positive reviews, and another for negative reviews. (2 points)



iii. In the training set, for each observation, using logistic regression, calculate $\Pr[y = 1|X = x]$. For a sequence of thresholds $\theta = 0, 0.01, 0.02, 0.03, \cdots, 1$, calculate the the TPR and FPR, and using these plot the ROC curve and calculate the AUC. Note that to calculate the AUC you need the area under the ROC curve. Repeat the same for the test set. Plot the ROC for the train and the test on the same graph. Also in the graph report the train and test AUC. In other words, one figure should show the ROC of the train and test, and values of the AUC. Use color coding and make sure to label the horizontal and vertical axes. (2 points)



iv. For $\theta = 0.5$, what is the type I and type II error? (2 points)

1. **For theta = 0.5, the Type I and Type II errors for training set are:**
   a. "Type 1 Error for Training Set: 0.01968"
   b. "Type 2 Error for Training Set: 0.08375"

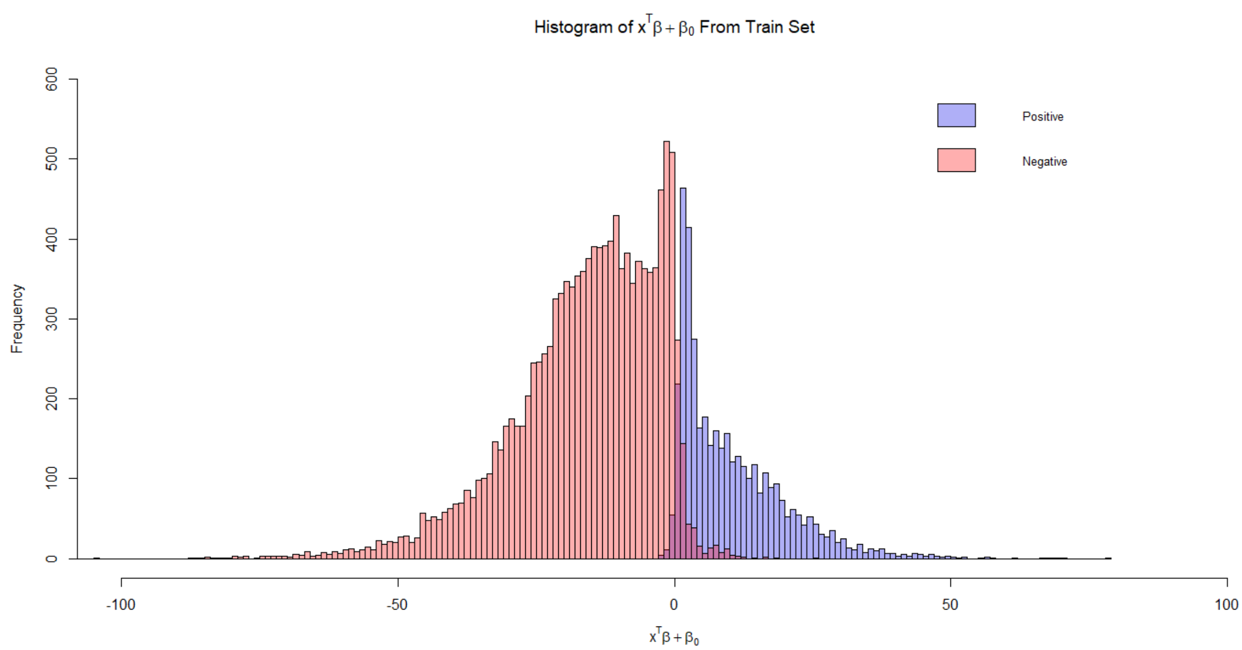
2. **For theta = 0.5, the Type I and Type II errors for test set are:**
   a. "Type 1 Error for Test Set: 0.09872"
   b. "Type 2 Error for Test Set: 0.28925"

v. For what θ, the type I error is equal (as much as possible) to the type II error? (2 points)

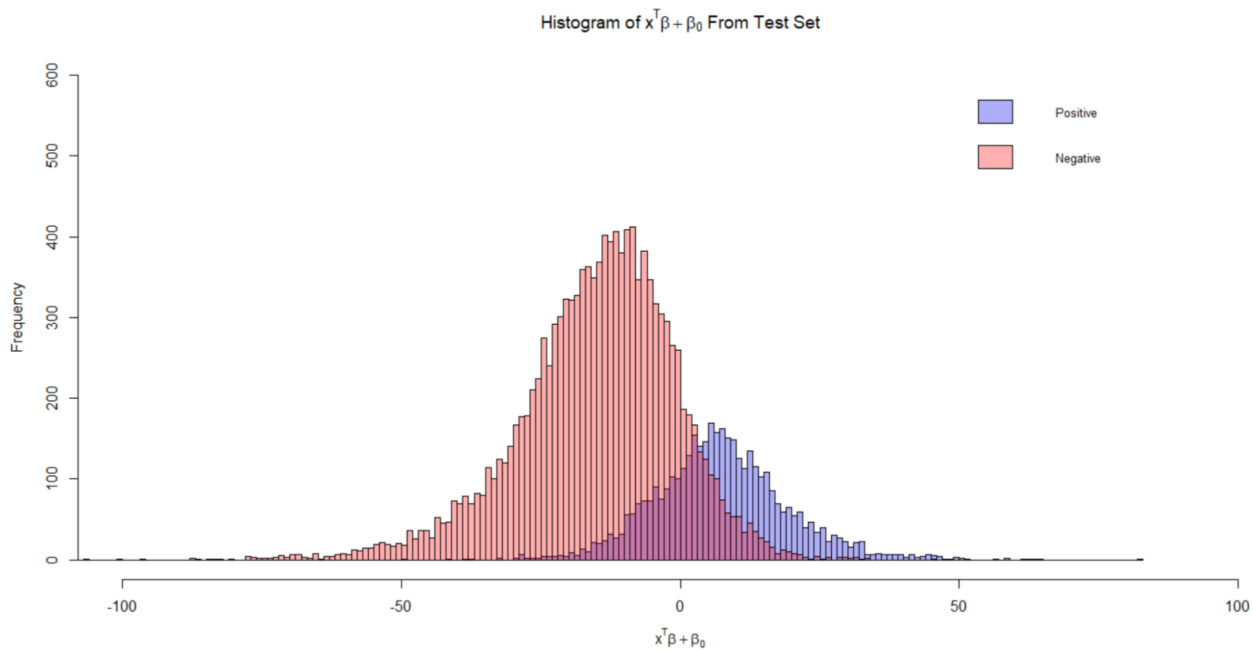
Set	Theta	Index	FPR	TPR
Train	0.32	33	0.04248	0.9585
Test	0.03	105	0.18024	0.8205

2. Fit a logistic regression model to the training data with $n = 16500$ with a twist. Use the "weights" argument in the glmnet function in R (there should be a similar argument in other languages) to place different weights for different observations. Specifically, since the number of positive observations is 40/125 of the number of negative observations, there is danger that the majority of negative reviews will make the classifier more sensitive to positive reviews. **In order to account for this imbalance, use the weight of 1 for positive observations, and use the weight of 40/125 for negative observations.** Let n_+ and n_- stand for the number of positive and negative observations, respectively. Then what we are doing here is essentially giving weight 1 to negative reviews and weight n_+/n_- to positive reviews. The objective here is to see if this procedure of fitting a weighted version can circumvent the problem faced by imbalance, as noted in the unweighted version.

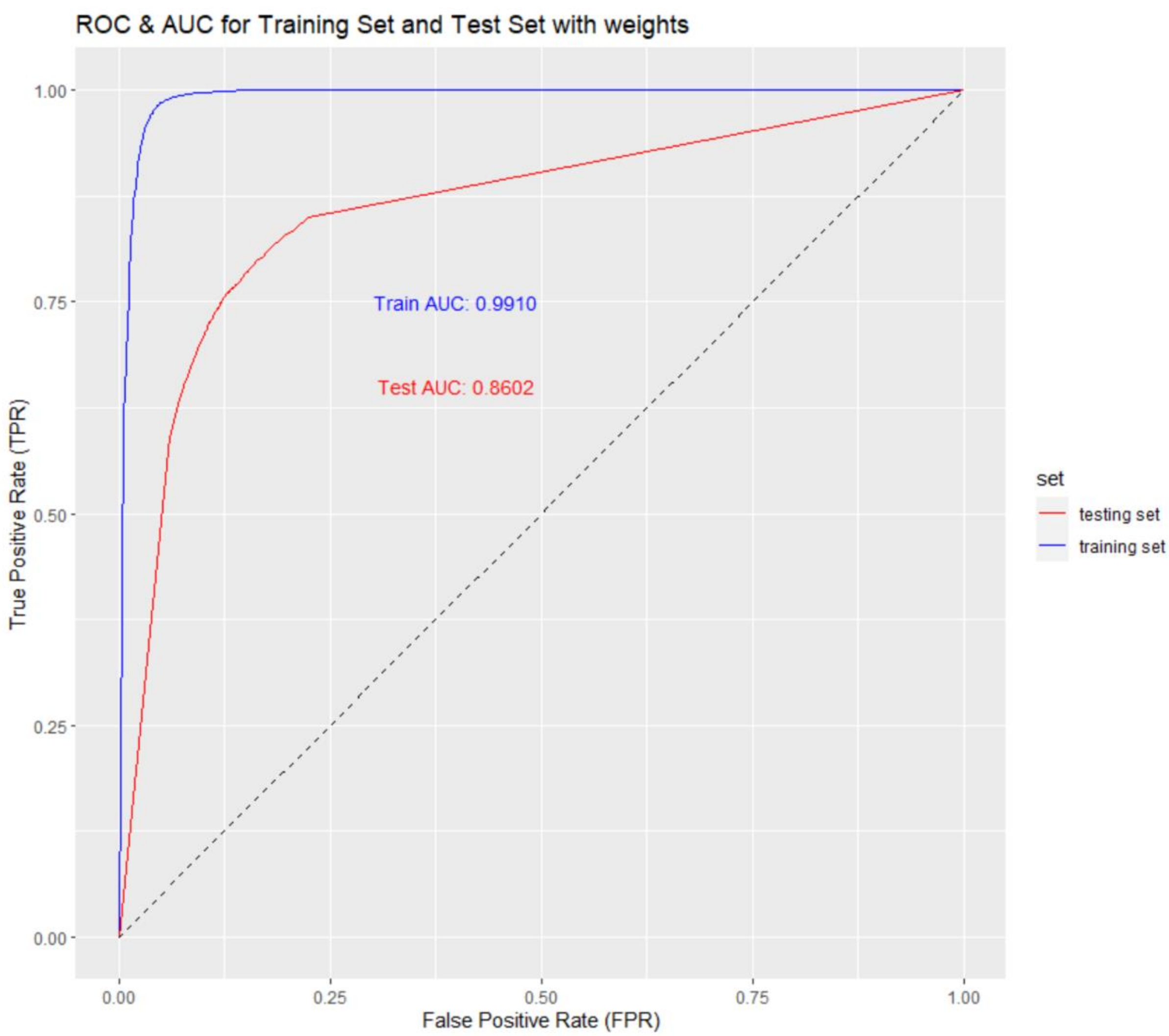
   (a) For each review in the training set calculate $x_i^\top \hat{\beta} + \hat{\beta}_0$ and two histograms on top of each other with different colors. A histogram of $x_i^\top \hat{\beta} + \hat{\beta}_0$ for positive reviews, and another for negative reviews. (2 points)



(b) For each review in the test set calculate $x_i^\top \hat{\beta} + \hat{\beta}_0$ and two histograms on top of each other with different colors. A histogram of $x_i^\top \hat{\beta} + \hat{\beta}_0$ for positive reviews, and another for negative reviews. (2 points)



(c) In the training set, for each observation, using logistic regression, calculate $\Pr[y = 1|X = x]$. For a sequence of thresholds $\theta = 0, 0.01, 0.02, 0.03, \cdots, 1$, calculate the the TPR and FPR, and using these plot the ROC curve and calculate the AUC. Note that to calculate the AUC you need the area under the ROC curve. Repeat the same for the test set. Plot the ROC for the train and the test on the same graph. Also in the graph report the train and test AUC. In other words, one figure should show the ROC of the train and test, and values of the AUC. Use color coding and make sure to label the horizontal and vertical axes. (2 points)



(d) For $\theta = 0.5$, what is the type I and type II error? (2 points)

1. **For theta = 0.5, the Type I and Type II errors for training set are:**
   c. "Type 1 Error for Training Set: 0.04688"
   d. "Type 2 Error for Training Set: 0.0175"

2. **For theta = 0.5, the Type I and Type II errors for test set are:**
   c. "Type 1 Error for Test Set: 0.11896"
   d. "Type 2 Error for Test Set: 0.2545"

(e) For what θ, the type I error is equal (as much as possible) to the type II error? (2 points)

Set	Theta	Index	FPR	TPR
Train	0.61	62	0.03552	0.965
Test	0.05	107	0.18304	0.81725

Appendix

```
rm(list = ls())     #delete objects
cat("\014")         #clear console

library(keras)
library(tensorflow)
library(tidyverse)
library(glmnet)
library(latex2exp)

p                    =     2500
imdb                 =     dataset_imdb(num_words = p, skip_top = 00) #, skip_
top = 10
train_data           =     imdb$train$x
train_labels         =     imdb$train$y
test_data            =     imdb$test$x
test_labels          =     imdb$test$y

numberWords.train    =   max(sapply(train_data, max))
numberWords.test     =   max(sapply(test_data, max))

vectorize_sequences <- function(sequences, dimension = p) {
  results <- matrix(0, nrow = length(sequences), ncol = dimension)
  for (i in 1:length(sequences))
    results[i, sequences[[i]]] <- 1
  results
}

X.train          =        vectorize_sequences(train_data)
X.test           =        vectorize_sequences(test_data)

y.train          =        as.numeric(train_labels)
n.train          =        length(y.train)
y.test           =        as.numeric(test_labels)
n.test           =        length(y.test)

# Sampling
train_sample_ind <- c(which(y.train==0)[1:12500],which(y.train==1)[1:4000])
test_sample_ind <- c(which(y.test==0)[1:12500],which(y.test==1)[1:4000])

X.train <- X.train[train_sample_ind, ]
X.test  <- X.test[test_sample_ind , ]
y.train <- y.train[train_sample_ind]
y.test  <- y.test[test_sample_ind]

fit                  =        glmnet(X.train, y.train, family = "binomial"
, lambda=0.0)
beta0.hat            =        fit$a0
beta.hat             =        as.vector(fit$beta)


## part 1.a - i
distance.P = X.train[y.train == 1, ] %*% beta.hat + beta0.hat
distance.N = X.train[y.train == 0, ] %*% beta.hat + beta0.hat
```

```
breakpoints = pretty((min(c(distance.P,distance.N))-0.001):max(c(distance.P,d
istance.N)),n=150)
hg.pos = hist(distance.P, breaks=breakpoints, plot=FALSE)
hg.neg = hist(distance.N, breaks=breakpoints, plot=FALSE)
color1 = rgb(0,0,230,max = 255, alpha = 80, names = "lt.blue")
color2 = rgb(255,0,0, max = 255, alpha = 80, names = "lt.pink")

plot(hg.pos, ylim = c(0,600), xlim = c(-100,100), col=color1, xlab=TeX('$x^T
\\beta + \\beta_0$'),
     main = TeX('Histogram of $x^T \\beta + \\beta_0$ From Train Set'))

plot(hg.neg, col=color2, add=TRUE)
legend("topright", inset=.02, c("Positive","Negative"), fill=c(color1,color2)
, horiz=FALSE, cex=0.8, box.lty=0)



## part 1.a - ii
distance.P = X.test[y.test == 1, ] %*% beta.hat + beta0.hat
distance.N = X.test[y.test == 0, ] %*% beta.hat + beta0.hat


breakpoints = pretty((min(c(distance.P,distance.N))-0.001):max(c(distance.P,d
istance.N)),n=150)
hg.pos = hist(distance.P, breaks=breakpoints, plot=FALSE)
hg.neg = hist(distance.N, breaks=breakpoints, plot=FALSE)
color1 = rgb(0,0,230,max = 255, alpha = 80, names = "lt.blue")
color2 = rgb(255,0,0, max = 255, alpha = 80, names = "lt.pink")

plot(hg.pos, ylim = c(0,600), xlim = c(-100,100), col=color1, xlab=TeX('$x^T
\\beta + \\beta_0$'),
     main = TeX('Histogram of $x^T \\beta + \\beta_0$ From Train Set'))

plot(hg.neg, col=color2, add=TRUE)
legend("topright", inset=.02, c("Positive","Negative"), fill=c(color1,color2)
, horiz=FALSE, cex=0.8, box.lty=0)



## part 1.a - iii
# To a create a dummy sequence of thresholds (theta) = 0, 0.1, 0.2, 0.3, 0.4,
....,1
thrs_seq   <- c(seq(0,1, by = 0.01))
FPR_train <- TPR_train <- FPR_test <- TPR_test <- rep(0, length(thrs_seq))

prob.train              =        exp(X.train %*% beta.hat + beta0.hat)/(1 + e
xp(X.train %*% beta.hat + beta0.hat))
prob.test               =        exp(X.test  %*% beta.hat + beta0.hat)/(1 + e
xp(X.test  %*% beta.hat + beta0.hat))

for (i in 1:length(thrs_seq)){
  thrs                  =        thrs_seq[i]
  print(paste('For the threshold sequence:',sprintf("%.2f" , thrs_seq[i])))

  # for training set
```

```
  y.hat.train             =        ifelse(prob.train > thrs, 1, 0) #table(y.h
at.train, y.train)
  FP.train                =        sum(y.train[y.hat.train==1] == 0) # false
positives = negatives in the data that were predicted as positive
  TP.train                =        sum(y.hat.train[y.train==1] == 1) # true p
ositives = positives in the data that were predicted as positive
  P.train                 =        sum(y.train==1) # total positives in the d
ata
  N.train                 =        sum(y.train==0) # total negatives in the d
ata
  FPR.train               =        FP.train/N.train # false positive rate = t
ype 1 error = 1 - specificity
  TPR.train               =        TP.train/P.train # true positive rate = 1
- type 2 error = sensitivity = power
  typeI.err.train         =        FPR.train
  typeII.err.train        =        1 - TPR.train
  FPR_train[i]            =        typeI.err.train
  TPR_train[i]            =        1 - typeII.err.train
  print(paste('FPR for training set is',FPR_train[i]))
  print(paste('TPR for training set is',TPR_train[i]))

  # for test set
  y.hat.test              =        ifelse(prob.test > thrs,1,0) #table(y.hat.
test, y.test)
  FP.test                 =        sum(y.test[y.hat.test==1] == 0) # false po
sitives = negatives in the data that were predicted as positive
  TP.test                 =        sum(y.hat.test[y.test==1] == 1) # true pos
itives = positives in the data that were predicted as positive
  P.test                  =        sum(y.test==1) # total positives in the da
ta
  N.test                  =        sum(y.test==0) # total negatives in the da
ta
  TN.test                 =        sum(y.hat.test[y.test==0] == 0)# negatives
in the data that were predicted as negatives
  FPR.test                =        FP.test/N.test # false positive rate = typ
e 1 error = 1 - specificity
  TPR.test                =        TP.test/P.test # true positive rate = 1 -
type 2 error = sensitivity = recall
  typeI.err.test          =        FPR.test
  typeII.err.test         =        1 - TPR.test
  FPR_test[i]             =        typeI.err.test
  TPR_test[i]             =        1 - typeII.err.test
  print(paste('FPR for test set is',FPR_test[i]))
  print(paste('TPR for test set is',TPR_test[i]))
}

train = data.frame(FPR = FPR_train, TPR = TPR_train, Set = 'Train', Threshold
= thrs_seq)
test  = data.frame(FPR = FPR_test,  TPR = TPR_test,  Set = 'Test',  Threshold
= thrs_seq)
df    = rbind(train, test)

# Using colAUC method in catools library to get the AUC value
library(caTools)

# ROC for training set
AUC_train = colAUC(prob.train, y.train, plotROC = F)[1]
```

```r
# ROC for test set
AUC_test = colAUC(prob.test, y.test, plotROC = F)[1]

# Plot the ROC curves
ggplot(data = df, aes(x=FPR, y = TPR, col = Set))+
  geom_line(show.legend = T)+
  labs(title = 'ROC Curves for Training and Test from IMDB dataset', x = 'Fal
se Positive Rate (FPR)', y = 'True Positive Rate (TPR)' ) +
  annotate(geom="text",
           x=c(0.5,0.5),
           y=c(0.4,0.5),
           label=c(paste('AUC of Test: ',round(AUC_test, 3)), paste('AUC of T
rain: ',round(AUC_train, 3))),
           color=c('red', 'blue'))+
  scale_color_manual(values=c('red', 'blue'))




## Part 1.d - iv
print( paste('Type 1 Error for Training Set:',df[df$Threshold == 0.5 & df$Set
=='Train', 'FPR'] ))
print( paste('Type 2 Error for Training Set:',(1 - df[df$Threshold == 0.5 & d
f$Set=='Train', 'TPR'] )))
print( paste('Type 1 Error for Test Set:',df[df$Threshold == 0.5 & df$Set=='T
est', 'FPR'] ))
print( paste('Type 2 Error for Test Set:',(1 - df[df$Threshold == 0.5 & df$Se
t=='Test', 'TPR'] )))



## Part 1.d - v
df_train = df[df$Set == 'Train', ]
df_test = df[df$Set == 'Test', ]

df_train[which.min(abs(df_train$FPR - (1-df_train$TPR))), 'Threshold']
df_test [which.min(abs(df_test$FPR - (1-df_test$TPR))), 'Threshold']

## for train
df_train[which.min(abs(df_train$FPR - (1-df_train$TPR))), ]
## for test
df_test [which.min(abs(df_test$FPR - (1-df_test$TPR))), ]
```

```
## part - 2

## part 2.a

wgt <- c(rep((40/125),12500), rep(1,4000))
fit                     =        glmnet(X.train, y.train, family = "binomial"
, lambda=0.0, weights = wgt)

beta0.hat               =        fit$a0
beta.hat                =        as.vector(fit$beta)


## part 1.a - i
distance.P = X.train[y.train == 1, ] %*% beta.hat + beta0.hat
distance.N = X.train[y.train == 0, ] %*% beta.hat + beta0.hat


breakpoints = pretty((min(c(distance.P,distance.N))-0.001):max(c(distance.P,d
istance.N)),n=150)
hg.pos = hist(distance.P, breaks=breakpoints, plot=FALSE)
hg.neg = hist(distance.N, breaks=breakpoints, plot=FALSE)
color1 = rgb(0,0,230,max = 255, alpha = 80, names = "lt.blue")
color2 = rgb(255,0,0, max = 255, alpha = 80, names = "lt.pink")

plot(hg.pos, ylim = c(0,600), xlim = c(-100,100), col=color1, xlab=TeX('$x^T
\\beta + \\beta_0$'),
     main = TeX('Histogram of $x^T \\beta + \\beta_0$ From Train Set'))

plot(hg.neg, col=color2, add=TRUE)
legend("topright", inset=.02, c("Positive","Negative"), fill=c(color1,color2)
, horiz=FALSE, cex=0.8, box.lty=0)




## part 2.b
distance.P = X.test[y.test == 1, ] %*% beta.hat + beta0.hat
distance.N = X.test[y.test == 0, ] %*% beta.hat + beta0.hat


breakpoints = pretty((min(c(distance.P,distance.N))-0.001):max(c(distance.P,d
istance.N)),n=150)
hg.pos = hist(distance.P, breaks=breakpoints, plot=FALSE)
hg.neg = hist(distance.N, breaks=breakpoints, plot=FALSE)
color1 = rgb(0,0,230,max = 255, alpha = 80, names = "lt.blue")
color2 = rgb(255,0,0, max = 255, alpha = 80, names = "lt.pink")

plot(hg.pos, ylim = c(0,600), xlim = c(-100,100), col=color1, xlab=TeX('$x^T
\\beta + \\beta_0$'),
     main = TeX('Histogram of $x^T \\beta + \\beta_0$ From Train Set'))

plot(hg.neg, col=color2, add=TRUE)
legend("topright", inset=.02, c("Positive","Negative"), fill=c(color1,color2)
, horiz=FALSE, cex=0.8, box.lty=0)
```

```
## part 2.c
# To a create a dummy sequence of thresholds (theta) = 0, 0.1, 0.2, 0.3, 0.4,
....,1
thrs_seq   <- c(seq(0,1, by = 0.01))
FPR_train <- TPR_train <- FPR_test <- TPR_test <- rep(0, length(thrs_seq))

prob.train              =       exp(X.train %*% beta.hat + beta0.hat)/(1 + e
xp(X.train %*% beta.hat + beta0.hat))
prob.test               =       exp(X.test  %*% beta.hat + beta0.hat)/(1 + e
xp(X.test  %*% beta.hat + beta0.hat))

for (i in 1:length(thrs_seq)){
  thrs                    =       thrs_seq[i]
  print(paste('For the threshold sequence:',sprintf("%.2f" , thrs_seq[i])))

  # for training set
  y.hat.train             =       ifelse(prob.train > thrs, 1, 0) #table(y.h
at.train, y.train)
  FP.train                =       sum(y.train[y.hat.train==1] == 0) # false
positives = negatives in the data that were predicted as positive
  TP.train                =       sum(y.hat.train[y.train==1] == 1) # true p
ositives = positives in the data that were predicted as positive
  P.train                 =       sum(y.train==1) # total positives in the d
ata
  N.train                 =       sum(y.train==0) # total negatives in the d
ata
  FPR.train               =       FP.train/N.train # false positive rate = t
ype 1 error = 1 - specificity
  TPR.train               =       TP.train/P.train # true positive rate = 1
- type 2 error = sensitivity = power
  typeI.err.train         =       FPR.train
  typeII.err.train        =       1 - TPR.train
  FPR_train[i]            =       typeI.err.train
  TPR_train[i]            =       1 - typeII.err.train
  print(paste('FPR for training set is',FPR_train[i]))
  print(paste('TPR for training set is',TPR_train[i]))

  # for test set
  y.hat.test              =       ifelse(prob.test > thrs,1,0) #table(y.hat.
test, y.test)
  FP.test                 =       sum(y.test[y.hat.test==1] == 0)  # false po
sitives = negatives in the data that were predicted as positive
  TP.test                 =       sum(y.hat.test[y.test==1] == 1)  # true pos
itives = positives in the data that were predicted as positive
  P.test                  =       sum(y.test==1)  # total positives in the da
ta
  N.test                  =       sum(y.test==0)  # total negatives in the da
ta
  TN.test                 =       sum(y.hat.test[y.test==0] == 0)# negatives
in the data that were predicted as negatives
  FPR.test                =       FP.test/N.test # false positive rate = typ
e 1 error = 1 - specificity
  TPR.test                =       TP.test/P.test # true positive rate = 1 -
type 2 error = sensitivity = recall
  typeI.err.test          =       FPR.test
  typeII.err.test         =       1 - TPR.test
  FPR_test[i]             =       typeI.err.test
```

```
  TPR_test[i]             =        1 - typeII.err.test
  print(paste('FPR for test set is',FPR_test[i]))
  print(paste('TPR for test set is',TPR_test[i]))
}

train = data.frame(FPR = FPR_train, TPR = TPR_train, Set = 'Train', Threshold
= thrs_seq)
test  = data.frame(FPR = FPR_test,  TPR = TPR_test,  Set = 'Test',  Threshold
= thrs_seq)
df    = rbind(train, test)

# Using colAUC method in catools library to get the AUC value
library(caTools)

# ROC for training set
AUC_train = colAUC(prob.train, y.train, plotROC = F)[1]

# ROC for test set
AUC_test = colAUC(prob.test, y.test, plotROC = F)[1]

# Plot the ROC curves
ggplot(data = df, aes(x=FPR, y = TPR, col = Set))+
  geom_line(show.legend = T)+
  labs(title = 'ROC Curves for Training and Test from IMDB dataset', x = 'Fal
se Positive Rate (FPR)', y = 'True Positive Rate (TPR)' ) +
  annotate(geom="text",
           x=c(0.5,0.5),
           y=c(0.4,0.5),
           label=c(paste('AUC of Test: ',round(AUC_test, 3)), paste('AUC of T
rain: ',round(AUC_train, 3))),
           color=c('red', 'blue'))+
  scale_color_manual(values=c('red', 'blue'))


## Part 2.d
print( paste('Type 1 Error for Training Set:',df[df$Threshold == 0.5 & df$Set
=='Train', 'FPR'] ))
print( paste('Type 2 Error for Training Set:',(1 - df[df$Threshold == 0.5 & d
f$Set=='Train', 'TPR'] )))
print( paste('Type 1 Error for Test Set:',df[df$Threshold == 0.5 & df$Set=='T
est', 'FPR'] ))
print( paste('Type 2 Error for Test Set:',(1 - df[df$Threshold == 0.5 & df$Se
t=='Test', 'TPR'] )))


## Part 2.e
df_train = df[df$Set == 'Train', ]
df_test = df[df$Set == 'Test', ]

df_train[which.min(abs(df_train$FPR - (1-df_train$TPR))), 'Threshold']
df_test [which.min(abs(df_test$FPR - (1-df_test$TPR))), 'Threshold']

## for train
df_train[which.min(abs(df_train$FPR - (1-df_train$TPR))), ]
## for test
df_test [which.min(abs(df_test$FPR - (1-df_test$TPR))), ]
```